



**HAL**  
open science

# EdP géométriques pour le traitement et la classification de données sur graphes

Matthieu Toutain

► **To cite this version:**

Matthieu Toutain. EdP géométriques pour le traitement et la classification de données sur graphes. Informatique [cs]. Université de Caen Normandie, 2015. Français. NNT: . tel-01258738

**HAL Id: tel-01258738**

**<https://hal.science/tel-01258738v1>**

Submitted on 19 Jan 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**Université de Caen Normandie**

École doctorale SIMEM

**Thèse de doctorat**

présentée et soutenue le : 11 Décembre 2015

par

**Matthieu Toutain**

pour obtenir le

**Doctorat de l'Université de Caen Normandie**

**Spécialité : Informatique et applications**

**EdP géométriques pour le traitement et la  
classification de données sur graphes**

Directeur de Thèse : Abderrahim Elmoataz

Jury

**Rapporteurs**

Gabriel Peyré	Directeur de recherche, Paris-Dauphine, Paris
Hugues Talbot	Professeur, ESIEE, Paris

**Examineurs**

Mahmoud Melkemi	Professeur, Université de Haute Alsace
Patrice Abry	Directeur de recherche, ENS Lyon
Jalal Fadili	Professeur, Université de Caen
Loïc Simon	Maître de conférences, Université de Caen
Abderrahim Elmoataz	Professeur, Université de Caen





# Remerciements

Les travaux de recherche présentés dans ce manuscrit ont été initiés par Abderrahim Elmoataz qui m'a encadré durant les trois années de cette thèse. Je le remercie pour m'avoir fait découvrir le monde de la recherche. Je le remercie, pour son investissement et ses conseils avisés qui m'ont permis de mener à bien ces trois années de doctorat.

Je tiens à remercier Gabriel Peyré, Hugues Talbot, Mahmoud Melkemi, Patrice Abry, Jalal Fadili et Loïc Simon d'avoir accepté de faire partie de mon jury. Je remercie tout particulièrement Gabriel Peyré et Hugues Talbot pour avoir accepté de rapporter ma thèse. Je les remercie d'avoir consacré de leur temps à la lecture de mon manuscrit. Je tiens aussi à remercier tout particulièrement Loïc Simon d'avoir consacré de son temps à la relecture de mon manuscrit, ainsi que pour ses corrections, discussions très fructueuses et ses conseils.

Je remercie également les membres de l'association Coeur et Cancer ainsi que le conseil régional de Basse-Normandie pour leur soutien financier, sans lequel cette thèse n'aurait jamais pu commencer.

Je remercie tous les membres du GREYC, permanents, doctorants et l'équipe administrative pour leur accueil au sein du laboratoire durant les 5 années passées. Je remercie tout particulièrement les membres du Greyc Poker Tour et les doctorants de l'équipe image pour les moments de détente que nous avons partagé ensemble.

Merci à Alexandra, tous mes proches, mes amis et ma famille qui ont été présents et m'ont soutenu durant ces trois années.



# Table des matières

<b>Introduction générale</b>	<b>1</b>
<b>I Graphes et opérateurs</b>	<b>11</b>
<b>1 Graphes</b>	<b>13</b>
1.1 Définitions et notations utilisées . . . . .	13
1.2 Domaines et constructions de graphe . . . . .	16
1.2.1 Domaines non organisés . . . . .	18
1.2.2 Domaines organisés . . . . .	21
1.3 Conclusion . . . . .	26
<b>2 Calcul discret sur graphe</b>	<b>27</b>
2.1 Opérateurs sur graphe . . . . .	28
2.1.1 Fonctions définies sur un graphe . . . . .	28
2.1.2 Opérateur de différence pondérée . . . . .	29
2.1.3 Opérateur de divergence pondérée . . . . .	29
2.1.4 Opérateur gradient . . . . .	30
2.1.5 Opérateurs $p$ -Laplace . . . . .	30
2.2 Opérateurs directionnels . . . . .	33
2.2.1 Opérateurs de différences directionnelles . . . . .	34
2.2.2 Opérateurs gradients directionnels . . . . .	34
2.2.3 Laplacien infini sur graphe . . . . .	36
2.3 Opérateurs moyennneurs non-locaux . . . . .	38
2.3.1 Régularisation par $p$ -Laplacien . . . . .	38
2.3.2 Opérateurs morphologiques non-locaux . . . . .	40
2.4 Extension des filtres de choc sur graphe . . . . .	40
2.4.1 Filtres de choc . . . . .	41

2.4.2	Extension sur graphe . . . . .	41
2.5	Conclusion . . . . .	42
<b>II Une nouvelle famille de <math>p</math>-Laplacien sur graphe</b>		<b>45</b>
<b>3</b>	<b><math>p</math>-Laplacien normalisé sur graphe pondéré</b>	<b>51</b>
3.1	Introduction . . . . .	52
3.2	$p$ -Laplacien normalisé . . . . .	53
3.2.1	$p$ -Laplacien . . . . .	53
3.2.2	$p$ -Laplacien normalisé . . . . .	54
3.3	$p$ -Laplacien normalisé sur graphe . . . . .	55
3.3.1	Opérateurs statistiques . . . . .	55
3.3.2	Définition du $p$ -Laplacien normalisé sur graphe . . . . .	55
3.3.3	Connexion avec différents opérateurs différentiels locaux et non-locaux . . . . .	56
3.3.4	Relation avec les jeux de type Tug-of-War . . . . .	60
3.4	Problème de Dirichlet associé . . . . .	64
3.4.1	Existence et unicité de la solution . . . . .	64
3.5	Équation de diffusion . . . . .	67
3.5.1	Équation . . . . .	68
3.5.2	Propriétés . . . . .	69
3.6	Expérimentations . . . . .	72
3.6.1	Filtrage et simplification d'image et de données . . . . .	72
3.6.2	Interpolation sur graphe . . . . .	75
3.6.3	Inpainting non-local . . . . .	76
3.7	Conclusion . . . . .	80
<b>4</b>	<b><math>p</math>-Laplacien et Laplacien infini avec termes de gradient</b>	<b>81</b>
4.1	Introduction . . . . .	82
4.2	$p$ -Laplacien non-local . . . . .	83
4.3	Une nouvelle classe de $p$ - et $\infty$ -Laplacien sur graphe . . . . .	84
4.3.1	Définition . . . . .	85
4.3.2	Connexion avec différents opérateurs différentiels . . . . .	87
4.3.3	Relations avec certains jeux de Tug-of-War . . . . .	90
4.4	Équations aux différences partielles associées à l'opérateur proposé	91

4.4.1	Équation de diffusion . . . . .	92
4.4.2	Problème de Dirichlet . . . . .	99
4.5	Application aux problèmes inverses sur graphes pondérés . . . . .	102
4.5.1	Restauration et simplification d'image . . . . .	102
4.5.2	Contours actifs . . . . .	111
4.5.3	Interpolation . . . . .	114
4.6	Conclusion . . . . .	119
<b>5</b>	<b>Équation de Poisson et de Hamilton-Jacobi sur graphe</b>	<b>121</b>
5.1	Introduction . . . . .	122
5.2	Tug-of-War et équation de Poisson infini . . . . .	123
5.3	Équation de Poisson infini non-locale avec termes de gradients . . . . .	125
5.3.1	Du Tug-of-War à l'équation de Poisson sur graphe . . . . .	125
5.3.2	Calcul de distance généralisé sur graphe . . . . .	128
5.4	Applications . . . . .	129
5.4.1	Distances géodésiques pondérées . . . . .	129
5.4.2	Segmentation et classification de données . . . . .	132
5.4.3	Efficacité du schéma de résolution . . . . .	139
5.5	Conclusion . . . . .	139
<b>III Équations d'ensembles de niveaux pour le clustering et la classification de données</b>		<b>141</b>
<b>6</b>	<b>Équations d'ensembles de niveaux pour le clustering et la classification de données</b>	<b>143</b>
6.1	Introduction . . . . .	144
6.2	EDPs géométriques basées sur les ensembles de niveaux sur graphe	145
6.2.1	Formulation continue . . . . .	145
6.2.2	Transposition sur graphe . . . . .	146
6.2.3	Lien avec l'équation eikonale . . . . .	147
6.2.4	Algorithme de propagation de marqueurs utilisant l'équation eikonale sur graphe . . . . .	147
6.3	Processus d'évolution multi-classes . . . . .	150
6.3.1	Équation d'évolution de courbe multi-classes . . . . .	150
6.3.2	Accélération de l'algorithme . . . . .	151

6.4	Classification semi-supervisée . . . . .	156
6.4.1	Méthode d'évaluation . . . . .	156
6.4.2	Construction des graphes . . . . .	157
6.4.3	Fonctions de potentiels . . . . .	158
6.4.4	Influence du paramètre $\sigma$ . . . . .	158
6.4.5	Évaluation et comparaison . . . . .	159
6.4.6	Temps d'exécutions . . . . .	161
6.4.7	Applications en microscopie virtuelle . . . . .	163
6.5	Clustering / Classification non-supervisée . . . . .	169
6.5.1	Algorithme proposé . . . . .	169
6.5.2	Évaluation . . . . .	172
6.6	Conclusion . . . . .	175
	<b>Conclusion générale</b>	<b>177</b>
	<b>Liste des publications</b>	<b>185</b>
	<b>References</b>	<b>187</b>
	<b>Liste des figures</b>	<b>199</b>
	<b>Liste des algorithmes</b>	<b>201</b>

# Introduction générale





---

Les récentes avancées technologiques en termes de procédés d'acquisition et de simulations numériques engendrent actuellement une très grande quantité de données numériques, pouvant être de différents types ou de différentes natures. Ces données peuvent provenir de différents domaines d'applications, telle que l'imagerie numérique, les réseaux complexes (sociaux, biologiques ou informatiques), l'informatique graphique ou encore la bio-informatique. Ces données peuvent être, par exemple, des images, des vidéos, des maillages, des nuages de points, des réseaux ou encore des bases de données.

D'une manière générale, ces données sont de grandes dimensions, en termes de volume ou de description.

Une grande partie de ces données peut être représentée comme des fonctions sur des domaines structurés. C'est le cas par exemple des signaux 1D, des images ou des vidéos. Cependant, une partie de ces données est définie sur des domaines irréguliers ou complexes (au sens de la topologie). On peut par exemple citer les maillages ou les données définies sur des variétés. On peut aussi citer les données définies sur des réseaux ou encore les nuages de points plongés dans un espace de grande dimension. Cet aspect constitue un obstacle majeur à leur traitement.

Néanmoins, ces données étant discrètes par nature, de par leur acquisition ou leur production, les graphes constituent une des structures naturelle adaptée à leur représentation. Les sommets de ce graphe vont alors représenter les données et ses arêtes vont représenter les interactions entre ces données. Ces interactions vont représenter leur similarité et vont dépendre d'une métrique sur cet ensemble de données. Les interactions peuvent alors modéliser une proximité géométrique des données mais aussi d'autres mesures de similarités, en fonction de l'application. Un même ensemble de données peut donc être représenté par différents graphes, chacun possédant un type d'interaction différent. On peut par exemple citer le cas des images, où on peut retrouver différents types d'interaction (locales ou non-locales), selon la construction du graphe, ce qui permet de retrouver aisément des méthodes de traitement d'images locales ou non-locales [GO08, BCM08, PBC08].

L'exploitation et le traitement de ces données représente un défi majeur pour les communautés de traitement d'images et de données. Il existe différentes méthodes de traitement et d'analyse dépendants du type de données et de leur représentation. Nous proposons ici d'adapter des outils ayant fait leurs preuves dans le cadre Euclidien, vers le domaine de représentation commun de ces données : les graphes.

Historiquement, dans le domaine de l'analyse, du traitement, du clustering et de la classification de données sur graphe, plusieurs méthodes ont été proposées. Elles sont basées sur des concepts provenant de la théorie des

graphes, ainsi que la minimisation d'énergies et les propriétés spectrales du Laplacien sur graphes [Chu97]. Les méthodes basées sur la minimisation d'énergies ont principalement été appliquées à la régularisation et à la segmentation de données. On peut par exemple citer, pour la segmentation d'image, les coupes de graphes [BJ01], les marches aléatoires [Gra06] et plus récemment le Power Watershed [CGNT11]. On peut aussi citer des méthodes de classification transductive [BNS06], des méthodes utilisant la minimisation de la variation totale pour approcher les coupes sur graphes [BTUvB13] ainsi que les méthodes de [BF12] utilisant la fonctionnelle de Ginzburg-Landau pour approcher la variation totale, dans le cadre de l'apprentissage. Les approches basées sur la théorie spectrale ont montré leur efficacité en traitement des données pour le débruitage de variétés [HM06], la segmentation d'images [SM00] mais également en classification de données [VL07] ou pour l'extraction de communautés dans des réseaux complexes [KBCL09] et bien d'autres applications.

Plus récemment, deux approches distinctes visant à transposer des outils développés pour le traitement du signal et des images au traitement de données sur graphes, ont connu un intérêt croissant.

La première repose sur la généralisation des ondelettes aux graphes. On peut notamment citer les travaux de [CM06], ceux de [JO05] sur les méthodes multi-échelles ou encore les travaux de [HVG11] sur les transformées spectrales.

La deuxième approche est basée sur la transcription des équations aux dérivées partielles (EDP) sur graphes. Celle-ci consiste à utiliser le calcul discret pour reformuler des problèmes d'EDP sur des graphes. Différentes méthodes et formalismes ont été développés afin d'effectuer ces transcriptions. Parmi les plus populaires, on peut citer les travaux de [Hir03], ceux de [GP10] ainsi que ceux de [vGGOB14], où encore le cadre des équations aux différences partielles (EdP).

Le cadre des EdPs est un ensemble d'opérateurs discrets mimant le comportement des EDPs sur le domaine des graphes, en remplaçant les opérateurs différentiels par des opérateurs de différences, définis dans le cadre de fonctions sur graphes. Historiquement, cette méthode de discrétisation a été introduite par [CFL28] où les auteurs y introduisent la méthode des différences finies comme un moyen simple pour manipuler numériquement les EDPs. Plus récemment, ces EdPs ont suscité un grand intérêt et sont même devenues un sujet d'étude en soi [BM05, Neu06, PC11, LC12]. L'utilisation de ces EdP sur graphes de topologie arbitraire pour le traitement d'images et de données suscite aussi un intérêt croissant. On peut notamment citer les travaux de [ELB08] ou encore ceux de [GP10], ainsi que les références connexes. Parmi celles-ci, on peut citer l'introduction de la variation totale pour des filtres

---

digitaux par [COS01], ou encore [ZS04] qui ont utilisé la variation totale sur graphe pour la classification de données semi-supervisée.

Sur le même principe, les auteurs de [BEM07, LEB07, ELB08] ont introduit une transcription discrète du calcul non-local sur graphe. Ils ont proposé une définition discrète sur graphe des opérateurs de divergence et de gradient, menant à une généralisation du  $p$ -Laplacien sur graphes. Cette formulation permet de transcrire de nombreuses EDPs et méthodes variationnelles sur graphes, à partir d'EdPs dont les coefficients dépendent des données. En particulier, elle permet de revisiter différents problèmes variationnels, comme la régularisation discrète sur des graphes pondérés, conduisant à une nouvelle famille de processus de diffusions, basée sur la généralisation du  $p$ -Laplacien sur graphe. Ces processus sont paramétrés par la structure du graphe et le degré  $p$  de lissage. On peut donc retrouver des processus locaux comme non-locaux, tel que ceux définis par [GO08], en utilisant simplement une construction de graphe locale ou non-locale. Plus récemment, [TEL08, TEL11] ont proposé une définition des gradients directionnels sur graphes. Ces opérateurs permettent d'obtenir des EdPs décrivant des processus morphologiques sur graphes et peuvent être interprétés comme la transcription sur graphe des EDPs morphologiques continues [BM92, VDBS94, AGLM93]. En utilisant la définition des gradients directionnels, les auteurs de [DEL13] ont proposé l'adaptation de l'équation eikonale sur graphes ainsi qu'un algorithme de résolution efficace de celle-ci. Cette adaptation permet le calcul de distance généralisé sur des graphes de topologies arbitraires et peut être appliquée à la segmentation d'image ou encore à la classification semi-supervisée de données. Toujours en utilisant ce cadre, les auteurs de [EDLL14] ont proposé une définition étendant le Laplacien infini sur graphe, ainsi que la courbure moyenne sur graphe et leur application en traitement d'images et de données sur graphe [CED14].

## Contributions principales

Dans ce manuscrit, nous suivons la ligne de recherche basée sur les EdPs et initiée par [BEM07, LEB07, ELB08, TEL11, DEL13]. Ainsi les contributions de ce manuscrit exploitent le cadre des EdPs sur graphes pour transcrire des outils mathématiques continus et étudier leur application au traitement de données sur graphe. Les principales contributions de nos travaux sont les suivantes :

## Transcription du $p$ -Laplacien normalisé sur graphe

- Récemment introduit en lien avec des jeux stochastiques, nous proposons une adaptation et une généralisation du  $p$ -Laplacien normalisé sur graphe, en utilisant le cadre des EdPs. Cette adaptation peut être considérée comme une nouvelle classe de  $p$ -Laplacien sur graphe, sous forme d'une non-divergence, interpolant, pour  $1 \leq p \leq 2$  entre le 1-Laplacien et le Laplacien normalisé et pour  $2 \leq p \leq \infty$  entre le Laplacien et le Laplacien infini.
- Beaucoup de problèmes en traitement du signal pouvant être considérés comme des problèmes d'interpolation, nous étudions le problème de Dirichlet associé à cet opérateur et nous montrons l'existence et l'unicité de la solution à cette équation.
- Nous étudions aussi l'équation de diffusion associée à cet opérateur et nous montrons que le processus de résolution numérique de cette équation permet de retrouver divers filtres utilisés en traitement d'image.

## $p$ -Laplacien et Laplacien infini avec termes de gradients sur graphe

- Nous introduisons ensuite une nouvelle classe de  $p$ -Laplacien et de Laplacien infini sur graphes définis comme une combinaison de termes de gradients à coefficients variables. Nous montrons que cet opérateur est une généralisation des opérateurs défini sur graphe dans le cadre des EdPs. Les cas particuliers de ce nouvel opérateur permettent de retrouver les  $p$ -Laplacien et Laplacien infini sur graphe, mais aussi les opérateurs de gradients morphologiques proposés par [TEL11].
- Comme pour le  $p$ -Laplacien normalisé, nous étudions le problème de Dirichlet associé à cet opérateur et montrons l'existence et l'unicité de la solution. Nous étudions aussi l'équation de diffusion associée à cet opérateur et montrons que sa résolution permet de retrouver des processus de régularisation de données ainsi que des processus morphologiques sur graphes.

## Équation de Poisson sur graphe

- Nous proposons une généralisation de l'équation de Poisson infini sur graphe, utilisant le Laplacien infini avec termes de gradient.

- 
- Nous montrons que l'utilisation de cet opérateur mène à une équation hybride entre une équation de Poisson-infini et de Hamilton-Jacobi et qu'un des cas particuliers de celle-ci permet de retrouver l'équation eikonale sur graphe.
  - Nous utilisons cette équation pour le calcul de distance généralisé sur graphe et nous montrons que celle-ci peut être utilisée dans le cadre de la segmentation semi-supervisée de données sur graphe.

## **Ensemble de niveaux pour la classification de données**

- En nous basant sur la formulation des équations d'évolution de front par ensemble de niveaux sur graphe, nous proposons une formulation permettant de faire évoluer plusieurs fronts simultanément en fonction de la courbure sur graphe.

## **Évaluation quantitative dans le cadre de la classification semi-supervisée et du clustering**

- Nous proposons ensuite d'évaluer quantitativement, dans le cadre de la classification semi-supervisée et du clustering, les méthodes proposées dans cette thèse ainsi que celles précédemment proposées par [ELB08, DEL13].

## **Plan du manuscrit**

Ce manuscrit est divisé en trois parties distinctes. Dans la première partie, nous rappelons les notations et définitions essentielles à la compréhension de ce manuscrit. Dans la deuxième partie, nous présentons les travaux théoriques réalisés pendant cette thèse et enfin, dans la troisième partie nous proposons de les appliquer et de les évaluer quantitativement dans le cadre de la classification et du clustering de données.

## **Graphes et opérateurs**

Dans cette première partie, nous rappelons certaines définitions sur les graphes et les notations qui seront utilisées dans la suite de ce manuscrit. Ces notions sont à la base des opérateurs de différence sur graphe dont nous traitons aux chapitres suivants et sont essentielles à la compréhension des travaux

présentés dans cette thèse. Ensuite, nous rappelons les notions de calcul discret sur graphe, précédemment introduites par [ELB08, BEM09, TEL11]. Nous rappelons tout d'abord les notions d'opérateurs différentiels sur graphe, en introduisant les notions de fonction, de différence, de divergence, de gradient, de  $p$ -Laplacien, définis sur des graphes pondérés. Nous présentons ensuite les opérateurs différentiels directionnels, proposés par [TEL11]. Enfin, nous présentons l'utilisation de ces opérateurs au travers de processus de traitement non-locaux sur graphe.

## Une nouvelle famille de $p$ -Laplacien sur graphe

Dans cette seconde partie, nous proposons tout d'abord une adaptation du  $p$ -Laplacien normalisé sur graphe pondéré de topologie arbitraire en utilisant le cadre des EdPs. Cette adaptation nous permet d'introduire une nouvelle classe de  $p$ -Laplacien sur graphe sous la forme d'une non-divergence. Cet opérateur interpole entre le 1-Laplacien, le Laplacien infini et le Laplacien sur graphe et est obtenu à partir d'opérateurs statistiques transposés dans le cadre des graphes. Comme beaucoup de problèmes en traitement d'images sont formulés comme des problèmes d'interpolations, nous étudions le problème de Dirichlet associé à cet opérateur et nous montrons l'existence et l'unicité de la solution à cette équation. Nous étudions aussi l'équation de diffusion associée à cet opérateur et montrons que le processus de résolution numérique de cette équation permet de retrouver divers filtres utilisés en traitement d'image et de les étendre au domaine des graphes.

Nous proposons ensuite un opérateur  $p$ -Laplacien et Laplacien infini défini comme une combinaison de termes de gradients à coefficients variables. Nous montrons que cet opérateur est une généralisation des opérateurs défini sur graphe dans le cadre des EdPs. En effet les cas particuliers de cet opérateur permettent de retrouver les  $p$ -Laplacien anisotropes et le Laplacien infini, mais aussi les opérateurs de gradients morphologiques introduits par [ELB08, TEL11]. Nous montrons aussi que cet opérateur permet de retrouver différents schémas de discrétisation existants, dans le cas d'opérateurs locaux et non-locaux. Cet opérateur unifie les formulations discrètes du  $p$ -Laplacien, du Laplacien infini, du  $p$ -Laplacien normalisé et du  $p$ -Laplacien non-local sur graphe, que ce soit sur des domaines réguliers ou irréguliers. Comme pour le  $p$ -Laplacien normalisé, nous étudions le problème de Dirichlet ainsi que l'équation de diffusion associée à cet opérateur. Nous montrons l'existence et l'unicité du problème de la solution au problème de Dirichlet et proposons et étudions un schéma numérique de résolution à l'équation de diffusion. Nous appliquons ensuite ces deux processus au traitement d'image et de donnée, en illustrant

---

leurs applications dans les tâches de filtrage, de segmentation et d’inpainting sur différents types de données, tels que les images, les nuages de points 3D ou encore les bases de données.

Enfin, dans le dernier chapitre de cette partie, nous proposons une adaptation de l’équation de Poisson infini sur graphe pondéré, en utilisant la formulation du Laplacien infini avec termes de gradient sur graphe pondéré introduite précédemment dans cette partie. Nous faisons aussi le lien entre cette équation et la version biaisée du Tug-of-War. En utilisant cette formulation, nous proposons une équation hybride de Poisson infini et de Hamilton-Jacobi. Nous montrons aussi le lien entre cette version de l’équation de Poisson infini et l’adaptation de l’équation eikonale sur graphe pondéré [DEL13]. Notre but ici est d’utiliser cette extension de l’équation de Poisson afin de calculer des distances sur tout type de données discrètes pouvant être représentées par un graphe pondéré. Nous montrons, à travers différentes expérimentations, que cette formulation peut être utilisée pour la résolution de différentes applications en traitement d’image, de nuages de point 3D et de données de grandes dimensions, en utilisant une unique formulation.

## Équations d’ensembles de niveaux pour le clustering et la classification de données

Dans cette dernière partie, nous proposons d’utiliser et d’évaluer les algorithmes proposés dans la partie précédente, ainsi que ceux précédemment introduits par [ELB08] pour les tâches de classification transductive (semi-supervisée) et de clustering (non-supervisée). Dans ce cadre, nous proposons aussi d’évaluer l’algorithme de propagation de front, basé sur l’équation eikonale et transposé sur graphe par [DEL13]. Pour ce faire, nous donnons une description rapide de celui-ci au début de cette partie. Nous proposons ensuite une formulation, basée sur la transposition des ensembles de niveaux sur graphes, permettant de faire évoluer plusieurs courbes à la fois sur un graphe. À partir de cette formulation, nous proposons d’utiliser le schéma de résolution proposé par [TEL11], et proposons un algorithme combinant la propagation de front sur graphe et la formulation proposée, dans le cadre de la classification semi-supervisée, afin d’accélérer les temps de calcul et d’améliorer les résultats en termes de taux de classification. Nous proposons aussi une implantation GPU de cet algorithme, accélérant les calculs d’un ordre de grandeur.

Nous comparons ensuite ces différents algorithmes ( $p$ -Laplacien normalisé sur graphe,  $p$ -Laplacien sur graphe, propagation de front sur graphe, combinaison propagation de front / ensemble de niveaux) ainsi que l’algorithme MTV proposé par [BTUVB13] sur différentes bases de données de la littérature. Nous



appliquons aussi ces algorithmes à la classification de données dans le cadre de la microscopie virtuelle.

Nous proposons ensuite un algorithme de clustering, inspiré par les récents travaux de [BHL<sup>+</sup>14]. Celui-ci permet d'utiliser des algorithmes de classification transductive dans un cadre non-supervisé. Nous proposons de reprendre son fonctionnement afin d'utiliser les algorithmes de classification semi-supervisée que nous proposons, permettant ainsi d'effectuer du clustering avec ceux-ci. Nous comparons les algorithmes que nous proposons avec l'algorithme INGRES proposé par [BHL<sup>+</sup>14] ainsi qu'à la méthode NMFR proposée par [YHD<sup>+</sup>12], en termes de pureté de cluster, sur vingt bases de données.

**Première partie**  
**Graphes et opérateurs**



# Chapitre 1

## Graphes

### Sommaire

---

<b>1.1 Définitions et notations utilisées</b> . . . . .	<b>13</b>
<b>1.2 Domaines et constructions de graphe</b> . . . . .	<b>16</b>
1.2.1 Domaines non organisés . . . . .	18
1.2.2 Domaines organisés . . . . .	21
<b>1.3 Conclusion</b> . . . . .	<b>26</b>

---

Dans ce chapitre nous rappelons certaines définitions sur les graphes et les notations qui seront utilisées dans la suite de ce manuscrit. Ces notions sont à la base des opérateurs de différence sur graphe dont nous traitons au chapitre suivant et sont essentielles à la compréhension des travaux présentés dans cette thèse.

### 1.1 Définitions et notations utilisées

Un graphe pondéré, noté  $G = (V, E, w)$ , est composé d'un ensemble de sommets  $V$  (figure 1.1(a)), d'un ensemble d'arêtes  $E$  (figure 1.1(b)) et d'une fonction de poids  $w$  (figure 1.1(c)). Dans ce manuscrit, nous utilisons seulement des graphes orientés.

**Définition 1.** *Sommets*

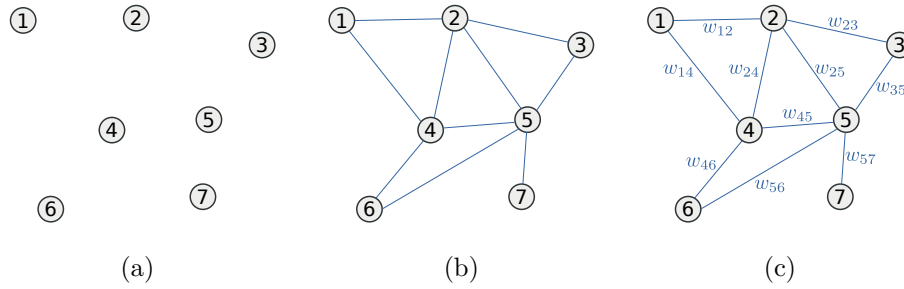


FIGURE 1.1 – Illustration des différents éléments composant un graphe pondéré. (a) Ensemble de sommets  $V$  ; (b) Ensemble d'arêtes  $E$  ajouté à l'ensemble de sommets ; (c) Fonction de poids  $w$  sur les arêtes du graphe.

L'ensemble  $V$  est composé de sommets  $u \in V$ . Chaque sommet est une représentation abstraite d'un élément de la structure de données représentée par le graphe. Soit  $\Omega \subset \mathbb{R}^m$  le domaine discret d'une structure de données. On a alors  $V \subseteq \Omega$ .

**Définition 2.** Arêtes

L'ensemble d'arêtes  $E$  est composé de paires de sommets  $(u, v)$ . Une arête représente la connexion entre 2 sommets  $u$  et  $v$ . On dit alors que les sommets  $u$  et  $v$  sont adjacents, ou voisins, que l'on notera  $u \sim v$ . L'arête est dite incidente aux sommets  $u$  et  $v$ . L'ensemble  $E$  est un sous ensemble de  $V \times V$ . Dans ce manuscrit, nous considérons des graphes sans boucle, ni arête multiple et dont les arêtes sont symétriques. On peut donc définir l'ensemble  $E$  tel que :

$$E = \{(u, v) \in V \times V | u \sim v \text{ et } u \neq v\}. \quad (1.1)$$

En considérant la symétrie des arêtes, on peut aussi noter que si  $(u, v) \in E$ , alors  $(v, u) \in E$ .

**Définition 3.** Voisinage d'un sommet

On notera  $N(u)$  le voisinage d'un sommet  $u$ , c'est-à-dire l'ensemble des sommets adjacents à un sommet, tel que :

$$N(u) = \{v \in V | (u, v) \in E\}. \quad (1.2)$$

**Définition 4.** Fonction de poids

La fonction de poids  $w$  représente la similarité entre deux sommets du graphe. Elle est définie telle que  $w : V \times V \rightarrow [0, 1]$ . Cette fonction est symétrique :  $\forall (u, v) \in V \times V, w(u, v) = w(v, u)$ . Afin d'alléger les écritures, nous utiliserons dans la suite de ce manuscrit la notation condensée  $w_{uv} = w(u, v)$ . Par convention, lorsque le couple  $(u, v) \notin E$ , on affectera la valeur 0 à  $w_{uv}$ . Nous pouvons résumer les propriétés de la fonction de poids :

$$w_{uv} \begin{cases} \in [0, 1] & \forall (u, v) \in E \\ = 0 & \forall (u, v) \notin E \\ = w_{vu} & (\text{symétrie}). \end{cases} \quad (1.3)$$

**Définition 5.** Degré d'un sommet

Le degré, ou valence, d'un sommet  $\delta : V \rightarrow \mathbb{N}$ , représente le nombre de sommets adjacents au sommet considéré. Il est défini tel que :

$$\delta(u) = |N(u)|, \quad (1.4)$$

où  $|\cdot|$  désigne le cardinal d'un ensemble.

Le degré pondéré d'un sommet  $\delta_w : V \rightarrow \mathbb{R}^+$  est défini tel que :

$$\delta_w(u) = \sum_{v \in N(u)} w(u, v). \quad (1.5)$$

Il caractérise localement la quantité totale d'interaction d'un sommet avec ses sommets adjacents. Dans le cas d'un graphe non pondéré ( $w_{uv} = 1$  si  $(u, v) \in E$ ,  $w_{uv} = 0$  sinon) on a  $\delta_w(u) = \delta(u)$ .

Le degré normalisé  $\delta^N(u)$  est défini tel que  $\delta^N : V \rightarrow \mathbb{R}^+$  :

$$\delta^N(u) = \frac{\delta_w(u)}{\delta(u)}. \quad (1.6)$$

Il représente la quantité moyenne d'interaction d'un sommet avec ses sommets adjacents.

Soit  $G = (V, E, w)$  un graphe pondéré et  $\mathcal{A} \subset V$  un sous ensemble de  $V$ . Il existe deux notions de frontières d'une partie d'un graphe : les frontières de sommets et la frontière des arêtes.

**Définition 6.** Frontières de sommets

Nous définissons ici deux frontières de sommets : la frontière des sommets interne et externe.

- La frontière des sommets interne, noté  $\partial^- \mathcal{A}$  est définie comme l'ensemble des sommets  $u \in \mathcal{A}$  ayant au moins un voisin dans le complémentaire de  $\mathcal{A}$  (noté  $\mathcal{A}^c$ ). Cet ensemble est défini tel que :

$$\partial^- \mathcal{A} = \{u \in \mathcal{A} | \exists v \in \mathcal{A}^c, v \sim u\}. \quad (1.7)$$

- La frontière des sommets externe, noté  $\partial^+ \mathcal{A}$  est définie comme l'ensemble des sommets  $u \in \mathcal{A}^c$  ayant au moins un voisin dans  $\mathcal{A}$  :

$$\partial^+ \mathcal{A} = \{u \in \mathcal{A}^c | \exists v \in \mathcal{A}, v \sim u\} = \partial^- \mathcal{A}^c. \quad (1.8)$$

**Définition 7.** Frontière des arêtes

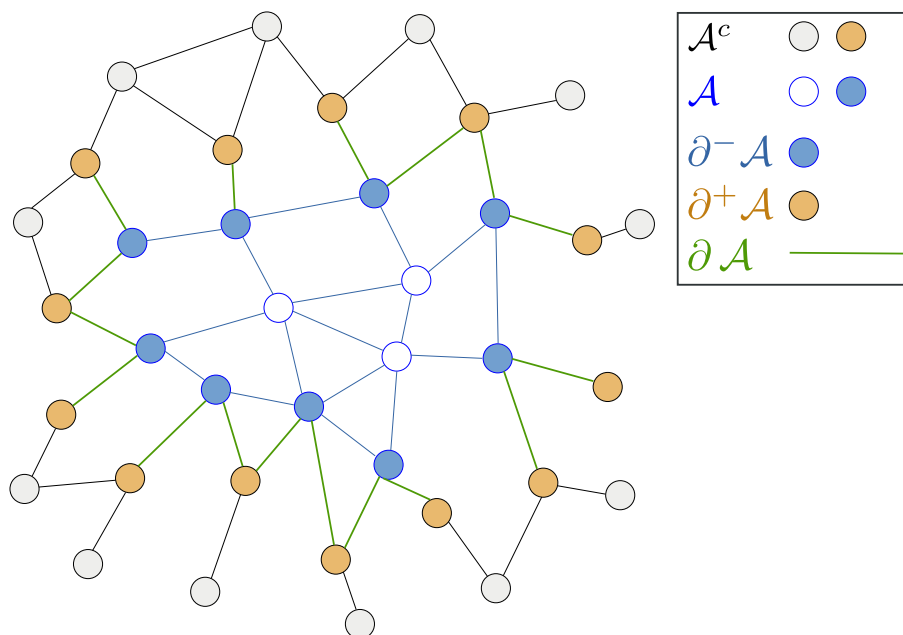


FIGURE 1.2 – Illustration des différentes frontières d’un ensemble de sommet.

La frontière des arêtes de  $\mathcal{A}$ , notée  $\partial\mathcal{A}$ , est l’ensemble des arêtes de  $G$  incidentes à un sommet de  $\mathcal{A}$  et de  $\mathcal{A}^c$  :

$$\partial\mathcal{A} = \{(u, v) \in E \mid u \in \mathcal{A}, v \in \mathcal{A}^c\} = \{\partial^-\mathcal{A} \times \partial^+\mathcal{A}\} \cap E. \quad (1.9)$$

Ces définitions de frontières sont illustrées à la figure 1.2.

## 1.2 Domaines et constructions de graphe

Dans cette section nous présentons les différentes constructions de graphe et fonctions de poids que nous utiliserons dans ce manuscrit. Le choix du type de construction du graphe est primordiale pour obtenir le traitement souhaité des données représentées par le graphe. En effet, la construction et la pondération d’un graphe dépendent de l’application considérée ainsi que du type de domaine sur lequel vivent les données à traiter.

Nous considérons dans cette section un domaine  $\Omega \subset \mathbb{R}^m$ , une fonction d’attributs  $\mathcal{F} : \Omega \rightarrow \mathbb{R}^c$  associant un vecteur d’attribut à chaque élément de  $\Omega$ , ainsi qu’un graphe  $G = (V, E, w)$  avec  $V \subseteq \Omega$ . La fonction d’attribut  $\mathcal{F}$  représente les données à traiter. Les différents domaines considérés peuvent être organisés ou non organisés. Par exemple, la grille discrète régulière des coordonnées des pixels d’une image en deux dimensions peut être considérée

comme un domaine organisé  $\Omega \subset \mathbb{R}^2$  et  $\mathcal{F} : \Omega \rightarrow \mathbb{R}^c$  représente l'intensité ou la valeur des canaux de chaque couleur associée à un pixel de  $\Omega$ . Un domaine non organisé pourrait être un nuage de points à  $m$ -dimensions, sans a priori sur les relations de voisinage des points.

Afin de construire des graphes sur des domaines organisés ou non, nous utilisons une métrique, ou distance, que ce soit pour l'établissement du voisinage d'un sommet ou encore la pondération des arêtes de ces graphes. Une métrique  $d$  sur un ensemble  $\Omega$  est une application de  $\Omega \times \Omega$  dans l'ensemble des nombres réels :

$$d : \Omega \times \Omega \rightarrow \mathbb{R}^+. \quad (1.10)$$

Cette application associe à chaque couple de points  $(u, v)$  une distance, avec  $u, v \in V \subseteq \Omega$ . La distance Euclidienne (ou distance de Minkowski d'ordre 2) est une des métriques les plus utilisées :

$$d_2(u, v) = \sqrt{\sum_{i=1}^c (\mathcal{F}_i(u) - \mathcal{F}_i(v))^2}. \quad (1.11)$$

Nous utiliserons aussi dans ce manuscrit la distance de Manhattan (ou distance de Minkowski d'ordre 1) :

$$d_1(u, v) = \sum_{i=1}^c |\mathcal{F}_i(u) - \mathcal{F}_i(v)|, \quad (1.12)$$

ainsi que la distance de Chebyshev (ou distance de Minkowski d'ordre  $\infty$ ) :

$$d_\infty(u, v) = \max_{i=1}^c |\mathcal{F}_i(u) - \mathcal{F}_i(v)|. \quad (1.13)$$

D'une manière générale, la distance de Minkowski peut s'écrire :

$$d_p(u, v) = \left( \sum_{i=1}^c |\mathcal{F}_i(u) - \mathcal{F}_i(v)|^p \right)^{1/p}. \quad (1.14)$$

Afin de pondérer les graphes que nous construisons, nous utilisons des mesures (ou fonctions) de similarité. Les mesures de similarité sont des fonctions à valeurs réelles quantifiant la similarité, ou affinité, entre deux objets. Dans le cas des graphes pondérés, nous utiliserons des fonctions de similarités  $s$  définies telles que  $s : V \times V \rightarrow [0, 1]$ . Une mesure de similarité peut être vue comme l'inverse d'une fonction de distance, *i.e.* sa valeur étant plus grande lorsque les objets sont proches et inversement, plus les objets sont éloignés plus sa valeur est petite. Dans notre cas, cette mesure de similarité sera proche de



1 lorsque les objets seront assez similaires et proche de 0 lorsque les objets seront éloignés. L'utilisation d'une mesure de similarité particulière dépend de l'application et il n'y a pas de règle générale pour le choix de cette fonction.

Dans ce manuscrit, nous considérerons en général les mesures de similarités suivantes :

$$s_1(u, v) = 1, \tag{1.15}$$

$$s_2(u, v) = \frac{1}{d(u, v)} \in [0, 1], \tag{1.16}$$

$$s_3(u, v) = \exp\left(\frac{-d(u, v)^2}{\sigma^2}\right). \tag{1.17}$$

Par la suite nous définirons d'autres mesures de similarités propres à une application ou une démonstration. Nous pouvons maintenant définir la fonction de poids  $w : V \times V \rightarrow [0, 1]$  à l'aide d'une fonction de similarité telle que :

$$w_{uv} = \begin{cases} s(u, v) & \forall (u, v) \in E, \\ 0 & \forall (u, v) \notin E. \end{cases} \tag{1.18}$$

Nous présentons maintenant les différents types de graphe que nous utiliserons dans la suite de ce manuscrit.

### 1.2.1 Domaines non organisés

Considérons le cas général d'un domaine non organisé. Par domaine non organisé nous entendons domaine ne possédant aucune structure ou relation de voisinage pouvant conduire intrinsèquement à la construction d'un graphe. Comme précédemment, l'ensemble des points  $V \subseteq \Omega$  de ce domaine est représenté par une fonction d'attributs  $\mathcal{F}$ . Construire un graphe à partir de ces données revient à définir une relation de voisinage pour chaque sommet du graphe. Nous montrons dans la suite que cette relation est basée sur une métrique entre les valeurs de la fonction d'attribut de chaque sommet du graphe.

#### Graphe complet

Un graphe complet est un graphe connectant chaque sommet  $u$  de  $V$  à l'ensemble des sommets  $V \setminus \{u\}$ . Cette structure de graphe est la plus simple à mettre en oeuvre, cependant elle est très coûteuse en mémoire ou en temps de calcul, l'ensemble des arêtes étant défini tel que :

$$E = V \times V \setminus (u, u), u \in V. \tag{1.19}$$

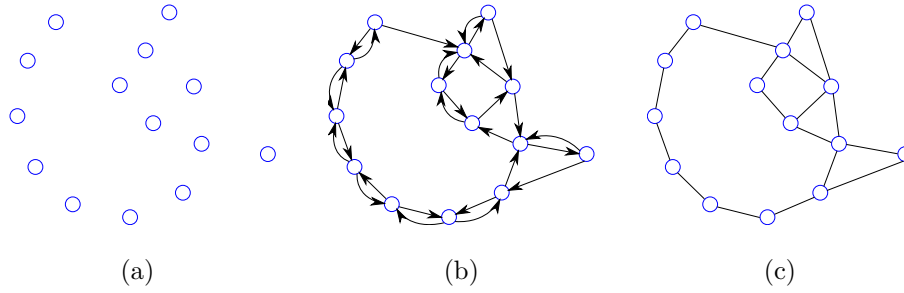


FIGURE 1.3 – Illustration de la construction d'un graphe des  $k$  plus proches voisins, avec  $k = 2$ . (a) Ensemble de sommets ; (b) Graphe orienté des 2-ppv ; (c) Graphe non orienté des 2-ppv.

### Graphe des $k$ plus proches voisins

Soient  $k$  un entier positif et  $d$  une métrique définissant une distance sur notre domaine  $\Omega$ . Le graphe des  $k$  plus proches voisins ( $k$ -ppv) est un graphe où chaque sommet est connecté à ses  $k$  plus proches voisins au sens de la métrique  $d$ . On peut interpréter cette définition de deux façons : le graphe créé est orienté (figure 1.3(b)), reliant chaque sommet à ses  $k$  plus proches voisins par un arc, garantissant ainsi que le cardinal de  $E$  soit égal à  $k|V|$ . Dans ce manuscrit nous utilisons des graphes non orientés. Nous considérons chaque arc du graphe produit comme une arête du graphe :

$$E = \{(u, v) | u \in N(v) \text{ ou } v \in N(u)\}. \quad (1.20)$$

Cette construction de graphe est illustrée à la figure 1.3(c). Elle garantit que chaque sommet a *au moins*  $k$  voisins. Elle implique aussi qu'un ou plusieurs sommets puissent avoir  $|V| - 1$  voisins. On peut remarquer qu'en définissant la valeur de  $k$  telle que  $k = |V| - 1$ , on retrouve la construction d'un graphe complet.

La figure 1.4 illustre la construction d'un graphe des  $k$ -ppv à partir d'une base de données d'images, chaque image pouvant être vue comme un vecteur composé des niveaux d'intensité des pixels de chaque image.

### Graphe de $\epsilon$ -voisinage

Soient  $\epsilon$  un réel strictement positif et  $d$  une métrique définissant une distance sur notre domaine  $\Omega$ . Un graphe de  $\epsilon$ -voisinage est un graphe où le voisinage d'un sommet  $u$  est l'ensemble des sommets dont la distance à  $u$  est inférieure à  $\epsilon$  au sens de la métrique  $d$ . On peut écrire l' $\epsilon$ -voisinage  $N_\epsilon$  d'un sommet  $u$  tel que :

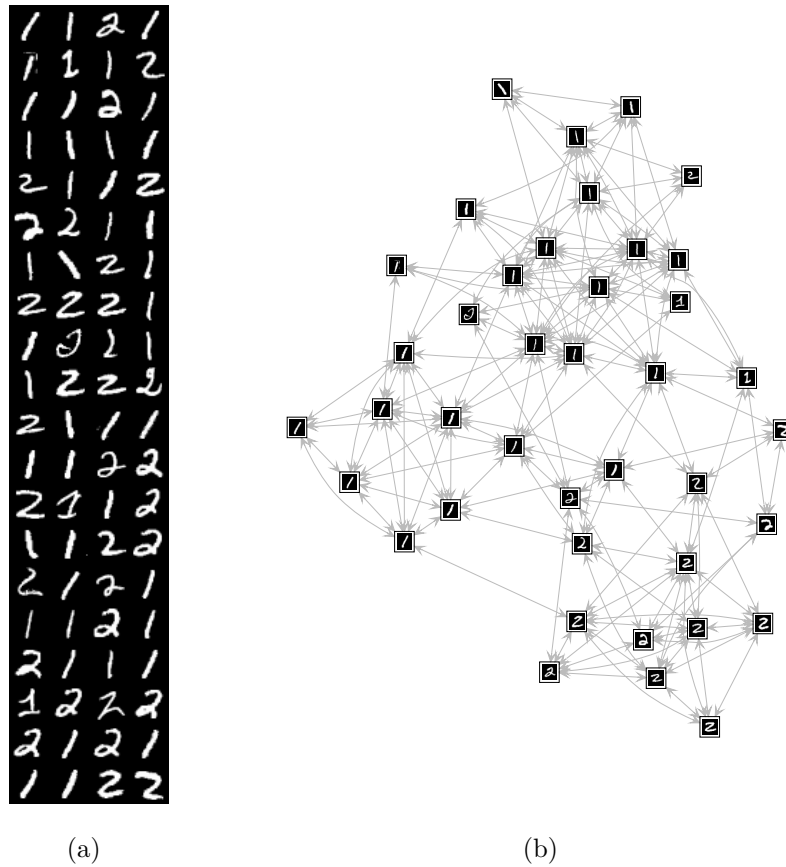


FIGURE 1.4 – Illustration de la construction d'un graphe des  $k$  plus proches voisins sur une base de données. (a) Base de données d'images de chiffres manuscrits; (b) Graphe des  $k$ -ppv construit à partir de (a);

$$N_\epsilon(u) = \{v \in V \mid d(u, v) \leq \epsilon\}. \quad (1.21)$$

La figure 1.5 illustre le voisinage d'un sommet en utilisant différentes métriques. On peut maintenant définir l'ensemble des arêtes  $E$  tel que :

$$E = \bigcup_{u \in V} \{u\} \times N_\epsilon(u). \quad (1.22)$$

On peut remarquer qu'en définissant  $\epsilon$  tel que  $\epsilon = \infty$ , on retrouve le voisinage d'un graphe complet.

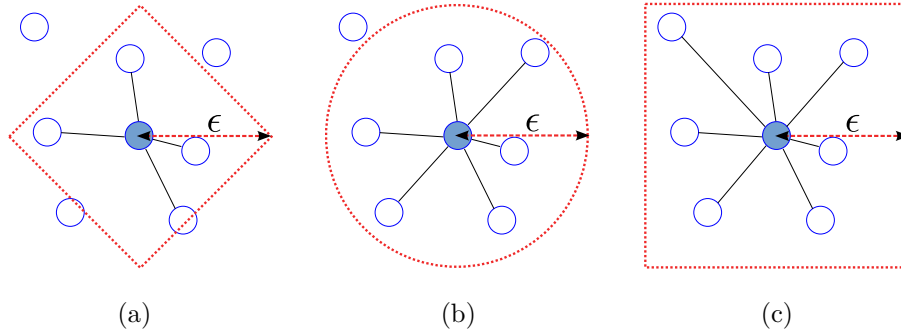


FIGURE 1.5 – Illustration du voisinage d’un sommet lors de la construction d’un graphe de  $\epsilon$ -voisinage en utilisant différentes métriques. (a) Utilisation de la distance de Manhattan; (b) Utilisation de la distance Euclidienne; (c) Utilisation de la distance de Chebyshev.

### 1.2.2 Domaines organisés

Dans cette section nous présentons différentes manières de construire un graphe à partir d’un domaine organisé. Un domaine organisé représente des données pour lesquelles l’organisation spatiale est connue a priori.

#### Signaux discrets

Considérons un signal continu  $g : \mathbb{R}^q \rightarrow \mathbb{R}^c$  que l’on va échantillonner avec un pas spatial constant  $\Delta x_i$  avec  $i = \llbracket 0, q - 1 \rrbracket$ . En considérant le pas spatial comme étant l’unité spatiale ( $\Delta x_i = 1$ ), on peut considérer le signal  $g$  comme étant une fonction  $f : \mathbb{Z}^q \rightarrow \mathbb{R}^c$  défini sur une grille à  $q$  dimensions. Pour  $q = 1$  on retrouve un signal 1D,  $q = 2$  correspond à un signal 2D, *e.g.* une image et  $q = 3$  correspond à une image volumique. L’ensemble des sommets du graphe  $V$  est défini comme un sous ensemble de  $\mathbb{Z}^q$ , ou chaque sommet représente un échantillon du signal.

Le voisinage d’un échantillon du signal sur une grille est généralement défini à partir d’une distance spatiale. On peut par exemple retrouver les voisinages standards utilisés en traitement d’image, par exemple avec les filtres utilisant un produit de convolution, en utilisant un  $\epsilon$  voisinage, avec  $\epsilon = 1$ , ou la fonction d’attribut  $\mathcal{F}$  représente les coordonnées d’un pixel. On peut retrouver un 4-voisinage avec une distance de Manhattan (figure 1.6(a)), ainsi qu’un 8-voisinage avec la distance de Chebyshev (figure 1.6(d)). En faisant varier  $\epsilon$ , afin d’étendre le voisinage d’un sommet, on peut remarquer qu’en utilisant la distance de Manhattan, on obtient un voisinage en forme de losange (figures 1.6(b) et 1.6(c)), correspondant à la boule  $\mathcal{L}_1$ , et en utilisant la distance de

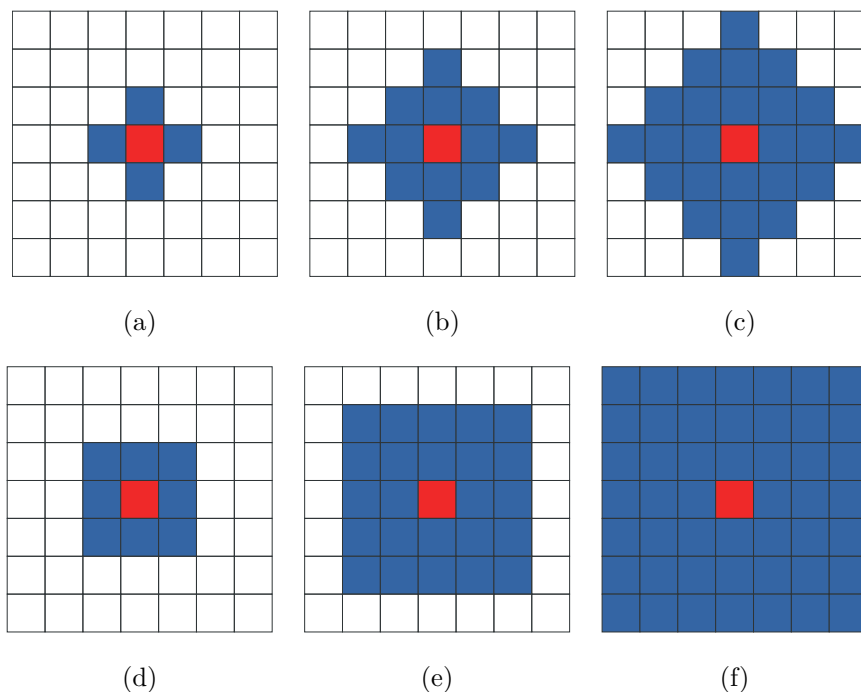


FIGURE 1.6 – Illustration du voisinage d’un sommet lors de la construction d’un graphe de  $\epsilon$  voisinage sur une grille en utilisant la distance de Manhattan ((a), (b), (c)) et la distance de Chebyshev ((d), (e), (f)). (a) et (d)  $\epsilon = 1$ ; (b) et (e)  $\epsilon = 2$ ; (c) et (f)  $\epsilon = 3$ ;

Chebyshev on obtient un voisinage de forme carré (figures 1.6(e) et 1.6(f)), correspondant à la boule  $\mathcal{L}_\infty$ .

Afin de pondérer les arêtes du graphe créé, on utilise généralement une fonction de poids représentant la similarité entre l’intensité ou le vecteur couleur de chaque pixel. La fonction d’attributs utilisée ici sera donc le signal  $f$ . On utilise généralement la mesure de similarité  $s_3$  (1.17), avec la mesure de distance  $d_2$  (1.11).

### Images texturées, représentation non-locale

Dans le cadre du traitement d’image, notamment en débruitage, on peut considérer les approches par patches, dites non-locales. Ces méthodes permettent de caractériser un pixel par un vecteur d’attributs plus étendu que la seule valeur d’intensité ou de couleur du pixel considéré. Un patch représentant un pixel  $u$  est une sous image carrée de l’image originale, centrée au pixel  $u$  de largeur  $W$ .

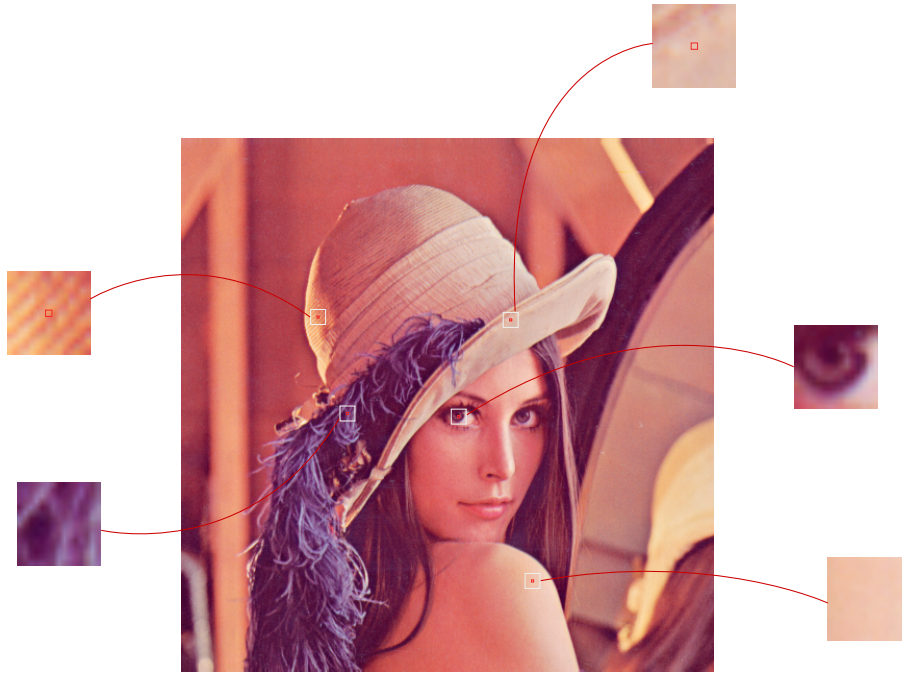


FIGURE 1.7 – Illustration des patches pris à différents endroits d’une image.

En utilisant un  $\epsilon$  voisinage et la distance de Chebyshev, on peut représenter un patch par la fonction d’attribut  $\mathcal{F}_W : V \rightarrow \mathbb{R}^{c \times W \times W}$  définie par :

$$\mathcal{F}_W(u) = (f_i(v), i \in \llbracket 1, c \rrbracket, v \in N_\epsilon(u) \cup u), \quad (1.23)$$

où  $c$  est la dimension du vecteur couleur  $f$  défini en chaque pixel de l’image. On peut calculer  $\epsilon$  en fonction de la largeur  $W$  avec la relation  $\epsilon = (W - 1)/2$ . La figure 1.7 montre des patches pris à différents endroits dans une image.

Les méthodes basées sur les patches ont d’abord été utilisées dans le cadre de la synthèse de texture [EL<sup>+</sup>99]. Elles ont été utilisées ensuite pour le filtrage et la restauration d’image et de maillage [GO07, BCM08, ELB08, BEM09]. Elles ont aussi été étendues aux nuages de points 3D dans le cadre de la restauration et de l’inpainting [LEL13]. Ces méthodes, comparées aux méthodes locales, sont plus efficaces, notamment pour restaurer des structures fines et répétitives de l’image.

Contrairement aux filtres habituellement utilisés en traitement d’image, les méthodes de débruitage par patches, introduites par [BCM08], réalisent une moyenne pondérée de la totalité des valeurs des pixels contenus dans l’image. Pour cette raison, elles sont dites "non-locales". La pondération utilisée est souvent une similarité Gaussienne ( $s_3$ ), utilisant comme métrique  $d$  une

distance entre patchs. En pratique, afin de réduire la complexité algorithmique, on n'utilisera pas toute l'image, mais une fenêtre de voisinage centré au pixel considéré. On parlera alors de semi non-localité.

On peut donc ici construire un graphe représentant la similarité entre patchs de pixels, en utilisant une fenêtre de voisinage pour chaque sommet du graphe (un sommet représentant un pixel de l'image). On utilisera la plupart du temps une fenêtre de voisinage carrée, en utilisant la distance de Chebyshev. La similarité entre les sommets peut être calculée comme expliqué plus haut, en utilisant une similarité Gaussienne. Afin de définir une distance entre patchs, on peut considérer un patch comme un vecteur d'attributs d'un sommet du graphe et ainsi utiliser une distance entre vecteurs. On peut aussi utiliser une distance pondérée par un noyau Gaussien spatial  $\mathcal{G}_\sigma : \mathbb{R}^2 \rightarrow \mathbb{R}$  de variance  $\sigma$ , centré sur le pixel considéré, afin de privilégier plus ou moins les valeurs du centre d'un patch. Une telle distance  $d_p$  est définie telle que :

$$d_p(u, v) = \sum_{i=0}^{W \times W} \sum_{j=0}^c \mathcal{G}_\sigma(x(i), y(i)) d(\mathcal{F}_W(u)(j \times W + i), \mathcal{F}_W(v)(j \times W + i)), \quad (1.24)$$

avec  $x$  et  $y$  les coordonnées des pixels des patchs, avec comme origine spatiale le centre du patch.

## Graphe d'adjacence de région

On peut considérer les partitions, ou cartes de régions, comme des domaines organisés : chaque région peut être considérée comme un objet du domaine et le voisinage d'une région peut être considéré comme les régions partageant la frontière de cette région. La génération d'une partition peut s'effectuer en utilisant différentes méthodes. Parmi ces méthodes, on peut citer la ligne de partage des eaux [VS91, CBNC09], les partitions d'énergie [AC04] ou encore les méthodes de génération de superpixels [BTEL14].

Ainsi, pour construire un graphe d'adjacence de région (aussi appelé RAG, de l'anglais region adjacency graph), on associe chaque région à un sommet. On va considérer que deux sommets sont voisins si les régions associées à ces sommets sont adjacentes, c'est-à-dire si ces deux régions partagent la même frontière.

La figure 1.8 illustre la construction d'un tel graphe.

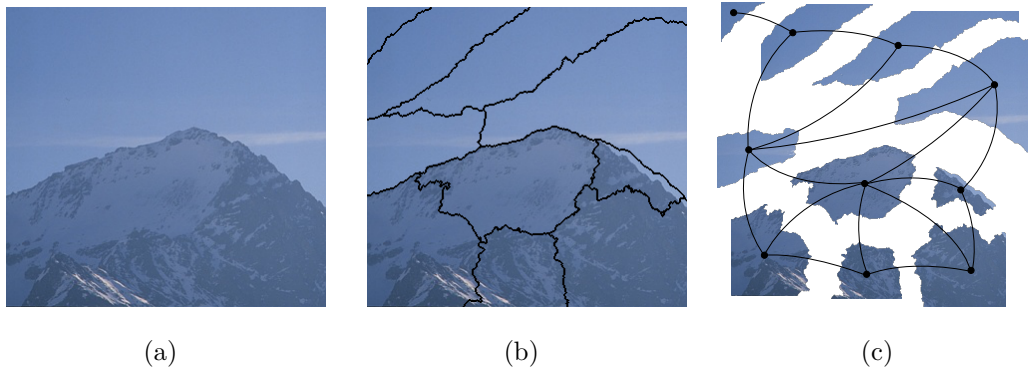


FIGURE 1.8 – Illustration de la construction d'un graphe d'adjacence de région. (a) image originale. (b) Partition de l'image. (c) Graphe d'adjacence de région correspondant à la partition de l'image.

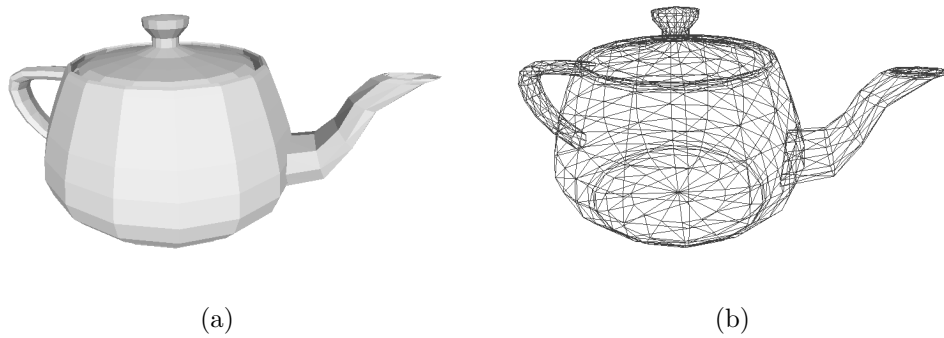


FIGURE 1.9 – Construction d'un graphe à partir d'un maillage 3D. (a) Un maillage 3D. (b) Le graphe induit par ce maillage.

### Maillage 3D

Les maillages 3D sont utilisés pour la représentation discrète de surface. Ils sont constitués de points dans  $\mathbb{R}^3$  et de cellules, ou faces (le plus souvent des triangles) reliant  $n$  de ces points, afin d'approximer une surface.

On peut donc construire directement un graphe sur un maillage en représentant les points de ce maillage par l'ensemble de sommet  $V$  de notre graphe, l'ensemble  $E$  étant constitué des arêtes des cellules reliant les points du maillage.

On peut aussi considérer la construction du graphe dual : chaque sommet du graphe représente une face du maillage. Le voisinage d'un sommet du graphe étant défini comme l'ensemble des faces adjacentes (partageant une arête) à la



face représentée par le sommet. Ce type de construction de graphe est similaire à la construction des RAGs.

### **1.3 Conclusion**

Dans ce premier chapitre, nous avons introduit les définitions et notations clés pour la suite du manuscrit. Nous avons aussi rappelé différentes manières de construire des graphes sur des domaines organisés ou non. Dans le prochain chapitre, nous rappelons des notions de calcul discret sur graphes, essentielles elles aussi à la compréhension des travaux qui seront présentés dans les chapitres suivants.

# Chapitre 2

## Calcul discret sur graphe

### Sommaire

---

<b>2.1 Opérateurs sur graphe</b>	<b>28</b>
2.1.1 Fonctions définies sur un graphe	28
2.1.2 Opérateur de différence pondérée	29
2.1.3 Opérateur de divergence pondérée	29
2.1.4 Opérateur gradient	30
2.1.5 Opérateurs $p$ -Laplace	30
<b>2.2 Opérateurs directionnels</b>	<b>33</b>
2.2.1 Opérateurs de différences directionnelles	34
2.2.2 Opérateurs gradients directionnels	34
2.2.3 Laplacien infini sur graphe	36
<b>2.3 Opérateurs moyenneurs non-locaux</b>	<b>38</b>
2.3.1 Régularisation par $p$ -Laplacien	38
2.3.2 Opérateurs morphologiques non-locaux	40
<b>2.4 Extension des filtres de choc sur graphe</b>	<b>40</b>
2.4.1 Filtres de choc	41
2.4.2 Extension sur graphe	41
<b>2.5 Conclusion</b>	<b>42</b>

---

Dans ce chapitre nous rappelons les notions de calcul discret sur graphe, précédemment introduites par [ELB08, BEM09, TEL11]. Nous rappelons tout d'abord les notions d'opérateurs différentiels sur graphe, en introduisant les notions de fonction, de différence, de divergence, de gradient, de  $p$ -Laplacien,

définis sur des graphes pondérés. Nous présentons ensuite les opérateurs différentiels directionnels, proposés par [TEL11]. Enfin, nous présentons l'utilisation de ces opérateurs au travers de processus de traitement non-locaux sur graphe.

## 2.1 Opérateurs sur graphe

### 2.1.1 Fonctions définies sur un graphe

#### Espace de fonctions

Soit  $\mathcal{H}(V)$  l'espace de Hilbert des fonctions à valeurs dans  $\mathbb{R}$  sur l'ensemble des sommets  $V$  d'un graphe. Chaque fonction  $f : V \rightarrow \mathbb{R}$  de  $\mathcal{H}(V)$  associe une valeur réelle  $f(u)$  à chaque sommet  $u \in V$ . On peut voir  $f$  comme un vecteur de  $\mathbb{R}^n$ , où  $n = |V|$ , chaque composante du vecteur  $f = [f_1, \dots, f_n]$  correspondant à un sommet de  $V$ . Par définition l'espace  $\mathcal{H}(V)$  est muni d'un produit scalaire  $\langle \cdot, \cdot \rangle$ , défini pour deux fonctions  $f, g \in \mathcal{H}(V)$  par :

$$\langle f, g \rangle = \sum_{u \in V} f(u)g(u). \quad (2.1)$$

De la même manière, nous définissons  $\mathcal{H}(E)$  comme l'espace de Hilbert des fonctions à valeurs réelles sur l'ensemble des arêtes  $E$  d'un graphe. Cet espace est muni du produit scalaire pour deux fonctions  $F, G \in \mathcal{H}(E)$ , défini tel que :

$$\langle F, G \rangle = \sum_{(u,v) \in E} F(u,v)G(u,v). \quad (2.2)$$

#### Intégrale et norme d'une fonction discrète

Dans le cadre d'un domaine discret fini  $A$ , l'intégrale d'une fonction peut être vue comme la somme des valeurs de la fonction pour tous les éléments composant le domaine discret :

$$\int_A f = \sum_{u \in A} f(u). \quad (2.3)$$

Sa norme  $\mathcal{L}_p$  est donnée par :

$$\|f\|_p = \left( \sum_{u \in A} |f(u)|^p \right)^{1/p}, \quad (2.4)$$

où  $1 \leq p < \infty$ . Dans le cas où  $p = \infty$ , la norme  $\mathcal{L}_\infty$  est donnée par :

$$\|f\|_\infty = \max_{u \in A} (|f(u)|). \quad (2.5)$$

Dans notre manuscrit,  $A$  peut soit désigner soit  $V$ , soit  $E$ .

### 2.1.2 Opérateur de différence pondérée

L'opérateur de différence pondéré d'une fonction  $f \in \mathcal{H}(V)$ , noté  $d_w : \mathcal{H}(V) \rightarrow \mathcal{H}(E)$ , est défini sur une paire de sommets  $(u, v) \in E$  tel que :

$$d_w(f)(u, v) = \sqrt{w_{uv}}(f(v) - f(u)), \quad (2.6)$$

avec  $w_{uv}$  une fonction de poids, telle que définie aux équations (1.3) et (1.18). On peut noter ici qu'en remplaçant  $w_{uv}$  dans l'équation (2.6) par  $1/h_{uv}^2$ , où  $h_{uv}$  est le pas de discrétisation entre les points  $u$  et  $v$ , on obtient :

$$d_w(f)(u, v) = \frac{(f(v) - f(u))}{h_{uv}}, \quad (2.7)$$

ce qui correspond à l'approximation décentrée du premier ordre de  $f'$  par la méthode des différences finies. On peut interpréter ceci comme l'opérateur de dérivée discrète appliqué le long d'une arête  $(u, v) \in E$  :

$$\left. \frac{\partial f}{\partial(u, v)} \right|_u = \partial_v f(u) = d_w(f)(u, v). \quad (2.8)$$

### 2.1.3 Opérateur de divergence pondérée

Afin de définir l'opérateur de divergence d'une fonction sur graphe, nous définissons d'abord l'opérateur adjoint de l'opérateur de différence pondéré. On peut retrouver l'opérateur adjoint  $d_w^* : \mathcal{H}(E) \rightarrow \mathcal{H}(V)$  par la relation suivante :

$$\langle d_w(f), F \rangle = \langle f, d_w^*(F) \rangle, \quad (2.9)$$

où  $f \in \mathcal{H}(V)$  et  $F \in \mathcal{H}(E)$ . En utilisant les définitions des différences (2.6) et des produits scalaires sur  $\mathcal{H}(V)$  (2.1) et  $\mathcal{H}(E)$  (2.2), l'expression de l'opérateur  $d_w^*$  en un sommet  $u$  peut s'écrire :

$$d_w^*(F)(u) = \sum_{v \in V} \sqrt{w_{uv}} (F(v, u) - F(u, v)). \quad (2.10)$$

L'opérateur de divergence, défini par :

$$\operatorname{div}_w = -d_w^*, \quad (2.11)$$

mesure le flot d'une fonction de  $\mathcal{H}(E)$  en chaque sommet du graphe. À partir des définitions précédentes, il est aisé de montrer que  $\sum_{u \in V} \sum_{v \in V} d_w(f)(u, v) = 0$ ,  $f \in \mathcal{H}(V)$  et que  $\sum_{u \in V} \operatorname{div}_w(F)(u) = 0$ ,  $F \in \mathcal{H}(E)$ .

On remarque aussi que si  $F$  est asymétrique, *i.e.*  $F(u, v) = -F(v, u)$ , l'opérateur  $\operatorname{div}_w$  peut s'écrire :

$$\operatorname{div}_w(f)(u) = 2 \sum_{v \in V} \sqrt{w_{uv}} (F(u, v)). \quad (2.12)$$

### 2.1.4 Opérateur gradient

En se basant sur la définition de l'opérateur de différence pondérée (2.6), nous introduisons l'opérateur de gradient pondéré d'une fonction  $f \in \mathcal{H}(V)$ , noté  $\nabla_w : \mathcal{H}(V) \rightarrow \mathcal{H}(V)^V$ , défini comme le vecteur des différences pondérées en un sommet  $u \in V$  :

$$\nabla_w(f)(u) = (d_w(f)(u, v))_{v \in V}. \quad (2.13)$$

Le gradient pondéré en un sommet  $u \in V$  peut être interprété comme une fonction appartenant à  $\mathcal{H}(V)$ . De ce fait nous pouvons donner sa norme  $\mathcal{L}_p$  et  $\mathcal{L}_\infty$ , représentant la variation locale de la fonction  $f$ , en se servant de la définition (2.4) :

$$\|\nabla_w(f)(u)\|_p = \left( \sum_{v \sim u} \sqrt{(w_{uv})^p} |f(v) - f(u)|^p \right)^{\frac{1}{p}}, \quad (2.14)$$

pour  $1 \leq p < \infty$  et de la définition (2.5) :

$$\|\nabla_w(f)(u)\|_\infty = \max_{v \sim u} (\sqrt{w_{uv}} |f(v) - f(u)|). \quad (2.15)$$

### 2.1.5 Opérateurs $p$ -Laplace

Le  $p$ -Laplacien sur graphe, qui peut être considéré comme une généralisation du  $p$ -Laplacien discret, a commencé à attirer l'attention dans des champs comme les mathématiques, l'apprentissage (machine learning), ainsi qu'en traitement d'images ou de variétés. Par exemple, pour  $p \neq 2$ , le  $p$ -Laplacien sur graphe a été étudié en relation avec les coupes de  $p$ -cheeger et la classification de données [HB10] ainsi que la classification semi-supervisée [ZS06]. Les EdPs sur graphe basées sur le  $p$ -Laplacien discret ont été étudiées comme un sujet

d'intérêt propre, par l'étude de l'existence et du comportement qualitatif de leurs solutions respectives [LC12, Neu06, PC11]. Dans cette section, nous rappelons différentes définitions de cet opérateur sur graphe, introduites par [ELB08, BEM09].

### $p$ -Laplacien isotrope

Soit  $p$  un réel positif. Le  $p$ -Laplacien isotrope sur graphe, noté  $\Delta_{w,p}^i : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$ , est défini comme l'analogie discret du  $p$ -Laplacien continu [Kaw11], à partir des opérateurs de différences (2.6) et de divergence (2.11), tel que :

$$\Delta_{w,p}^i(f)(u) = \frac{1}{2} \operatorname{div}_w \left( \|\nabla_w(f)\|_2^{p-2} d_w(f) \right)(u), \quad \text{où } 1 \leq p < \infty, \quad (2.16)$$

Par laquelle nous entendons :

$$\Delta_{w,p}^i(f)(u) = \frac{1}{2} \sum_{v \sim u} \sqrt{w_{uv}} \left( \|\nabla_w(f)(u)\|_2^{p-2} d_w(f)(u,v) - \|\nabla_w(f)(v)\|_2^{p-2} d_w(f)(v,u) \right), \quad \text{où } 1 \leq p < \infty. \quad (2.17)$$

En utilisant les définitions (2.6) et (2.11), on peut réécrire le  $p$ -Laplacien isotrope en un sommet  $u \in V$  comme suit :

$$\Delta_{w,p}^i(f)(u) = \frac{1}{2} \sum_{v \in V} w_{uv} \left( \|\nabla_w(f)(v)\|_2^{p-2} + \|\nabla_w(f)(u)\|_2^{p-2} \right) (f(v) - f(u)). \quad (2.18)$$

Pour certaines valeurs de  $p$ , on retrouve certaines définitions discrètes de la littérature :

- Pour  $p = 2$ , on peut écrire le  $p$ -Laplacien isotrope sous la forme :

$$\Delta_{w,2}^i(f)(u) = \frac{1}{2} \operatorname{div}_w \left( d_w(f) \right)(u) = \Delta_w(f)(u), \quad (2.19)$$

où  $\Delta_w$  est le Laplacien combinatoire sur graphe [Chu97], s'exprimant pour un sommet  $u \in V$  et une fonction  $f \in \mathcal{H}(V)$  par :

$$\Delta_w(f)(u) = \sum_{v \sim u} w_{uv} (f(v) - f(u)). \quad (2.20)$$

- Pour  $p = 1$ , le  $p$ -Laplacien isotrope s'écrit :

$$\Delta_{w,1}^i(f)(u) = \frac{1}{2} \operatorname{div}_w \left( \|\nabla_w(f)\|_2^{-1} d_w(f) \right) (u) = \frac{1}{2} \kappa_w^i(f)(u), \quad (2.21)$$

où  $\kappa_w^i$  est l'opérateur de courbure isotrope discret, défini en un sommet  $u \in V$  pour une fonction  $f \in \mathcal{H}(V)$  tel que :

$$\kappa_w^i(f)(u) = \sum_{v \in V} w_{uv} \left[ \frac{1}{\|\nabla_w(f)(v)\|_2} + \frac{1}{\|\nabla_w(f)(u)\|_2} \right] (f(v) - f(u)). \quad (2.22)$$

On retrouve ici l'analogie discret, pour  $w_{uv} = 1$ , de l'opérateur de courbure utilisé en traitement d'image par [OS00, CV+01].

### $p$ -Laplacien anisotrope

Le  $p$ -Laplacien anisotrope sur graphe d'une fonction  $f \in \mathcal{H}(V)$ , noté  $\Delta_{w,p}^a : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$ , est défini tel que :

$$\Delta_{w,p}^a(f)(u) = \frac{1}{2} \operatorname{div}_w \left( |d_w(f)|^{p-2} d_w(f) \right) (u), \quad \text{où } 1 \leq p < \infty. \quad (2.23)$$

En utilisant les définitions (2.6) et (2.11), on peut réécrire le  $p$ -Laplacien anisotrope en un sommet  $u \in V$  tel que :

$$\Delta_{w,p}^a(f)(u) = \sum_{v \in V} \sqrt{w_{uv}^p} |f(v) - f(u)|^{p-2} (f(v) - f(u)). \quad (2.24)$$

Pour  $p = 2$ , le  $p$ -Laplacien anisotrope devient :

$$\Delta_{w,2}^a(f)(u) = \sum_{v \in V} w_{uv} (f(v) - f(u)). \quad (2.25)$$

On remarque ici que  $\Delta_{w,2}^a = \Delta_{w,2}^i$ .

### Écriture unifiée

Ces deux formulations du  $p$ -Laplacien sur graphe peuvent s'écrire de la même manière :

$$\Delta_{w,p}(f)(u) = \sum_{v \in V} \gamma_p(f)(u, v) (f(v) - f(u)). \quad (2.26)$$

On peut retrouver les  $p$ -Laplacien isotrope et anisotrope en utilisant :

- $\gamma_p^i(f)(u, v) = \frac{1}{2} w_{uv} \left( \|\nabla f(v)\|_2^{p-2} + \|\nabla f(u)\|_2^{p-2} \right)$  pour le  $p$ -Laplacien isotrope.
- $\gamma_p^a(f)(u, v) = \sqrt{w_{uv}^p} |f(v) - f(u)|^{p-2}$  pour le  $p$ -Laplacien anisotrope.

### **$p$ -Laplacien normalisé**

En modifiant les expressions, soit des produits scalaires de  $\mathcal{H}(V)$  et  $\mathcal{H}(E)$ , soit de l'opérateur de différence pondéré, en définissant :

$$d_w(f)(u, v) = \frac{\sqrt{w_{uv}}}{\sum_{z \sim u} \sqrt{w_{uz}}} (f(v) - f(u)), \quad (2.27)$$

on peut définir une forme de  $p$ -Laplacien normalisé sur graphe :

$$\Delta_{w,p}^n(f)(u) = \sum_{v \in V} \left( \frac{\sqrt{w_{uv}}}{\sum_{z \in V} \sqrt{w_{uz}}} \right)^p |f(v) - f(u)|^{p-2} (f(v) - f(u)). \quad (2.28)$$

On peut aussi proposer une version du  $p$ -Laplacien normalisé en utilisant des opérateurs statistiques, tel que défini dans [EDL12] en relation avec le jeu du Tug-of-War, cependant, afin d'introduire l'un des opérateurs que nous proposons dans cette thèse, nous reviendrons sur cet opérateur au chapitre 3.

## **2.2 Opérateurs directionnels**

Dans cette section nous introduisons les opérateurs de différences et de gradients directionnels, définis par [TEL11]. Ces opérateurs ont été introduits en relation avec la morphologie mathématique (MM). Les deux opérateurs fondamentaux en MM sont la dilatation ( $\delta_B$ ) et l'érosion ( $\varepsilon_B$ ). Ils sont définis pour une fonction  $f^0 : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ ,  $x \in \Omega$  et  $B \subset \Omega$  un élément structurant tels que :

$$\delta_B : \sup\{f^0(x - z) : z \in B\} \text{ et } \varepsilon_B : \inf\{f^0(x + z) : z \in B\}. \quad (2.29)$$

Ces deux opérateurs forment la base de divers processus morphologiques, tels que l'ouverture ou la fermeture, ou encore les filtres de reconstruction [Hei94].

Il existe une alternative à l'érosion et la dilatation algébrique, basée sur des équations aux dérivées partielles non-linéaires [BM92, VDBS94, AGLM93]. La définition des processus morphologiques par des EDPs offre plusieurs avantages. Premièrement, elle offre d'excellents résultats en utilisant des éléments structurant dont la forme ne peut être correctement représentée sur une grille discrète. Deuxièmement, elle permet une précision sous-pixelique. Et troisièmement, on peut l'utiliser adaptativement, par exemple en introduisant dans les équations un terme de vitesse d'évolution locale [BBW07].



Soit  $B = \{z : \|z\|_p \leq 1\}$  un ensemble structurant induit par la norme  $\mathcal{L}_p$ . La dilatation et l'érosion peuvent être obtenues en utilisant les EDPs suivantes :

$$\frac{\partial f(x, t)}{\partial t} = \|\nabla f(x, t)\|_p, \quad (2.30)$$

pour la dilatation, et :

$$\frac{\partial f(x, t)}{\partial t} = -\|\nabla f(x, t)\|_p, \quad (2.31)$$

pour l'érosion, où  $f$  est une version modifiée de  $f^0$  avec comme condition initiale  $f(\cdot, 0) = f^0(\cdot)$ .

### 2.2.1 Opérateurs de différences directionnelles

Afin d'adapter sur graphe les processus morphologiques basés sur les EDPs, les auteurs de [TEL11] ont défini deux opérateurs de différences directionnelles d'une fonction  $f \in \mathcal{H}(V)$ , notés  $d_w^+$  pour l'opérateur externe et  $d_w^-$  pour l'opérateur interne, avec  $d_w^+, d_w^- : \mathcal{H}(V) \rightarrow \mathcal{H}(E)$  :

$$d_w^+(f)(u, v) = \sqrt{w_{uv}}(f(v) - f(u))^+, \quad (2.32)$$

et

$$d_w^-(f)(u, v) = \sqrt{w_{uv}}(f(v) - f(u))^- , \quad (2.33)$$

où  $(x)^+ = \max(x, 0)$  et  $(x)^- = -\min(x, 0)$ . Ces deux opérateurs vérifient la propriété  $d_w^+(f)(u, v) = d_w^-(f)(v, u)$ . Ils permettent de retrouver les opérateurs de différences classiques et les étendent aux graphes pondérés, permettant une adaptativité plus fine lors du calcul des différences.

### 2.2.2 Opérateurs gradients directionnels

À partir des opérateurs de différences directionnelles, on peut définir les gradients externes et internes, notés  $\nabla_w^+$  et  $\nabla_w^-$ , respectivement :

$$\nabla_w^+(f)(u) = (d_w^+(f)(u, v))_{v \in V}, \quad (2.34)$$

pour le gradient externe et

$$\nabla_w^-(f)(u) = (d_w^-(f)(u, v))_{v \in V}, \quad (2.35)$$

pour le gradient interne.

### Normes

Comme pour l'opérateur de gradient pondéré (2.13), nous rappelons les normes  $\mathcal{L}_p$  et  $\mathcal{L}_\infty$  de ces opérateurs. Ces normes sont définies en un sommet  $u$  par :

$$\|\nabla_w^\pm(f)(u)\|_p = \left( \sum_{v \sim u} \sqrt{(w_{uv})^p} |(f(v) - f(u))^\pm|^p \right)^{\frac{1}{p}}, \quad (2.36)$$

pour  $1 \leq p < \infty$  et :

$$\|\nabla_w^\pm(f)(u)\|_\infty = \max_{v \sim u} \left( \sqrt{w_{uv}} |(f(v) - f(u))^\pm| \right). \quad (2.37)$$

### Relations avec le gradient pondéré

La relation entre le gradient pondéré et sa variante directionnelle, pour une fonction  $f \in \mathcal{H}(V)$ , est donnée par :

$$\|\nabla_w(f)(u)\|_p^p = \|\nabla_w^+(f)(u)\|_p^p + \|\nabla_w^-(f)(u)\|_p^p, \quad (2.38)$$

pour  $1 \leq p < \infty$ . On peut déduire que :

$$\|\nabla_w^\pm(f)(u)\|_p \leq \|\nabla_w(f)(u)\|_p. \quad (2.39)$$

De même, pour  $p = \infty$  la relation entre la norme de l'opérateur gradient et les opérateurs de gradients directionnels est donnée par :

$$\|\nabla_w(f)(u)\|_\infty = \max \left( \|\nabla_w^+(f)(u)\|_\infty, \|\nabla_w^-(f)(u)\|_\infty \right). \quad (2.40)$$

### Processus morphologiques sur graphe

En s'inspirant des EDPs décrivant la dilatation (2.30) et l'érosion (2.31), les auteurs de [TEL08] ont proposé une adaptation discrète des processus morphologiques sur graphe à partir des gradients directionnels. Cette adaptation, pour une fonction  $f \in \mathcal{H}(V)$ , est définie par :

$$\frac{\partial f(u)}{\partial t} = +\|\nabla_w^+(f)(u)\|_p, \quad (2.41)$$

pour la dilatation, et :

$$\frac{\partial f(u)}{\partial t} = -\|\nabla_w^-(f)(u)\|_p, \quad (2.42)$$

pour l'érosion. Les auteurs de [TEL11], dans le cas particulier d'un graphe non pondéré ( $w_{uv} = 1, \forall (u, v) \in E$ ), ont montré le lien entre les opérateurs

morphologiques algébriques (2.29) et la norme  $\mathcal{L}_\infty$  des gradients directionnels (2.37). En effet, en considérant un élément structurant  $B$  comme le voisinage d'un sommet  $N(u)$ , les normes  $\mathcal{L}_\infty$  de ces gradients correspondent aux définitions des gradients externes et internes algébriques [HNTV92] :

$$\begin{aligned}\|\nabla_w^+(f)(u)\|_\infty &= \max_{v \sim u} (f(v), f(u)) - f(u) \\ &= \delta_B(f)(u) - f(u)\end{aligned}\tag{2.43}$$

$$\begin{aligned}\|\nabla_w^-(f)(u)\|_\infty &= f(u) - \min_{v \sim u} (f(v), f(u)) \\ &= f(u) - \varepsilon_B(f)(u).\end{aligned}\tag{2.44}$$

En utilisant un schéma de discrétisation explicite pour la dérivée temporelle dans les équations (2.41) et (2.42), on obtient :

$$f^{(n+1)}(u) = f^n(u) \pm \Delta t \|\nabla_w^\pm(f^n)(u)\|_p,\tag{2.45}$$

avec  $f^n = f(\cdot, t_0 + n\Delta t)$ .

En utilisant la réécriture de la norme  $\mathcal{L}_\infty$  des gradients directionnels pour un graphe non pondéré ((2.43) et (2.44)), ainsi que  $\Delta t = 1$ , on retrouve les opérations morphologiques algébriques usuelles :

$$f^{(n+1)}(u) = f^n(u) + \|\nabla_w^+(f^n)(u)\|_\infty = \delta_B(f)(u),\tag{2.46}$$

pour la dilatation, et :

$$f^{(n+1)}(u) = f^n(u) - \|\nabla_w^-(f^n)(u)\|_\infty = \varepsilon_B(f)(u),\tag{2.47}$$

pour l'érosion.

On peut aussi interpréter ces équations en utilisant la définition des frontières de sommets. Soit  $\mathcal{A} \subset V$  un sous ensemble de sommets et  $f : V \rightarrow [0, 1]$  initialisée telle que  $f = \chi_{\mathcal{A}}$ , avec  $\chi_{\mathcal{A}}$  la fonction caractéristique de l'ensemble  $\mathcal{A}$ . L'équation de dilatation appliquée à  $f$  peut être interprétée comme un processus de croissance déplaçant les sommets de  $\partial^+ \mathcal{A}$  vers  $\mathcal{A}$  (en interprétant  $f$  comme la fonction caractéristique de l'ensemble  $\mathcal{A}$ ), à l'inverse l'équation d'érosion peut être interprétée comme un processus de contraction qui déplace des sommets de  $\mathcal{A}$  vers  $\partial^- \mathcal{A}$ . Ces processus sont illustrés à la figure 2.1.

### 2.2.3 Laplacien infini sur graphe

Le Laplacien infini est lié à l'EDP du Laplacien infini, exprimée telle que :

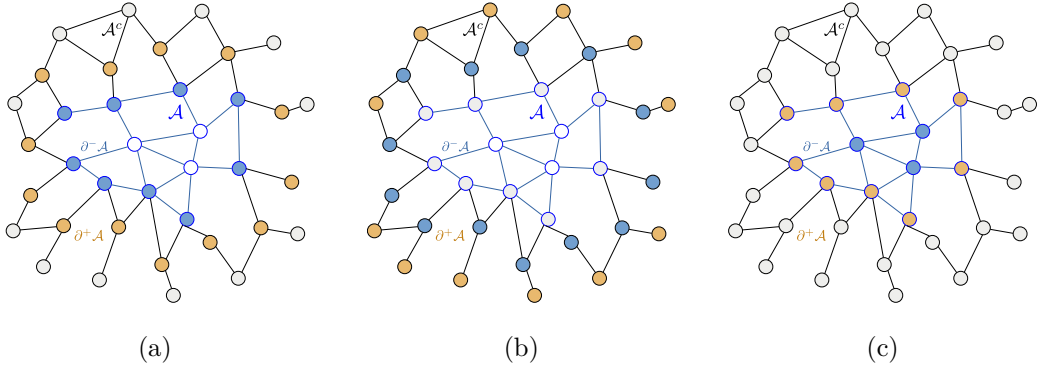


FIGURE 2.1 – Illustration de la dilatation et de l'érosion d'un ensemble de sommet. (a) Ensemble de sommet  $V$  et sous ensemble  $\mathcal{A}$ ; (b) Dilatation de l'ensemble  $\mathcal{A}$ ; (c) Érosion de l'ensemble  $\mathcal{A}$ .

$$\Delta_\infty f = 0, \quad (2.48)$$

où

$$\Delta_\infty f = \sum_{i,j}^m \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} \frac{\partial^2 f}{\partial x_i \partial x_j}. \quad (2.49)$$

[Aro67] a interprété formellement cette équation (2.48) comme la limite lorsque  $p$  tend vers l'infini de l'équation d'Euler-Lagrange :

$$\Delta_p f = \operatorname{div}(\|\nabla f\|_2^{p-2} \nabla f) = 0, \quad (2.50)$$

associée au problème de Dirichlet :

$$\inf_{f \in W^{1,p}(\Omega), f=b \text{ sur } \partial\Omega} \|\nabla f\|_{L,p(\Omega)}, \quad (2.51)$$

où  $1 < p < \infty$ .

Le Laplacien infini sur graphe, proposé par [EDLL11], est défini pour une fonction  $f \in \mathcal{H}(V)$  tel que :

$$\Delta_{w,\infty} f(u) = \frac{1}{2} \left[ \|\nabla_w^+(f)(u)\|_\infty - \|\nabla_w^-(f)(u)\|_\infty \right]. \quad (2.52)$$

Dans [EDLL11, EDLL14], les auteurs ont montré que, similairement au cas continu, cette définition correspond à la limite quand  $p$  tend vers l'infini de l'opérateur  $p$ -Laplacien.

## 2.3 Opérateurs moyennés non-locaux

Nous introduisons dans cette section la base des opérateurs qui seront utilisés dans la suite de ce manuscrit, notamment aux chapitres 3 et 4. Ces opérateurs sont basés sur les définitions précédentes des opérateurs et EdPs correspondantes.

### 2.3.1 Régularisation par $p$ -Laplacien

Soient  $G = (V, E, w)$  un graphe pondéré et  $f^0 \in \mathcal{H}(V)$  une fonction donnée de  $\mathcal{H}(V)$ . Dans la plupart des applications (acquisitions de données, transmission, etc.),  $f^0$  représente des données corrompues par un bruit. Nous considérons dans cette section le cas d'un bruit additif  $\mu \in \mathcal{H}(V)$ , tel que  $f^0 = h + \mu$ , où  $h \in \mathcal{H}(V)$  est la version non bruitée de  $f^0$ . Afin d'obtenir ou d'approximer la fonction inconnue  $h$ ,  $f^0$  est régularisée en cherchant à obtenir une fonction  $f \in \mathcal{H}(V)$  qui soit à la fois régulière sur  $G$  et proche de la fonction initiale  $f^0$ . On peut formaliser ce problème d'optimisation en utilisant l'énergie suivante :

$$E_{w,p}(f, f^0, \lambda) = R_{w,p}(f) + \frac{\lambda}{2} \|f - f^0\|_2^2, \quad (2.53)$$

tel que

$$h \approx \arg \min_{f: V \rightarrow \mathbb{R}} E_{w,p}(f, f^0, \lambda). \quad (2.54)$$

Le premier terme de l'énergie (2.53),  $R_{w,p}$  mesure la régularité de la fonction  $f$  sur le graphe  $G$ , le second mesure la proximité entre  $f$  et la fonction initiale  $f^0$ . Le paramètre  $\lambda \geq 0$  est un paramètre de fidélité spécifiant le compromis entre les deux termes de (2.53).

On peut mesurer la régularité de la solution désirée  $f$  en utilisant la  $p$ -énergie de Dirichlet, basée sur les variations locales de  $f$ , définie telle que :

$$\mathcal{R}_{w,p}(f) = \frac{1}{2p} \sum_{u \in V} \|\nabla_w(f)(u)\|_p^p. \quad (2.55)$$

Cette énergie est l'analogie discret pondéré de la  $p$ -énergie de Dirichlet de fonctions définies sur un domaine continu borné  $\Omega$  de l'espace Euclidien :  $\frac{1}{2p} \int_{\Omega} \|\nabla(f)(x)\|_p^p dx, f : \Omega \subset \mathbb{R}^m \rightarrow \mathbb{R}$ .

Lorsque  $p = 2$ , la fonctionnelle de régularisation (2.55) correspond à l'énergie de Dirichlet et le minimiseur (2.54) correspond à la régularisation de Tikhonov. Lorsque  $p = 1$ , on obtient la variation totale pour (2.55) et le modèle ROF pour l'énergie à minimiser (2.54).

Pour  $p \geq 1$ , les deux fonctionnelles du minimiseur (2.53) sont strictement convexes. Si la solution du problème (2.54) existe, alors elle est unique. La solution de ce problème de minimisation peut être obtenue en utilisant l'équation d'Euler-Lagrange de (2.53) :

$$\frac{\partial}{\partial f(u)} E_{w,p}(f, f^0, \lambda) = \frac{\partial}{\partial f(u)} \mathcal{R}_{w,p}(f) + \lambda(f(u) - f^0(u)) = 0. \quad (2.56)$$

Le terme correspondant à la variation du terme de régularisation par rapport à la fonction peut être calculé à l'aide du  $p$ -Laplacien sur graphe :

$$\frac{\partial}{\partial f(u)} \mathcal{R}_{w,p}(f) = -\Delta_{w,p}(f)(u). \quad (2.57)$$

On peut ainsi réécrire l'équation (2.56) sous la forme du système linéaire suivant :

$$\lambda f^0(u) = \left( \lambda + \sum_{v \sim u} \gamma_p^a(f)(u, v) \right) f(u) + \sum_{v \sim u} \gamma_p^a(f)(u, v) f(v) \quad (2.58)$$

Comme dans le cas continu, la solution au problème de minimisation peut être formulé comme un processus de diffusion. Les auteurs de [ELB08, BEM09] ont proposé le schéma itératif suivant, basé sur l'algorithme de Gauss-Jacobi :

$$\begin{cases} f^{(0)} = f^0 \\ f^{(n+1)}(u) = \frac{\lambda f^0(u) + \sum_{v \sim u} \gamma_p^a(f^{(n)})(u, v) f(v)}{\lambda + \sum_{v \sim u} \gamma_p^a(f^{(n)})(u, v)} \end{cases} \quad (2.59)$$

À chaque itération, la nouvelle valeur  $f^{(n+1)}$  en un sommet  $u$  dépend de deux quantités : la valeur de la fonction originale  $f^0(u)$  et une somme pondérée des valeurs existantes dans le voisinage de  $u$ . Ce schéma itératif décrit une famille de processus de diffusion, paramétrée par la structure de graphe, la fonction de poids, le paramètre de fidélité  $\lambda$  et le degré de régularité  $p$ . Le choix de chacun de ces paramètres permet de retrouver et d'étendre des filtres communément utilisés en traitement d'image. On peut retrouver, par exemple, le filtre bilatéral, le filtre de variation totale digitale, des filtres moyenneurs non-locaux, etc. Voir [ELB08] pour plus de détails. En particulier, pour  $p = 2$ , une itération de (2.59) correspond à :

$$f^{(n+1)}(u) = \frac{\lambda f^0(u) + \sum_{v \sim u} w_{uv} f(v)}{\lambda + \sum_{v \sim u} w_{uv}} \quad (2.60)$$

### Opérateur moyennneur non-local

Nous introduisons d'abord un opérateur moyennneur non-local, basé sur le processus de diffusion (2.59) utilisant le  $p$ -Laplacien sur graphe (2.26), pour  $p = 2$ . Lorsque le paramètre  $\lambda$  du processus de diffusion (2.59) est nul, le filtre (2.60) devient un filtre moyennneur non-local :

$$f^{(n+1)}(u) = NLMean(f^n)(u), \quad (2.61)$$

où

$$NLMean(f)(u) = \frac{\sum_{v \sim u} w_{uv} f(v)}{\sum_{v \sim u} w_{uv}} \quad (2.62)$$

est un opérateur moyennneur pondéré non-local.

### 2.3.2 Opérateurs morphologiques non-locaux

Nous introduisons ensuite deux opérateurs morphologiques non-locaux,  $NLD_\infty$  et  $NLE_\infty$ , où  $NLD_\infty, NLE_\infty : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$ , basés sur les schémas de discrétisation (2.46) et (2.47) des EdPs morphologiques (2.41) et (2.42), avec  $p = \infty$ . Ces opérateurs sont définis, pour une fonction  $f \in \mathcal{H}(V)$  en un sommet  $u \in V$  tels que :

$$NLD_\infty(f)(u) = f(u) + \|\nabla_w^+(f)(u)\|_\infty \quad (2.63)$$

et

$$NLE_\infty(f)(u) = f(u) - \|\nabla_w^-(f)(u)\|_\infty. \quad (2.64)$$

On peut remarquer que ces opérateurs correspondent à une itération des schémas morphologiques (2.46) et (2.47) :

$$f^{(n+1)}(u) = NLD_\infty(f^n)(u), \quad (2.65)$$

pour la dilatation et

$$f^{(n+1)}(u) = NLE_\infty(f^n)(u), \quad (2.66)$$

pour l'érosion.

## 2.4 Extension des filtres de choc sur graphe

Dans cette section, nous utilisons les définitions précédentes des opérateurs différentiels sur graphe afin de présenter une adaptation du filtre de choc sur graphe, que nous avons proposée dans [SET15].

### 2.4.1 Filtres de choc

En traitement d'image, le filtre de choc fait partie des filtres basés sur les EDPs. L'idée de ce filtre est d'effectuer une dilatation autour des maxima et une érosion autour des minima. Cette opération crée un "choc" entre les zones d'influences des extrema de l'image. La plupart des filtres de choc sont basés sur la définition de Kramer et Bruckner en termes de filtres de voisinage utilisant les minima/maxima [KB75] et sur la formulation d'Osher et Rudin [OR90] en termes d'EDPs.

Les filtres de choc offrent plusieurs avantages :

- Ils créent de fortes discontinuités aux frontières des régions présentes dans l'image et le signal devient plat à l'intérieur de ces régions.
- Ils satisfont le principe de minimum-maximum, l'intervalle des valeurs de l'image initiale et de l'image filtrée reste donc le même.
- Ils n'augmentent pas la norme  $\mathcal{L}_1$  de la dérivé du signal.
- Ils possèdent des propriétés de stabilité inhérentes.

Osher et Rudin [OR90] ont proposé le filtre de choc afin de renforcer les contours des régions dans une image floue  $f_0$  en utilisant l'équation suivante :

$$\begin{cases} \frac{\partial f}{\partial t}(x, t) = -\text{sign}(\Delta(f)(x, t))|\nabla f(x, t)|, \\ f(x, 0) = f_0(x), \end{cases} \quad (2.67)$$

La formulation continue du filtre de choc proposée par Kramer-Bruckner [GM01, KB75] est exprimée telle que :

$$\begin{cases} \frac{\partial f}{\partial t}(x, t) = -\text{sign}(f_{\eta\eta}(x, t))|\nabla f(x, t)|, \\ f(x, 0) = f_0(x), \end{cases} \quad (2.68)$$

où  $f_{\eta\eta}$  représente la dérivé seconde locale dans la direction du gradient de  $f$ , ce qui correspond au Laplacien infini  $\Delta_\infty f$ .

### 2.4.2 Extension sur graphe

Ces deux filtres (2.67) et (2.68) peuvent être réécrit tels que :

$$\begin{cases} \frac{\partial f}{\partial t}(x, t) = -\text{sign}(\Delta_p(f)(x, t))|\nabla(f)(x, t)|, \\ f(x, 0) = f_0(x), \end{cases} \quad (2.69)$$



où  $p \in \{2, \infty\}$ .

Comme le filtre de choc est composé de la combinaison de processus d'érosion et de dilatation, qui sont sensibles au bruit, ce filtre ne peut pas retirer certains types de bruit, tel que le bruit Gaussien ou encore le bruit "poivre et sel". Plusieurs variantes ont été proposées afin d'améliorer le filtre de choc original, tel que le toggle mapping morphologique [MS89], l'amélioration basée sur les EDPs [GSZ02] ou encore le filtre de choc améliorant la cohérence [Wei03].

Dans [SET15], nous avons proposé une discrétisation des filtres de choc classiques sur graphe pondéré en utilisant le formalisme des EdPs. Nous considérons un graphe  $G = (V, E, w)$  et une fonction initiale  $f_0$  définie sur  $V$ . En utilisant la définition de la norme des gradients directionnels (2.36) et (2.37) ainsi que celle du  $p$ -Laplacien (2.26) et du Laplacien infini sur graphe (2.52), nous avons proposé le filtre de choc sur graphe, défini sur une fonction  $f : V \subset \Omega \rightarrow \mathbb{R}$ , en utilisant l'EdP suivante :

$$\begin{cases} \frac{\partial f}{\partial t}(u, t) = (k_{p,w})^+ \|\nabla_w^+(f)(u, t)\|_p - (k_{p,w})^- \|\nabla_w^-(f)(u, t)\|_p, \\ f(u, 0) = f_0(u), \end{cases} \quad (2.70)$$

où  $p \in \{2, \infty\}$  et  $k_{p,w} = -\text{sign}(\Delta_{w,p}(f)(u))$ .

Pour  $p = 2$ , notre formulation permet de retrouver une version discrète de la formulation proposée par Osher et Rudin (2.67). Il en va de même pour  $p = \infty$ , ou notre formulation permet de retrouver une version discrète de la formulation proposée par Kramer-Bruckner (2.68). De plus, cette adaptation permet d'étendre les applications du filtre de choc à toutes données pouvant être représentées par un graphe, unifiant ainsi les traitements locaux et non-locaux.

## 2.5 Conclusion

Dans ce chapitre, nous avons rappelé certaines notions de calcul discret sur graphe, introduites pour la plupart par [ELB08, BEM09, TEL11]. Nous avons ainsi introduit la notion de fonction sur graphe, ainsi que les opérateurs nécessaires au calcul différentiel sur graphe, tels que les opérateurs de différences (directionnels ou non), de divergence, de gradients. À partir de ces opérateurs, nous avons rappelé la définition du  $p$ -Laplacien sur graphe [ELB08, BEM09], ainsi que la régularisation de fonction sur graphe associée à cet opérateur. Nous avons aussi rappelé les notions d'opérations morphologiques

## *2.5. Conclusion*

---

sur graphe, ainsi que le Laplacien infini sur graphe [EDLL11]. Enfin, nous avons présenté les différents filtres et processus morphologiques associés à ces opérateurs sur graphe. Toutes ces définitions et opérateurs sont la base des travaux que nous proposons et ils seront utilisés tout au long de ce manuscrit de thèse.



## Deuxième partie

### Une nouvelle famille de $p$ -Laplacien sur graphe



# Introduction

Durant la dernière décennie, l'intérêt pour le  $p$ -Laplacien et le Laplacien infini n'a cessé de croître. En effet, ces opérateurs, dans le cadre local continu, jouent un rôle important en géométrie et dans les équations aux dérivées partielles. Ils sont présents dans divers modèles et problèmes mathématiques et sont utilisés pour modéliser des processus importants en physique, en biologie, en économie et sont aussi utilisés en traitement d'image [Drá07, Lin06, Obe13].

Récemment, le  $p$ -Laplacien non-local a aussi suscité un intérêt croissant dans la littérature. L'étude théorique des problèmes de diffusion non-locale impliquant le  $p$ -Laplacien et le Laplacien infini non-local sont d'un grand intérêt du point de vue mathématique [AVMRTM10, CLM12]. Ces opérateurs apparaissent aussi dans une grande variété d'applications, y compris en biologie, en traitement d'image et en mécanique des milieux continus. Cette liste n'est pas exhaustive et le lecteur intéressé peut se référer au récent livre de [AVMRTM10] (et les références connexes).

Le  $p$ -Laplacien normalisé, ou *game*  $p$ -Laplacien, est une autre version du  $p$ -Laplacien continu, récemment introduit en connexion avec le jeu stochastique appelé Tug-of-War [PSSW09] (seulement sous sa forme infini), puis avec le Tug-of-War bruité [PS<sup>+</sup>08]. Une part de l'intérêt porté à cet opérateur est due au fait que pour certains cas particuliers on retrouve l'opérateur de courbure moyenne (pour  $p = 1$ ), le Laplacien infini (pour  $p = \infty$ ) et un multiple du Laplacien ordinaire (pour  $p = 2$ ).

Il existe plusieurs possibilités pour approximer les EDPs continues impliquant les différentes formulations du Laplacien sur des domaines discrets. Dans le cadre des domaines Euclidiens les schémas de discrétisation basés sur les différences finies, les éléments finis, les volumes finis, etc. ont beaucoup été étudiés et sont couramment utilisés [Tad12]. D'autre part, on peut rencontrer des processus physiques dans des environnements géométriques plus complexes, par exemple sur des nuages de points ou des surfaces. Traiter et analyser ce type de données est un challenge majeur et la discrétisation des opérateurs différentiels devient plus difficile, en comparaison des approches précédemment citées. On peut organiser les approches possibles pour ce type de données en

différentes catégories : les méthodes implicites [BCOS01, OF03, RM08], les méthodes explicites [SK07, Sta03] et les méthodes intrinsèques [LC11]. Pour plus de détails sur ces méthodes et leurs avantages respectifs, voir [LEL14b].

Il existe donc un intérêt grandissant pour l'adaptation et la résolution d'EDPs sur des données discrètes définies sur des graphes et réseaux. En effet, la plupart des données discrètes peuvent être représentées par un graphe dans lequel les sommets sont associés aux données et les arêtes correspondent aux relations entre les données. Afin de pouvoir transposer et résoudre les EDPs sur des graphes de topologies arbitraires, différentes méthodes de calcul vectoriel ont été proposées dans la littérature ces dernières années, voir [GP10] et les références connexes.

Dans cette seconde partie, nous proposons tout d'abord (au chapitre 3) une adaptation du  $p$ -Laplacien normalisé sur graphe pondéré de topologie arbitraire en utilisant le cadre des EdPs. Cette adaptation nous permet d'introduire une nouvelle classe de  $p$ -Laplacien sur graphe sous la forme d'une non-divergence. Cet opérateur interpole entre le 1-Laplacien, le Laplacien infini et le Laplacien sur graphe et est obtenu à partir d'opérateurs statistiques transposés dans le cadre des graphes. Comme beaucoup de problèmes en traitement d'images sont formulés comme des problèmes d'interpolations, nous étudions le problème de Dirichlet associé à cet opérateur et nous montrons l'existence et l'unicité de la solution à cette équation.

Nous proposons ensuite (au chapitre 4) un opérateur  $p$ -Laplacien et Laplacien infini défini comme une combinaison de termes de gradients à coefficients variables. Nous montrons que cet opérateur est une généralisation des opérateurs défini sur graphe dans le cadre des EdPs. En effet les cas particuliers de cet opérateur permettent de retrouver les  $p$ -Laplacien anisotropes et le Laplacien infini, mais aussi les opérateurs de gradients morphologiques définis précédemment au chapitre 2 et introduits par [ELB08, TEL11]. Nous montrons aussi dans ce chapitre que cet opérateur permet de retrouver différents schémas de discrétisation existants, dans le cas d'opérateurs locaux et non-locaux. Cet opérateur unifie les formulations discrètes du  $p$ -Laplacien, du Laplacien infini, du  $p$ -Laplacien normalisé et du  $p$ -Laplacien non-local sur graphe, que ce soit sur des domaines réguliers ou irréguliers.

Enfin, dans le dernier chapitre de cette partie, nous proposons une adaptation de l'équation de Poisson infini sur graphe pondéré, en utilisant la formulation du Laplacien infini avec termes de gradient sur graphe pondéré introduite au chapitre 4. Nous faisons aussi le lien entre cette équation et la version biaisée du Tug-of-War. En utilisant cette formulation, nous proposons une équation hybride de Poisson infini et de Hamilton-Jacobi. Nous montrons aussi le lien entre cette version de l'équation de Poisson infini et l'adaptation de

l'équation eikonale sur graphe pondéré [DEL13]. Notre but dans ce chapitre est d'utiliser cette extension de l'équation de Poisson afin de calculer des distances sur tout type de données discrètes pouvant être représentées par un graphe pondéré. Nous montrons, à travers différentes expérimentations, que cette formulation peut être utilisée pour la résolution de différentes applications en traitement d'image, de nuages de point 3D et de données de grandes dimensions, en utilisant une unique formulation.





# Chapitre 3

## $p$ -Laplacien normalisé sur graphe pondéré

### Sommaire

---

<b>3.1</b>	<b>Introduction</b>	<b>52</b>
<b>3.2</b>	<b><math>p</math>-Laplacien normalisé</b>	<b>53</b>
3.2.1	$p$ -Laplacien	53
3.2.2	$p$ -Laplacien normalisé	54
<b>3.3</b>	<b><math>p</math>-Laplacien normalisé sur graphe</b>	<b>55</b>
3.3.1	Opérateurs statistiques	55
3.3.2	Définition du $p$ -Laplacien normalisé sur graphe	55
3.3.3	Connexion avec différents opérateurs différentiels locaux et non-locaux	56
3.3.4	Relation avec les jeux de type Tug-of-War	60
<b>3.4</b>	<b>Problème de Dirichlet associé</b>	<b>64</b>
3.4.1	Existence et unicité de la solution	64
<b>3.5</b>	<b>Équation de diffusion</b>	<b>67</b>
3.5.1	Équation	68
3.5.2	Propriétés	69
<b>3.6</b>	<b>Expérimentations</b>	<b>72</b>
3.6.1	Filtrage et simplification d'image et de données	72
3.6.2	Interpolation sur graphe	75
3.6.3	Inpainting non-local	76

## Publication associée à ce chapitre

[TELM15] Abderrahim ELMOATAZ et Matthieu TOUTAIN : Normalized  $p$ -Laplacian on Weighted Graphs with Applications in Image Processing and Machine Learning. *Applied and computational harmonic analysis*, pages 1–42, under review.

### 3.1 Introduction

Le  $p$ -Laplacien normalisé, récemment introduit sous sa forme infini en lien avec le jeu stochastique appelé Tug-of-War par [PSSW09], puis avec le Tug-of-War bruité [PS<sup>+</sup>08] est une version normalisée du  $p$ -Laplacien. Dans le cas d’une équation homogène, la solution de l’EDP coïncide avec celle du  $p$ -Laplacien, pour lequel plusieurs approximations ont déjà été proposées. Certains de ces schémas sont basés sur les éléments finis [BL93]. Plusieurs schémas d’approximations utilisant les différences finies ont aussi été proposés pour le  $p$ -Laplacien normalisé, pour  $p = 1$ ,  $p = \infty$  et  $p \geq 2$  [Obe13]. Plus récemment un schéma d’approximation basé sur le semi-Lagrangien a été proposé par [FVGS13]. On peut aussi citer les approximations du  $p$ -Laplacien normalisé pour  $1 \leq p \leq \infty$  par des opérateurs statistiques [Rud14].

Dans ce chapitre, motivé par la volonté d’étendre cet opérateur sur tout type de domaines discrets, nous proposons une adaptation et une généralisation du  $p$ -Laplacien normalisé sur graphe pondéré de topologie arbitraire en utilisant le cadre des EdPs [ELB08, TEL11]. Cette adaptation peut être considérée comme une nouvelle classe de  $p$ -Laplacien sur graphe sous forme d’une non-divergence, interpolant entre le 1-Laplacien non-local, le Laplacien infini non-local et le 2-Laplacien non-local, sur graphe. Nous rappelons d’abord la définition du  $p$ -Laplacien normalisé tel que présenté par [PS<sup>+</sup>08], en faisant le lien avec le  $p$ -Laplacien variationnel. Nous proposons ensuite une définition du  $p$ -Laplacien normalisé sur graphe, à partir d’opérateurs statistiques. Nous montrons ensuite les relations entre l’opérateur proposé sur graphe et certains opérateurs différentiels locaux et non-locaux, ainsi qu’avec les jeux de type Tug-of-War.

Comme beaucoup de problèmes en traitement du signal peuvent être considérés comme des problèmes d’interpolation, nous étudions le problème de Dirichlet associé à cet opérateur et nous prouvons l’existence et l’unicité de

la solution associée. Nous étudions ensuite l'équation de diffusion associée à cet opérateur et nous montrons que le processus de résolution numérique de cette équation permet de retrouver divers filtres utilisés en traitement d'image. Enfin, nous proposons d'utiliser cet opérateur pour résoudre différents problèmes d'interpolations avec des applications en traitement d'image et en classification semi-supervisée.

## 3.2 *p*-Laplacien normalisé

### 3.2.1 *p*-Laplacien

Le *p*-Laplacien joue un rôle important en géométrie continue ainsi que dans le champ des équations aux dérivées partielles utilisées pour décrire divers phénomènes en physique et en biologie. Pour le lecteur intéressé par une introduction et plus de détails sur le sujet, nous recommandons la lecture de [Drá07, Lin06] ainsi que les références citées par ces derniers. Le *p*-Laplacien est un opérateur aux dérivées partielles elliptiques quasi-linéaire du second ordre. Il peut être formulé de la manière suivante :

$$\Delta_p u = \operatorname{div} \left( \frac{\nabla u}{|\nabla u|^{2-p}} \right), \quad 1 \leq p < \infty. \quad (3.1)$$

Cet opérateur découle de l'équation d'Euler-Lagrange minimisant l'énergie suivante :

$$E(u) = \frac{1}{p} \int_{\Omega} |\nabla u(x)|^p dx, \quad (3.2)$$

menant à l'EDP suivante :

$$\Delta_p u = 0, \quad (3.3)$$

avec des conditions aux bords de Dirichlet.

Beaucoup de problèmes inverses en traitement d'image, tel que le débruitage, la déconvolution, la segmentation ou l'inpainting sont formulés à l'aide de termes de régularisations basés sur des formulations faibles du *p*-Laplacien (3.1). On peut remarquer que pour  $p = 2$ , on retrouve l'opérateur de Laplace classique  $\Delta$ .

Le *p*-Laplacien est aussi lié à la régularisation de Tikhonov (pour  $p = 2$ ) ainsi qu'à la régularisation par variation totale (pour  $p = 1$ ). Pour  $p = \infty$ , le Laplacien infini est défini tel que [Aro67] :

$$\Delta_{\infty} u = \sum_{i=1}^m \sum_{j=1}^m \frac{\partial u}{\partial x_i} \frac{\partial u}{\partial x_j} \frac{\partial^2 u}{\partial x_i \partial x_j}. \quad (3.4)$$

### 3.2.2 $p$ -Laplacien normalisé

Le  $p$ -Laplacien normalisé, aussi appelé  $p$ -Laplacien des jeux, récemment introduit par [PS<sup>+</sup>08] pour modéliser des jeux stochastiques appelés Tug-of-War, s'écrit, pour  $1 \leq p < \infty$  :

$$\Delta_p^N f = \frac{1}{p} |\nabla f|^{2-p} \cdot \operatorname{div}(|\nabla f|^{p-2} \cdot \nabla f). \quad (3.5)$$

Lorsque  $p = \infty$ , le  $p$ -Laplacien normalisé s'écrit :

$$\Delta_\infty^N f = |\nabla f|^{-2} \Delta_\infty f. \quad (3.6)$$

On dit que  $\Delta_p^N f$  est normalisé parce qu'il est homogène de degré 1, *i.e.*  $\Delta_p^N(t f) = t \cdot \Delta_p^N f$  avec  $t \in \mathbb{R}$ , ce qui contraste avec le  $p$ -Laplacien qui est homogène de degré  $p - 1$ . Ainsi, les problèmes paraboliques impliquant le  $p$ -Laplacien normalisé sont invariants en échelle. Si  $f$  est une fonction lisse, l'équation 3.5 peut se réécrire de la façon suivante :

$$\begin{aligned} \Delta_p^N f &= \frac{(p-2)}{p} \Delta_\infty^N f + \frac{1}{p} \Delta f \\ &= \frac{(p-2)}{p} \Delta_\infty^N f + \frac{2}{p} \Delta_2^N f \\ &= \alpha(p) \Delta_\infty^N f + \beta(p) \Delta_2^N f \end{aligned} \quad (3.7)$$

avec  $\alpha(p) = (p-2)/p$  et  $\beta(p) = 2/p$ .

Le  $p$ -Laplacien des jeux pour  $p = 1$  peut s'écrire comme :

$$\Delta_1^N f = \operatorname{div}\left(\frac{\nabla(f)}{|\nabla(f)|}\right) |\nabla(f)|. \quad (3.8)$$

Avec la relation  $\Delta_1^N f = \Delta - \Delta_\infty^N f$ , l'équation (3.7) peut être réécrite telle que :

$$\Delta_p^N f = \alpha'(p) \Delta_2^N f + \beta'(p) \Delta_1^N f, \quad (3.9)$$

avec  $\alpha'(p) = \frac{(p-1)}{p}$  et  $\beta'(p) = \frac{2-p}{p}$ .

Nous remarquons que pour l'équation (3.7), afin que les valeurs des fonctions  $\alpha$  et  $\beta$  soient positives, nous considérons l'intervalle des valeurs pour la variable  $p$  tel que  $2 \leq p \leq \infty$ . Similairement, pour l'équation (3.9), l'intervalle des valeurs positives pour  $\alpha'$  et  $\beta'$  est  $1 \leq p \leq 2$ .

### 3.3 $p$ -Laplacien normalisé sur graphe

#### 3.3.1 Opérateurs statistiques

Afin de proposer une définition du  $p$ -Laplacien normalisé sur graphe, nous introduisons en premier lieu des opérateurs statistiques sur graphe, complétant ceux déjà définis au chapitre précédent ( $NLMean$  (2.62),  $NLD_\infty$  (2.63) et  $NLE_\infty$  (2.64)). Ces opérateurs seront utilisés afin de définir les différents opérateurs que nous proposons dans ce chapitre, ainsi que pour simplifier l'écriture des preuves d'existence et d'unicité de la solution au problème de Dirichlet introduit plus tard.

$$\begin{aligned} NLMidrange(f)(u) &= \frac{1}{2}(NLE_\infty(f)(u) + NLD_\infty(f)(u)), \\ NLMedian(f)(u) &= \text{median}(\nabla_w(f)(u)) + f(u), \end{aligned} \quad (3.10)$$

avec  $\text{median}$  l'opérateur utilisé en théorie des statistiques et probabilité.

Ces opérateurs sont des extensions sur graphe des opérateurs statistiques classiques. En affectant  $w(u, v) = 1$ , on peut voir que l'on retrouve les filtres statistiques Median et Midrange classiques.

#### 3.3.2 Définition du $p$ -Laplacien normalisé sur graphe

Afin de proposer une version discrète de (3.5), nous proposons de définir les versions normalisées du 1-Laplacien, 2-Laplacien et  $\infty$ -Laplacien telles que :

$$\begin{aligned} \Delta_{w,2}^N(f)(u) &= NLMean(f)(u) - f(u), \\ \Delta_{w,1}^N(f)(u) &= NLMedian(f)(u) - f(u), \\ \Delta_{w,\infty}^N(f)(u) &= NLMidrange(f)(u) - f(u). \end{aligned} \quad (3.11)$$

En utilisant les relations (3.7) et (3.9) ainsi que les opérateurs discrets que nous venons de définir (3.11), nous proposons une version du  $p$ -Laplacien normalisé sur graphe pondéré :

$$\Delta_{w,p}^N(f)(u) = \begin{cases} \frac{2}{p}\Delta_{w,2}^N(f)(u) + \frac{p-2}{p}\Delta_{w,\infty}^N(f)(u) & \text{si } 2 \leq p \leq \infty, \\ \frac{p-1}{p}\Delta_{w,2}^N(f)(u) + \frac{2-p}{p}\Delta_{w,1}^N(f)(u) & \text{si } 1 \leq p \leq 2. \end{cases} \quad (3.12)$$

En utilisant les définitions (3.11) et (3.12), le  $p$ -Laplacien normalisé sur graphe pondéré peut se réécrire :

$$\Delta_{w,p}^N(f)(u) = NLA(f)(u) - f(u), \quad (3.13)$$

avec  $NLA(f)(u)$  l'opérateur (nonlocal average) défini tel que :

$$NLA(f)(u) = \begin{cases} \frac{2}{p}NLMean(f)(u) + \frac{p-2}{p}NLMidrange(f)(u), \\ \text{pour } 2 \leq p \leq \infty, \\ \frac{2-p}{p}NLMedian(f)(u) + \frac{p-1}{p}NLMean(f)(u), \\ \text{pour } 1 \leq p \leq 2. \end{cases} \quad (3.14)$$

### 3.3.3 Connexion avec différents opérateurs différentiels locaux et non-locaux

Dans la section précédente, nous avons proposé une version discrète du  $p$ -Laplacien normalisé. Dans cette section, nous montrons que cet opérateur est une version discrète de différents opérateurs différentiels continus locaux et non-locaux.

#### Laplacien anisotrope normalisé

Soit  $\Omega$  un domaine borné, lisse et convexe de  $\mathbb{R}^m$ ,  $f : \Omega \rightarrow \mathbb{R}$  une fonction donnée. Le  $p$ -Laplacien normalisé anisotrope peut être exprimé de la façon suivante :

$$\Delta_p^N(f) = \frac{1}{p}|\nabla f|^{2-p} \sum_{i=1}^m \frac{\partial}{\partial x_i} \left[ \left| \frac{\partial f}{\partial x_i} \right|^{p-2} \frac{\partial f}{\partial x_i} \right]. \quad (3.15)$$

On peut discrétiser cette expression en utilisant l'approximation centrée de la méthode des différences finies :

$$\frac{\partial}{\partial x_i} f(x) \approx D_i(f)(x) = \frac{f(x + h_i/2) - f(x - h_i/2)}{h_i}. \quad (3.16)$$

Avec  $p = 2$  et en utilisant un pas de discrétisation spatial constant  $h_i = h$ , on obtient l'opérateur discret suivant :

$$\Delta_2^N(f)(x) = \frac{1}{2h^2} \left[ \sum_{i=1}^m f(x_i + h) + f(x_i - h) \right] - \frac{m}{h^2} f(x). \quad (3.17)$$

Nous montrons maintenant le lien entre cet opérateur discret et le Laplacien normalisé sur graphe que nous proposons. Soit un graphe pondéré  $G_g(V, E, w)$  représentant une grille à  $m$  dimensions. Soit  $u$  un sommet associé à un vecteur de coordonnées à  $m$  dimensions :  $u = (i_1 h_1, \dots, i_m h_m)^T$  avec  $i_j \in \mathbb{N}$  et  $h_j$  le pas spatial de la grille sur la dimension  $j$  et  $j = 1, \dots, m$ . Le voisinage de  $u$  peut être défini tel que :

$$N_g(u) = \{v : v = u \pm h_j e_j\}_{j=1, \dots, m}, \quad (3.18)$$

avec  $e_j = (q_k)_{k=1, \dots, m}^T$  un vecteur tel que  $q_k = 1$  si  $j = k$  et  $q_k = 0$  sinon.

En utilisant cette construction de graphe et en définissant la fonction de poids  $w_1$  telle que :

$$w_1(u, v_i) = 1, \quad (3.19)$$

on peut réécrire l'opérateur (3.12) avec  $p = 2$  tel que :

$$\begin{aligned} \Delta_{w_1, 2}^N(f)(u) &= \frac{\sum_{v \sim u} w_1(u, v) f(v)}{\sum_{v \sim u} w_1(u, v)} - f(u) \\ &= \frac{1}{2m} \left[ \sum_{i=1}^m f(v_i^+) + f(v_i^-) \right] - f(u), \end{aligned} \quad (3.20)$$

avec  $v_i^\pm = u \pm h_i e_i$ .

En considérant un pas spatial constant  $h_i = h \forall i \in \{1, \dots, m\}$  on retrouve la formulation discrète du 2-Laplacien normalisé (3.17) grâce à la relation suivante :

$$\Delta_{w_1, 2}^N(f) = \frac{h^2}{m} \Delta_2^N f. \quad (3.21)$$

### Laplacien non-local

Considérons maintenant un graphe Euclidien complet  $G = (V, E, w)$ . Pour  $p = 2$ , l'équation (3.13) peut être réécrite :

$$\Delta_{w, 2}^N(f)(u) = \frac{\sum_{v \in V} w(u, v) (f(v) - f(u))}{\sum_{v \in V} w(u, v)}. \quad (3.22)$$

Nous rappelons maintenant la définition du 2-Laplacien non-local d'une fonction  $f : \Omega \rightarrow \mathbb{R}$ , introduit par [AMRT08] :



$$\mathcal{L}_2(f)(x, t) = \int_{\Omega} J(x - y)(f(y, t) - f(x, t))dy. \quad (3.23)$$

En remplaçant le terme  $J(x - y)$  de l'équation (3.23) par  $w(u, v)$ , on peut voir que notre formulation (équation (3.22)) est l'analogie discret de (3.23) normalisé par le degré :

$$\Delta_{w,2}^N(f)(u) = \frac{\mathcal{L}_2(f)(u)}{\sum_{v \in V} w(u, v)}. \quad (3.24)$$

### 1-Laplacien

En utilisant notre formulation de  $\Delta_{w,1}^N$ , nous pouvons faire l'analogie avec le 1-Laplacien tel que défini dans [Rud14]. Soient  $\Omega \subset \mathbb{R}^m$ ,  $x \in \Omega$ ,  $u : \Omega \rightarrow \mathbb{R}$  une fonction lisse ayant un gradient non nul en  $x$ ,  $\epsilon = \sqrt{2h}$  et  $B_\epsilon(x)$  une boule de rayon  $\epsilon$  centrée en  $x$ . Les auteurs de cet article montrent la relation suivante :

$$u(x) - \operatorname{median}_{y \in \partial B_\epsilon(x)}(u(y)) = -\frac{h}{m-1} \Delta_1^N u(x) + o(h), \quad (3.25)$$

On peut réécrire cette relation telle que :

$$\Delta_1^N u(x) = \frac{m-1}{h} \left( \operatorname{median}_{y \in \partial B_\epsilon(x)}(u(y)) - u(x) \right) \quad (3.26)$$

Afin de montrer le lien entre notre formulation et celle définie à l'équation (3.26), nous définissons tout d'abord le sommet  $v_{\text{med}}$  tel que :

$$v_{\text{med}} = \arg \operatorname{median}_{v \sim u}(\sqrt{w(u, v)}(f(v) - f(u))), \quad (3.27)$$

avec  $\arg \operatorname{median}$  l'argument du median. En l'utilisant dans  $\Delta_{w,1}^N$  nous obtenons :

$$\Delta_{w,1}^N(f)(u) = \sqrt{w(u, v_{\text{med}})}(f(v_{\text{med}}) - f(u)). \quad (3.28)$$

Nous définissons la fonction de poids  $w_2(u, v)$  telle que :

$$w_2(u, v) = \begin{cases} \frac{(m-1)^2}{h^2}, & \text{si } v \in \partial B_\epsilon(u), \\ 0, & \text{sinon.} \end{cases} \quad (3.29)$$

En utilisant  $w_2$  dans l'opérateur  $\Delta_{w,1}^N$ , on obtient :

$$\begin{aligned}\Delta_{w_2,1}^N(f)(u) &= \frac{m-1}{h}(f(v_{\text{med}}) - f(u)) \\ \Delta_{w_2,1}^N(f)(u) &= \frac{m-1}{h} \left( \text{median}_{v \sim u}(f(v)) - f(u) \right)\end{aligned}\tag{3.30}$$

Comme on peut le voir, on retrouve exactement la même formulation que celle proposée dans [Rud14].

### $\infty$ -Laplacien

De même, dans le cas où  $p = \infty$ , on peut faire l'analogie avec la discrétisation d'Oberman du Laplacien infini [Obe13]. Soit  $f(x)$  une fonction lisse ayant un gradient non nul en  $x$ . La discrétisation effectuée par [Obe13] est telle que :

$$\Delta_{\infty} f(x) = \min_{|y-x|=\epsilon} \frac{(f(y) - f(x))}{\epsilon^2} + \max_{|y-x|=\epsilon} \frac{(f(y) - f(x))}{\epsilon^2}.\tag{3.31}$$

Afin de faire le lien avec cette formulation, nous utilisons la construction de graphe représentant une grille à  $m$  dimensions ( $G_g$ ), mais en utilisant la fonction de poids  $w_3(u, v)$  telle que :

$$w_3(u, v) = \begin{cases} \frac{4}{\epsilon^4}, & \text{si } v \in \partial^- N_{\epsilon}(u) \\ 0, & \text{sinon.} \end{cases}\tag{3.32}$$

où nous rappelons que  $N_{\epsilon}(u)$  est l' $\epsilon$ -voisinage d'un sommet, défini à l'équation (1.21). En utilisant cette fonction de poids dans  $\Delta_{w,\infty}^N$ , nous obtenons :

$$\begin{aligned}\Delta_{w_3,\infty}^N(f)(u) &= \frac{1}{2} [\max_{v \sim u} (\sqrt{w_3(u, v)}(f(v) - f(u))^+) - \max_{v \sim u} (\sqrt{w_3(u, v)}(f(v) - f(u))^-)] \\ &= \frac{1}{\epsilon^2} [\max_{v \sim u} (f(v) - f(u))^+ - \max_{v \sim u} (f(v) - f(u))^-].\end{aligned}\tag{3.33}$$

En définissant le voisinage du sommet  $u$  comme  $N(u) \cup \{u\}$ , nous obtenons :

$$\begin{aligned}\Delta_{w_3,\infty}^N(f)(u) &= \frac{1}{\epsilon^2} [\max_{v \sim u} (f(v) - f(u)) - \max_{v \sim u} (f(u) - f(v))] \\ &= \frac{1}{\epsilon^2} [\max_{v \sim u} (f(v) - f(u)) + \min_{v \sim u} (f(v) - f(u))] \\ &= \max_{v \sim u} \frac{f(v) - f(u)}{\epsilon^2} + \min_{v \sim u} \frac{f(v) - f(u)}{\epsilon^2}.\end{aligned}\tag{3.34}$$

### $\infty$ -Laplacien de Hölder

Toujours en utilisant  $p = \infty$ , mais en considérant un graphe complet, notre formulation correspond au Laplacien infini de Hölder proposé par [CLM12] :

$$\Delta_{w_4, \infty}(f)(x) = \frac{1}{2} \left[ \max_{y \in \Omega, y \neq x} \left( \frac{f(y) - f(x)}{|y - x|^s} \right) + \min_{y \in \Omega, y \neq x} \left( \frac{f(y) - f(x)}{|y - x|^s} \right) \right] \quad (3.35)$$

avec

$$w_4(x, y) = \begin{cases} \frac{1}{|y-x|^{2s}}, & \text{si } y \in \Omega \text{ et } y \neq x \\ 0, & \text{sinon.} \end{cases} \quad (3.36)$$

Cet opérateur est dérivé de la minimisation de l'énergie de la forme :

$$\int_{\Omega} \int_{\Omega} \frac{|f(y) - f(x)|^p}{|x - y|^{p \times s}} dx dy \quad (3.37)$$

avec  $p \rightarrow \infty$ . Dans [CLM12], il a été prouvé que l'équation suivante :

$$\begin{cases} \Delta_{w_4, \infty}^N(f)(x) = 0, & x \in \Omega \\ f(x) = g(x), & x \in \partial\Omega \end{cases} \quad (3.38)$$

avec des conditions sur  $g$ , possède une solution unique.

### 3.3.4 Relation avec les jeux de type Tug-of-War

Plusieurs EDPs (le  $p$ -Laplacien pour  $p \geq 2$  et le Laplacien infini) sont liées à un jeu stochastique appelé jeu du Tug-of-War (ou tir à la corde, en français). Nous montrons dans cette section que le  $p$ -Laplacien normalisé sur graphe, pour un graphe particulier, coïncide avec la fonction de valeur de ce jeu. Ce jeu est un jeu à tour aléatoire à somme nulle, nous commençons donc cette section par une description de ce type de jeu. Nous décrivons ensuite les particularités du Tug-of-War et la formulation de sa fonction de valeur comme une EDP dans un espace métrique. Nous ferons ensuite le lien entre ce jeu et le  $p$ -Laplacien normalisé sur graphe, pour  $p = \infty$ . Nous montrons ensuite que la formulation de cet opérateur permet aussi de faire le lien avec le Tug-of-War non-local, en changeant simplement la construction du graphe. Enfin nous décrivons le Tug-of-War bruité et montrons aussi le lien entre la fonction de valeur de ce jeu et le  $p$ -Laplacien normalisé sur graphe.

### Jeux à tour aléatoire

On considère deux joueurs (joueur I et joueur II). Le jeu se déroule sur un ensemble d'état  $X$ . L'ensemble des états terminaux est noté  $Y$  tel que  $Y \subset X$ . Soient  $F : Y \rightarrow \mathbb{R}$  une fonction de gain sur l'ensemble des états terminaux et  $f : X \setminus Y \rightarrow \mathbb{R}$  une fonction de paiement d'étape,  $E_I$  et  $E_{II}$  deux graphes avec comme ensemble de sommets  $X$ , représentant les transitions possibles pour chacun des joueurs. Le jeu se déroule de la manière suivante : un jeton est initialement placé en  $x_0 \in X \setminus Y$  (état initial du jeu). Au  $k^{\text{eme}}$  tour, on tire au sort équitablement le joueur qui peut jouer. Celui ci peut déplacer le jeton à la position  $x_k$  tel que  $(x_{k-1}, x_k)$  est une arête du graphe de transition. Le jeu se termine lorsque  $x_k \in Y$ . Les gains du joueur I sont définis par la somme suivante :  $F(x_k) + \sum_{i=0}^{k-1} f(x_i)$ . Le but du joueur I est de maximiser ses gains et, comme le jeu est à somme nulle, le but du joueur II est de les minimiser.

### Tug-of-War

Le Tug-of-War étant un jeu à tour aléatoire, il est défini de la même manière, avec quelques spécificités : il y a maintenant un seul graphe de transition non dirigé  $E := E_I = E_{II}$ , l'ensemble  $Y$  est l'union de deux "ensembles cibles"  $Y^I$  et  $Y^{II}$ ,  $F \equiv 1$  sur  $Y^I$  et  $F \equiv 0$  sur  $Y^{II}$ , et il n'y a en général pas de fonction de paiement d'étape ( $f = 0$ ). La valeur du jeu quand celui-ci commence en  $x$  est donnée par :

$$\begin{aligned} u_I(x) &= \sup_{S_I} \inf_{S_{II}} \hat{F}(S_I, S_{II}), \text{ pour le joueur I} \\ u_{II}(x) &= \inf_{S_{II}} \sup_{S_I} \hat{F}(S_I, S_{II}), \text{ pour le joueur II,} \end{aligned} \quad (3.39)$$

avec  $S_I$  et  $S_{II}$  les stratégies des deux joueurs et  $\hat{F}$  le gain total espéré à la fin du jeu. Une stratégie (pour un joueur) est une façon de choisir le prochain mouvement du joueur et peut être vue comme une fonction de tous les mouvements précédemment joués ainsi que de tous les tirages au sort de jeton. C'est une application de l'ensemble des coups joués vers le coup à jouer. Le jeu a une valeur quand  $u_I(x) = u_{II}(x) = u(x)$ , avec  $u$  tel que :

$$u(x) = \frac{1}{2} \left( \sup_{y:(x,y) \in E} u(y) + \inf_{y:(x,y) \in E} u(y) \right) \quad (3.40)$$

À l'origine, ce jeu a été introduit et utilisé par [PSSW09] pour prouver que toute fonction  $F$  bornée, Lipschitz et à valeurs réelles sur le sous ensemble  $Y$  d'un espace de longueur  $X$  admet une extension absolument minimale (**AM**),

*i.e.* une extension de Lipschitz  $u : X \rightarrow \mathbb{R}$  pour laquelle la constante de Lipschitz  $Lip_U u$  est égale à la constante de Lipschitz  $Lip_{\partial U} u$  pour tout ouvert  $U \subset X \setminus Y$ . Les mêmes auteurs montrent que lorsque  $X$  est la fermeture d'un domaine borné  $U \subset \mathbb{R}^m$  et que  $Y$  est sa frontière, l'extension Lipschitz  $u$  de  $F$  est AM si et seulement si elle est infini harmonique à l'intérieur de  $X \setminus Y$ , *i.e.*  $u$  est la solution de viscosité de  $\Delta_\infty u = 0$ .

### Tug-of-War dans un espace métrique

Le Tug-of-War dans un espace métrique, ou  $\varepsilon$ -turn Tug-of-War, est défini sur  $\Omega$  un espace métrique  $(X, d)$ . L'ensemble des arêtes  $E$  devient dans ce jeu  $E_\varepsilon$ , défini tel que  $x \sim y$  si et seulement si  $d(x, y) < \varepsilon$ . La fonction de valeur se nomme désormais  $u_\varepsilon$ , avec une fonction de gains terminaux  $F$  et une fonction de paiement d'étape  $\varepsilon^2 f$ . Le jeu se termine de la même façon que précédemment : quand  $x_k \in Y$ . Les gains du joueur I sont de :  $F(x) + \varepsilon^2 \sum_{i=0}^{k-1} f(x_i)$ .

Pour  $f = 0$  en tout point de  $X$ , [PSSW09] ont montré que, étant donné un point de départ  $x_0$ , la valeur continue du jeu  $u : \lim_{\varepsilon \rightarrow 0} u_\varepsilon$  pour les deux joueurs est la fonction infini-harmonique suivante :

$$u(x) = \frac{1}{2} \left[ \max_{y \in B_\varepsilon(x)} u(y) + \min_{y \in B_\varepsilon(x)} u(y) \right], \quad \forall x \in \Omega, \quad (3.41)$$

où  $B_\varepsilon(x) = \{y \mid |x - y| \leq \varepsilon\}$  et  $u$  étendant  $F$  sur  $\Omega$ .

Nous montrons maintenant que cette formulation coïncide avec le problème de Dirichlet associé au  $p$ -Laplacien normalisé sur graphe (3.12), avec  $p = \infty$ . Soit un graphe Euclidien  $G = (V, E, w_\varepsilon)$ , avec  $V = \Omega \subset \mathbb{R}^m$ ,  $E = \{(x, y) \in V \times V \mid w_\varepsilon(x, y) > 0\}$  et

$$w_\varepsilon(x, y) = \begin{cases} 1, & \text{si } |y - x| \leq \varepsilon \\ 0, & \text{sinon.} \end{cases} \quad (3.42)$$

On peut redéfinir les opérateurs max et min en utilisant la définition de la norme  $\mathcal{L}_\infty$  des gradients morphologiques (2.37) tels que :

$$\begin{aligned} \max_{y \in B_\varepsilon(x)} u(y) &= \|\nabla_{w_\varepsilon}^+(u)(x)\|_\infty + u(x) \\ \min_{y \in B_\varepsilon(x)} u(y) &= u(x) - \|\nabla_{w_\varepsilon}^-(u)(x)\|_\infty. \end{aligned} \quad (3.43)$$

En remplaçant max et min dans l'équation (3.41) par leurs formulations (3.43), on obtient :

$$\begin{aligned} u(x) &= \frac{1}{2} \left[ \|\nabla_{w_\varepsilon}^+(u)(x)\|_\infty(x) - \|\nabla_{w_\varepsilon}^-(u)(x)\|_\infty \right] + u(x) \\ \frac{1}{2} \left[ \|\nabla_{w_\varepsilon}^+(u)(x)\|_\infty(x) - \|\nabla_{w_\varepsilon}^-(u)(x)\|_\infty \right] &= 0, \end{aligned} \quad (3.44)$$

Ce qui coïncide avec la formulation (3.12) pour  $p = \infty$  :

$$\Delta_{w_\varepsilon, \infty}^N(u)(x) = 0. \quad (3.45)$$

### Tug-of-War non-local

Nous montrons maintenant que pour un graphe Euclidien général, toujours pour  $p = \infty$ , le  $p$ -Laplacien normalisé est lié au Tug-of-War non-local. Le jeu est similaire au jeu précédent, excepté le fait que  $B_\varepsilon$  est remplacé par un voisinage  $N(x_{k-1}) \subset \Omega$  défini tel que :

$$N(x_{k-1}) = \{x \in \Omega \mid w(x, x_{k-1}) > 0\} \cup \{x_{k-1}\} \quad (3.46)$$

Dans cette version du jeu, le jeton est déplacé en  $x_k$ , tel que  $x_k = x_k^I$  avec une probabilité  $P$  tel que :

$$P = \frac{\sqrt{w(x_{k-1}, x_k^I)}}{\sqrt{w(x_{k-1}, x_k^I)} + \sqrt{w(x_{k-1}, x_k^{II})}}, \quad (3.47)$$

et  $x_k = x_k^{II}$  avec une probabilité  $1 - P$ . La fonction de valeur du jeu pour les joueurs I et II satisfait maintenant la relation :

$$\begin{aligned} \max_{y \in N(x)} \sqrt{w(x, y)} (u(y) - u(x)) + \\ \min_{y \in N(x)} \sqrt{w(x, y)} (u(y) - u(x)) &= 0, \end{aligned} \quad (3.48)$$

qui peut se réécrire sous la forme :

$$\Delta_{w, \infty}^N(u)(x) = 0. \quad (3.49)$$

### Tug-of-War bruité

Si on modifie le jeu de la façon suivante : au tour  $k$ , les joueurs I et II jouent au point  $x_k \in \Omega$  au Tug-of-War précédemment décrit avec la probabilité  $\alpha$ , mais

ils jouent aussi une position aléatoire (toujours situé dans un rayon  $\varepsilon$  autour du point  $x_k$ ) avec la probabilité  $\beta$  (avec  $\alpha + \beta = 1$ ). La fonction du jeu satisfait maintenant l'équation suivante :

$$u(x) = \frac{\alpha}{2} \left[ \max_{y \in B_\varepsilon(x)} u(y) + \min_{y \in B_\varepsilon(x)} u(y) \right] + \frac{\beta}{|B_\varepsilon(x)|} \int_{B_\varepsilon(x)} u(y) dy, \quad (3.50)$$

avec  $u(x) = F(x) \forall x \in \partial\Omega$  et  $\alpha$  et  $\beta \in \mathbb{R}^+$ . Une preuve détaillée de l'existence et l'unicité de ce type de fonction est montrée dans [MPR12].

En utilisant la même construction de graphe que pour le Tug-of-War local ainsi que la même fonction de poids  $w_\varepsilon$ , l'équation (3.50) peut être réécrite dans le contexte des EdPs sur graphe : nous avons déjà montré que le terme de droite en  $\alpha$  de l'équation (3.50) peut se réécrire tel que (3.44). Le terme en  $\beta$  peut être réécrit tel que :

$$\frac{1}{|N(x)|} \int_{y \in N(x)} u(y) dy = \Delta_{w_\varepsilon, 2}^N(u)(x) + u(x). \quad (3.51)$$

On peut ainsi réécrire l'équation (3.50) telle que :

$$u(x) = \alpha(\Delta_{w_\varepsilon, \infty}^N(u)(x) + u(x)) + \beta(\Delta_{w_\varepsilon, 2}^N(u)(x) + u(x)), \quad (3.52)$$

ce qui est équivalent à :

$$\alpha \Delta_{w_\varepsilon, \infty}^N(u)(x) + \beta \Delta_{w_\varepsilon, 2}^N(u)(x) = 0. \quad (3.53)$$

On peut aisément voir que l'on retrouve le  $p$ -Laplacien normalisé sur graphe (3.12) avec  $2 \leq p \leq \infty$  :

$$\Delta_{w_\varepsilon, p}^N(u)(x) = 0. \quad (3.54)$$

## 3.4 Problème de Dirichlet associé

Dans cette section, nous nous intéressons au problème de Dirichlet associé à l'équation du  $p$ -Laplacien normalisé  $\Delta_{w, p}^N(f)(u) = 0$  et nous montrons que ce problème possède une solution unique.

### 3.4.1 Existence et unicité de la solution

**Théorème 1.** *Soient  $G = (V, E, w)$  un graphe connexe pondéré,  $g : V \rightarrow \mathbb{R}$  une fonction associant une valeur réelle à chaque sommet de  $G$  et  $A \subset V$*

l'ensemble des sommets pour lesquels la valeur de la fonction est inconnue. Il existe une fonction unique  $f \in \mathcal{H}(V)$  telle que  $f$  vérifie l'équation suivante :

$$\begin{cases} \Delta_{w,p}^N(f)(u) = 0 & u \in A \\ f(u) = g(u) & u \in \partial A \end{cases} \quad (3.55)$$

*Démonstration.* En utilisant l'opérateur  $NLA$  (3.14), on peut réécrire (3.55) telle que :

$$\begin{cases} NLA(f)(u) - f(u) = 0 & u \in A \\ f(u) = g(u) & u \in \partial A \end{cases} \quad (3.56)$$

Nous prouvons d'abord l'unicité de la solution en utilisant le principe de comparaison. Nous prouvons ensuite son existence en utilisant le théorème du point fixe de Brouwer.

### Unicité de la solution

Soient deux fonctions  $f$  et  $h$  prenant les mêmes valeurs sur  $\partial A$  et telles que  $f = NLA(f)$  et  $h = NLA(h)$  sur  $A$ . Dans ces conditions, nous pouvons appliquer le lemme 1 entre  $f$  et  $h$ , ce qui nous dit que l'inégalité  $f \leq h$  sur  $\partial A$  s'étend à  $A$ . En interchangeant le rôle de  $f$  et  $h$  dans ce lemme, nous obtenons l'inégalité opposée. Nous pouvons donc conclure que  $f = h$  sur  $A$ .

### Existence de la solution

Nous prouvons maintenant l'existence de la solution. Nous rappelons d'abord le théorème du point fixe de Brouwer : une fonction continue d'un sous ensemble convexe compact d'un espace Euclidien vers lui-même a un point fixe.

Nous identifions  $\mathcal{H}(V)$  comme  $\mathbb{R}^q$  et nous considérons l'ensemble  $K = \{f \in \mathcal{H}(V) \mid f(u) = g(u) \forall u \in \partial A \text{ et } m \leq f(u) \leq M \forall u \in A\}$ , où  $m = \min_{\partial A}(g(u))$  et  $M = \max_{\partial A}(g(u))$ . Par définition,  $K$  est un sous ensemble convexe compact de  $\mathbb{R}^n$ . Il est aisé de montrer que  $f \rightarrow NLA(f)$  est continue et prend de  $K$  vers  $K$ . En utilisant le théorème du point fixe de Brouwer, on peut déduire que  $NLA$  a un point fixe, qui est solution de  $NLA(f) = f$ .

□

### Lemme 1. Principe de comparaison

Soient deux fonctions  $f$  et  $h$ . Si  $f = NLA(f)$  et  $h = NLA(h)$  avec  $f \leq h$  sur  $\partial A$ , alors  $f \leq h$  sur le domaine  $V$ .



*Démonstration.* Nous procédons par l'absurde, en supposant qu'il existe un réel  $M$  tel que :

$$M = \max_V (f - h) > 0.$$

Soit  $B = \{u \in A : f(u) - h(u) = M\}$ . Par construction,  $B \neq \emptyset$  et  $B \cap \partial A = \emptyset$ . Nous supposons qu'il existe  $u_\lambda \in B$  et  $v_\lambda \in N(u_\lambda)$ , tel que  $v_\lambda \notin B$ . Sinon, si nous avons  $v \notin B$  pour chaque  $u \in A$  et pour chaque  $v \in N(u)$ , cela implique que  $B \cap \partial A \neq \emptyset$ , puisque le graphe est connexe : il y a contradiction. À partir de la définition de  $M$ , nous avons :

$$\begin{aligned} f(u_\lambda) - h(u_\lambda) &\geq f(u) - h(u) \quad \forall u \in N(u_\lambda) \\ h(u) - h(u_\lambda) &\geq f(u) - f(u_\lambda) \quad \forall u \in N(u_\lambda). \end{aligned}$$

En particulier, on peut écrire :

$$h(v_\lambda) - h(u_\lambda) > f(v_\lambda) - f(u_\lambda).$$

À partir de ces inégalités et en utilisant les définitions de  $NLE_\infty$  et  $NLD_\infty$ , nous avons :

$$\begin{aligned} &\max_{u \sim u_\lambda} \left( \sqrt{w(u_\lambda, u)} \max(h(u) - h(u_\lambda), 0) \right) \\ &\geq \\ &\max_{u \sim u_\lambda} \left( \sqrt{w(u_\lambda, u)} \max(f(u) - f(u_\lambda), 0) \right) \\ NLD_\infty(h)(u_\lambda) - h(u_\lambda) &\geq NLD_\infty(f)(u_\lambda) - f(u_\lambda) \end{aligned} \quad (3.57)$$

Similairement :

$$\begin{aligned} &\max_{u \sim u_\lambda} \left( \sqrt{w(u_\lambda, u)} \min(h(u) - h(u_\lambda), 0) \right) \\ &\geq \\ &\max_{u \sim u_\lambda} \left( \sqrt{w(u_\lambda, u)} \min(f(u) - f(u_\lambda), 0) \right) \\ h(u_\lambda) - NLE_\infty(h)(u_\lambda) &\geq f(u_\lambda) - NLE_\infty(f)(u_\lambda) \end{aligned} \quad (3.58)$$

En utilisant le même raisonnement avec l'opérateur  $NLMean$ , nous obtenons :

$$\frac{\sum_{u \sim u_\lambda} w(u_\lambda, u)h(u)}{\sum_{u \sim u_\lambda} w(u_\lambda, u)} - h(u_\lambda) \geq \frac{\sum_{u \sim u_\lambda} w(u_\lambda, u)f(u)}{\sum_{u \sim u_\lambda} w(u_\lambda, u)} - f(u_\lambda)$$

$$NLMean(h)(u_\lambda) - h(u_\lambda) > NLMean(f)(u_\lambda) - f(u_\lambda) \quad (3.59)$$

L'inégalité précédente est stricte puisqu'on sait qu'il existe un sommet  $v \in N(u_\lambda)$  tel que  $h(v) - h(u_\lambda) > f(v) - f(u_\lambda)$ .

Pour finir, en utilisant l'opérateur  $NLMedian$ , nous obtenons :

$$\begin{aligned} \sqrt{w(u_\lambda, u)}(h(u) - h(u_\lambda)) &\geq \sqrt{w(u_\lambda, u)}(f(u) - f(u_\lambda)) \\ \text{median}(\sqrt{w(u_\lambda, u)}(h(u) - h(u_\lambda))) &\geq \text{median}(\sqrt{w(u_\lambda, u)}(f(u) - f(u_\lambda))) \\ NLMedian(h)(u_\lambda) - h(u_\lambda) &\geq NLMedian(f)(u_\lambda) - f(u_\lambda) \end{aligned} \quad (3.60)$$

À partir des relations (3.57), (3.58), (3.59) et (3.60), nous pouvons écrire l'inégalité suivante pour  $2 \leq p \leq \infty$  :

$$\begin{aligned} \frac{p-2}{p} NLMidrange(h)(u_\lambda) + \frac{2}{p} NLMean(h)(u_\lambda) - h(u_\lambda) \\ > \\ \frac{p-2}{p} NLMidrange(h)(u_\lambda) + \frac{2}{p} NLMean(f)(u_\lambda) - f(u_\lambda). \end{aligned}$$

et pour  $1 \leq p \leq 2$  :

$$\begin{aligned} \frac{2-p}{p} NLMedian(h)(u_\lambda) + \frac{p-1}{p} NLMean(h)(u_\lambda) - h(u_\lambda) \\ > \\ \frac{2-p}{p} NLMedian(f)(u_\lambda) + \frac{p-1}{p} NLMean(f)(u_\lambda) - f(u_\lambda). \end{aligned}$$

Nous pouvons maintenant écrire, pour  $1 \leq p \leq \infty$  :

$$\begin{aligned} NLA(h)(u_\lambda) - h(u_\lambda) &> NLA(f)(u_\lambda) - f(u_\lambda) \\ h(u_\lambda) - h(u_\lambda) &> f(u_\lambda) - f(u_\lambda) \\ 0 &> 0 \end{aligned}$$

ce qui montre une contradiction et conclue la preuve du lemme 1. □

## 3.5 Équation de diffusion

Dans cette section, nous étudions l'équation de diffusion non-locale associée à l'opérateur  $\Delta_{w,p}^N$  que nous proposons. Nous montrons que la résolution de cette équation permet d'obtenir différents processus de filtrage itératif sur graphe en fonction du paramètre  $p$  et de retrouver des filtres classiquement utilisés en traitement d'image.

### 3.5.1 Équation

Soient un graphe  $G = (V, E, w)$  et une fonction  $f : V \times [0, T] \rightarrow \mathbb{R}$ . Nous considérons l'équation de diffusion suivante, pour  $1 \leq p \leq \infty$  :

$$\begin{cases} \frac{\partial f(u,t)}{\partial t} = \Delta_{w,p}^N(f)(u,t), \\ f(u,t=0) = f_0(u), \end{cases} \quad (3.61)$$

où  $f_0 : V \rightarrow \mathbb{R}$  est la valeur initiale de  $f$  au temps  $t = 0$ .

Afin de résoudre cette équation, nous utilisons simplement le schéma explicite d'Euler afin de discrétiser la dérivée temporelle de  $f$  :

$$\frac{\partial f(u,t)}{\partial t} = \frac{f^{n+1}(u) - f^n(u)}{\Delta t}, \quad (3.62)$$

où  $f^n(u) = f(u, n\Delta t)$ .

En utilisant ce schéma de discrétisation, on obtient le schéma de résolution itératif suivant :

$$\begin{cases} f^{n+1}(u) = f^n(u) + \Delta t \Delta_{w,p}^N(f^n)(u), \\ f^0(u) = f_0(u). \end{cases} \quad (3.63)$$

En utilisant l'opérateur  $NLA$  (3.14), on peut réécrire ce processus itératif tel que :

$$\begin{cases} f^{n+1}(u) = f^n(u) + \Delta t (NLA(f^n)(u) - f^n(u)), \\ f^0(u) = f_0(u). \end{cases} \quad (3.64)$$

En utilisant  $\Delta t = 1$ , ce processus devient :

$$\begin{cases} f^{n+1}(u) = NLA(f^n)(u), \\ f^0(u) = f_0(u). \end{cases} \quad (3.65)$$

En utilisant la définition de l'opérateur  $NLA$  (3.14), ce processus itératif permet de retrouver différents filtres, selon la valeur de  $p$  :

- Dans le cas où  $p = 1$ , le processus itératif (3.65) devient alors :

$$\begin{cases} f^{n+1}(u) = NLMedian(f^n)(u), \\ f^0(u) = f_0(u), \end{cases} \quad (3.66)$$

où  $NLMedian$  est l'opérateur défini à l'équation (3.10). Pour un graphe construit sur une image, en utilisant une 8-connexité et  $w_{uv} = 1$ , on peut remarquer qu'une itération de ce processus correspond au filtre median classique.

- Dans le cas où  $p = 2$ , le processus itératif (3.65) devient alors :

$$\begin{cases} f^{n+1}(u) = NLMean(f^n)(u), \\ f^0(u) = f_0(u), \end{cases} \quad (3.67)$$

où  $NLMean$  est l'opérateur défini à l'équation (2.62). On retrouve ici un filtre moyenneur classique, toujours dans le cas d'un graphe construit sur une image en utilisant une 8-connexité et  $w(u, v) = 1$ . On peut, par exemple, aussi retrouver le filtre des moyennes non-locales [BCM05], en utilisant une construction de graphe non-locale, *e.g.* un graphe des  $k$ -ppv, et une fonction de poids  $w_{uv}$  basée sur la similarité entre patches.

- Dans le cas où  $p = \infty$ , le processus itératif (3.65) devient alors :

$$\begin{cases} f^{n+1}(u) = NLMidrange(f^n)(u), \\ f^0(u) = f_0(u), \end{cases} \quad (3.68)$$

où  $NLMidrange$  est l'opérateur défini à l'équation (3.10). Toujours en utilisant un graphe construit sur une image en utilisant une 8-connexité et  $w(u, v) = 1$ , on retrouve un filtre mid-range classique (i.e. le filtre mid-range classique est la moyenne arithmétique des valeurs minimums et maximums d'un voisinage de pixels).

### 3.5.2 Propriétés

Dans cette section nous montrons des propriétés importantes du processus de filtrage (3.65). Pour ce faire, nous utilisons la version matricielle de ce processus afin de calculer une suite correspondant à des versions traitées du signal initial  $f_0$ . Nous montrons ensuite que ce processus satisfait le principe d'extremum, aussi appelé principe de minimum-maximum (PMM).

Soit une fonction donnée  $f : V \rightarrow \mathbb{R}$ , le processus itératif (3.65) appliqué à  $f$  peut s'écrire tel que :

$$F^{(n+1)} = \Phi_p(F^{(n)})F^{(n)}, \quad (3.69)$$

où  $F^{(n)} = (f^n(u))_{u \in V}^T \in \mathbb{R}^{|V|}$  et  $\Phi_p(F^{(n)})$  la matrice des coefficients dépendant de la fonction de poids, du paramètre  $p$  et du vecteur  $F$  à l'itération  $n$ .

En fonction du paramètre  $p$  cette matrice peut s'écrire :

- Pour  $p = 1$  :

$$\Phi_1(F^{(n)})_{uv} = \begin{cases} 1 - w(u, v_0), & \text{si } u = v \\ w(u, v_0), & \text{si } v \sim u \\ 0, & \text{sinon,} \end{cases} \quad (3.70)$$

où  $v_0 = \arg \operatorname{median}_{v \sim u}(\sqrt{w(u, v)}(f^n(v) - f^n(u)))$ .

- Pour  $p = 2$  :

$$\Phi_2(F^{(n)})_{uv} = \begin{cases} \frac{w(u, v)}{\sum_{z \sim u} w(u, z)}, & \text{si } u \sim v \\ 0, & \text{sinon.} \end{cases} \quad (3.71)$$

- Pour  $p = \infty$  :

$$\Phi_\infty(F^{(n)})_{uv} = \begin{cases} 1 - \frac{1}{2}(\sqrt{w(u, v_M)} + \sqrt{w(u, v_m)}), & \text{si } u = v \\ \frac{1}{2}\sqrt{w(u, v_M)}, & \text{si } v = v_M \\ \frac{1}{2}\sqrt{w(u, v_m)}, & \text{si } v = v_m \\ 0, & \text{sinon,} \end{cases} \quad (3.72)$$

où

$$\begin{aligned} v_M &= \arg \max_{v \sim u}(\sqrt{w(u, v)}(f^n(v) - f^n(u))^+) \quad \text{et} \\ v_m &= \arg \max_{v \sim u}(\sqrt{w(u, v)}(f^n(v) - f^n(u))^-). \end{aligned}$$

On peut montrer que ces trois matrices ont les propriétés suivantes :

- (P1) La somme de chaque ligne est égale à 1,
- (P2) Cette matrice est à coefficients positifs :  $\Phi_p(F^{(n)})_{uv} \geq 0, \forall (u, v) \in E, p \in \{1, 2, \infty\}$ ,
- (P3) La valeur moyenne de  $f$  n'est pas préservée.

### 3.5. Équation de diffusion

---

Nous pouvons maintenant définir la matrice générale  $\Phi_p$ . Pour  $1 \leq p \leq 2$ ,  $\Phi_p$  est définie telle que :

$$\Phi_p(F) = \alpha\Phi_1(F) + \beta\Phi_2(F), \quad (3.73)$$

où  $\alpha = \frac{2-p}{p}$  et  $\beta = \frac{2p-2}{p}$ .

Pour  $2 \leq p \leq \infty$ ,  $\Phi_p$  est définie telle que :

$$\Phi_p(F) = \alpha'\Phi_2(F) + \beta'\Phi_\infty(F), \quad (3.74)$$

où  $\alpha' = \frac{2}{p}$  et  $\beta' = \frac{p-2}{p}$ .

On peut remarquer ici que pour  $1 \leq p \leq 2$ , on a les propriétés suivantes pour  $\alpha$  et  $\beta$  :  $0 \leq \alpha \leq 1$ ,  $0 \leq \beta \leq 1$  et  $\alpha + \beta = 1$ .

Ces propriétés sont également vrai pour  $\alpha'$  et  $\beta'$  lorsque  $2 \leq p \leq \infty$ .

Pour  $1 \leq p \leq \infty$ , la matrice  $\Phi_p$  est donc définie comme une combinaison linéaire des matrices  $\Phi_1$ ,  $\Phi_2$  et  $\Phi_\infty$ . En utilisant les propriétés énoncées pour  $\alpha$ ,  $\beta$ ,  $\alpha'$  et  $\beta'$ , on peut en déduire que la matrice  $\Phi_p$  possède les mêmes propriétés que  $\Phi_1$ ,  $\Phi_2$  et  $\Phi_\infty$ .

Pour une fonction initiale  $f \in \mathcal{H}(V)$ , le processus de filtrage (3.69) génère une suite unique  $(F^{(n)})_{n \in \mathbb{N}}$ . Nous montrons maintenant que cette suite respecte un principe de maximum-minimum (PMM).

**Proposition 1.** *Le schéma itératif de filtrage (3.65) satisfait le PMM :*

*Soient  $f \in \mathcal{H}(V)$  une fonction initiale et  $(F^{(n)})_{n \in \mathbb{N}}$  la suite des fonctions filtrées en utilisant (3.69). On a alors :*

$$a \leq F_u^{(n)} \leq b \quad \forall u \in V, \forall n \in \mathbb{N}, \quad (3.75)$$

où

$$a := \min_{u \in V} f(u), \quad (3.76)$$

$$b := \max_{u \in V} f(u), \quad (3.77)$$

*Démonstration.* Cette propriété vient du fait que, pour tout  $u \in V$  et  $n \in \mathbb{N}$ , en utilisant les propriétés (P1) et (P2), on obtient les inégalités suivantes :

$$F_u^{(n+1)} = \sum_{v \in V} \Phi_p(F^{(n)})_{uv} F_v^{(n)} \stackrel{(P2)}{\leq} \max_{z \in V} F_z^{(n)} \sum_{v \in V} \Phi_p(F^{(n)})_{uv} \stackrel{(P1)}{=} \max_{z \in V} F_z^{(n)}, \quad (3.78)$$

et

$$F_u^{(n+1)} = \sum_{v \in V} \Phi_p(F^{(n)})_{uv} F_v^{(n)} \stackrel{(P2)}{\geq} \min_{z \in V} F_z^{(n)} \sum_{v \in V} \Phi_p(F^{(n)})_{uv} \stackrel{(P1)}{=} \min_{z \in V} F_z^{(n)}. \quad (3.79)$$

□

## 3.6 Expérimentations

Dans cette section, nous illustrons le comportement du processus de diffusion ainsi que le problème de Dirichlet impliquant le  $p$ -Laplacien normalisé sur graphe à travers différents problèmes inverses tels que le filtrage ou l'interpolation de fonction sur graphe. Notre but ici n'est pas de comparer notre approche aux méthodes de l'état de l'art mais d'illustrer le potentiel de cette formulation à travers différentes illustrations.

### 3.6.1 Filtrage et simplification d'image et de données

Dans cette section, nous illustrons le comportement du processus de diffusion impliquant le  $p$ -Laplacien normalisé sur graphe que nous proposons.

La figure 3.1 illustre le comportement du processus de diffusion sur une image, en faisant varier le paramètre  $p$  et en utilisant plusieurs types de construction de graphe et de fonctions de similarité. Les deux premières colonnes montrent le résultat de la diffusion pour 10 et 100 itérations, en utilisant une 8-connexité (noté Local dans la figure) et en utilisant une fonction de poids  $w_{uv} = 1$ . Pour la troisième colonne, nous avons utilisé la même construction de graphe, mais en utilisant une fonction de similarité Gaussienne ( $s_3$  à l'équation (1.17)) avec  $d = d_2$  (équation (1.11)) la distance Euclidienne dans l'espace des intensités de pixels. Pour les dernières colonnes, nous avons utilisé une construction de graphe non-locale, en utilisant un graphe des  $k$ -ppv avec  $k = 5$  dans l'espace des patches de l'image, en considérant les patches comme des vecteurs, comme défini à l'équation (1.23). La métrique utilisée ici pour la création du  $k$ -ppv est la distance Euclidienne. Nous avons pondéré le graphe en utilisant une similarité Gaussienne, toujours en utilisant la distance entre patches.

La figure 3.2 montre le processus de diffusion appliqué sur une base de données de chiffre manuscrit. Pour cette illustration, chaque imagerie de la base de données est représentée par un sommet du graphe. Les images sont considérées comme un vecteur de  $\mathbb{R}^q$  où  $q$  est le nombre de pixel de l'image, chaque élément du vecteur correspond à la valeur d'intensité du pixel

### 3.6. Expérimentations

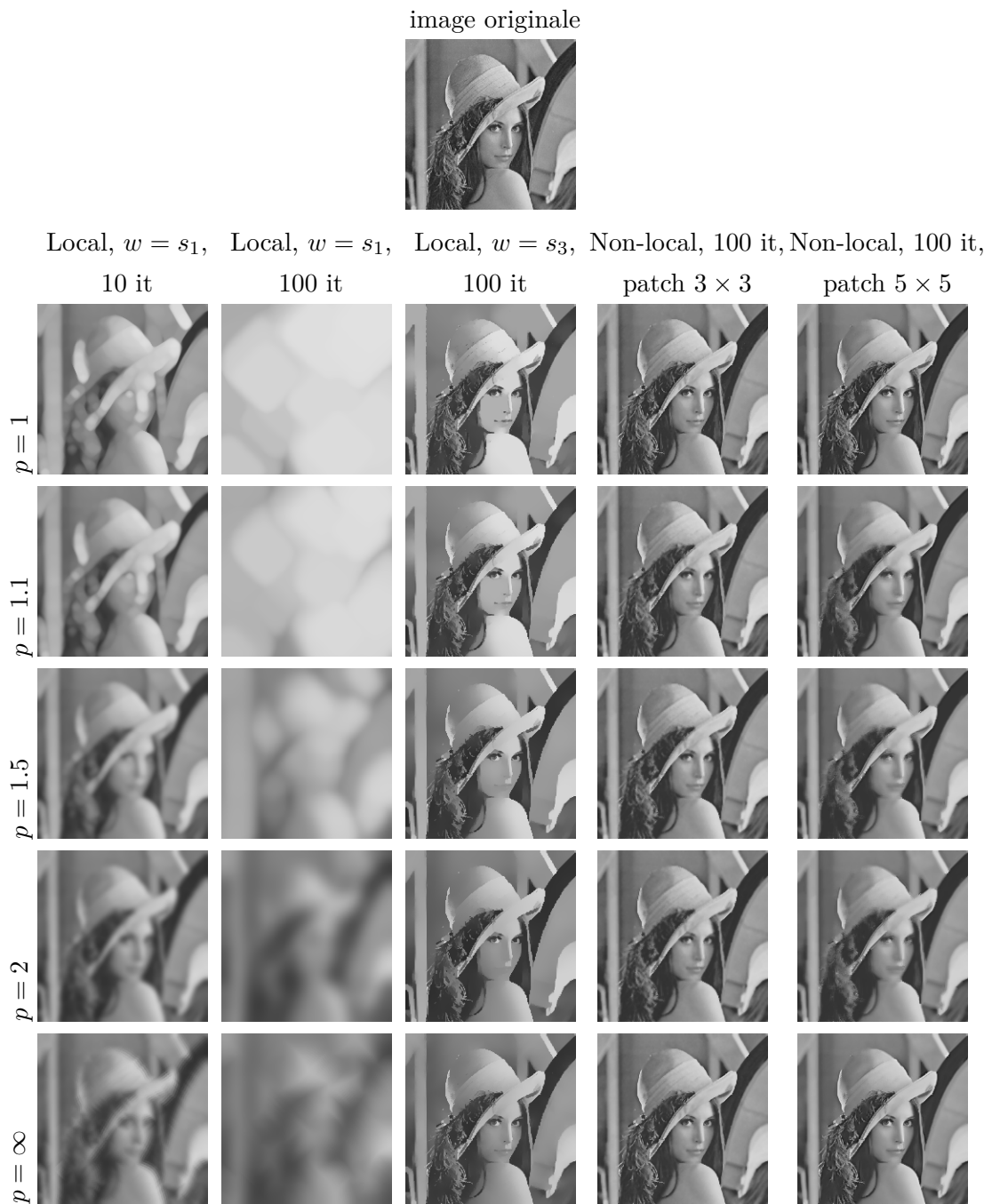


FIGURE 3.1 – Processus de diffusion appliqué sur une image, en faisant varier la paramètre  $p$ , la construction et la pondération du graphe. Voir le texte pour plus de détails.



correspondant. Pour cette expérimentation nous avons créé un graphe des  $k$ -ppv, en utilisant la métrique  $d_2$  entre les vecteurs correspondant aux imagettes et  $w_{uv}$  la similarité Gaussienne. On peut voir sur la figure 3.2 que ces différents filtres ont tendances à uniformiser les données.

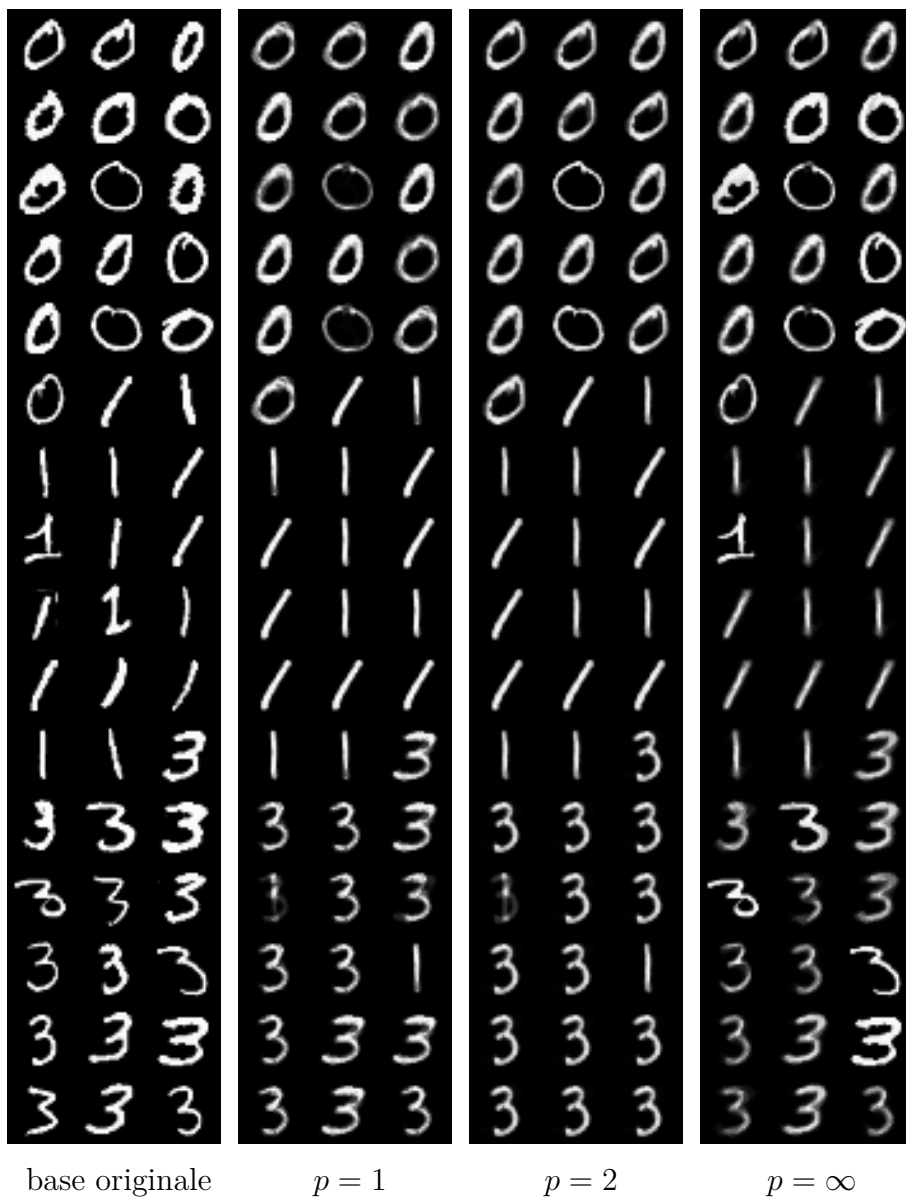


FIGURE 3.2 – Simplification de base de données en utilisant le processus de diffusion basé sur le  $p$ -Laplacien normalisé sur graphe, avec différentes valeurs pour le paramètre  $p$ . Voir le texte pour plus de détails.

### 3.6.2 Interpolation sur graphe

Différentes tâches en traitement d'image, vision par ordinateur et apprentissage peuvent être formulées comme un problème d'interpolation. La colorisation d'image et de vidéo, l'inpainting et la segmentation semi-supervisée sont des exemples de ces problèmes d'interpolation.

L'interpolation de données consiste à reconstruire de nouvelles valeurs pour des données manquantes en cohérence avec un ensemble de données connues. Nous proposons ici d'utiliser le  $p$ -Laplacien normalisé sur graphe comme un cadre unifié pour les tâches de segmentation semi-supervisée, de classification de données et d'inpainting. Nous considérons ces problèmes d'interpolations comme solution du problème de Dirichlet suivant :

$$\begin{cases} \Delta_{w,p}^N(f)(u) = 0 & u \in V_0 \\ f(u) = g(u) & u \in V - V_0, \end{cases} \quad (3.80)$$

où  $V_0 \subset V$  est le sous ensemble de sommets associé à l'information manquante. La valeur initiale de la fonction  $g$  dépend de l'application et sera définie pour chacune d'entre elles dans la suite.

Dans le cas de la segmentation semi-supervisée d'image, les approches basées sur les graphes sont devenues de plus en plus populaires. Plusieurs algorithmes basés sur les graphes pour la segmentation d'image ont été proposés, tels que les graph-cuts [BJ01], le random walker [Gra06], des algorithmes de plus courts chemins [BS09, FSdAL04], la ligne de partage des eaux [Ber05, CBNC09, VS91], ou des cadres unifiant certaines des méthodes précédentes (tel que le power watershed) [CGNT11, SG07]. Diverses approches populaires [BS09, CBNC09, DEL13, FSdAL04, Mey94] effectuent un partitionnement de graphe à partir d'un ensemble de germes donnés par l'utilisateur et une métrique. Le lecteur intéressé peut se référer à [DEL13] pour plus de détails.

Nous considérons ce problème comme un problème d'interpolation, où la fonction à interpoler est la fonction de label. En utilisant l'équation (3.80) et en considérant deux classes  $A$  et  $B$  ( $A \cup B$  est l'ensemble de germes initiaux donnés par l'utilisateur), la fonction de label initiale  $g$  est définie telle que :

$$\begin{cases} g(u) = -1, & \text{si } u \in A, \\ g(u) = 1, & \text{si } u \in B, \\ g(u) = 0, & \text{sinon.} \end{cases} \quad (3.81)$$

À convergence, l'appartenance d'un sommet à une classe peut être calculée simplement en seillant le signe de  $f$ .

*Remarque* : Dans le cas où il y a plus de deux classes ( $N$  classes), on peut effectuer une segmentation multi-classes en résolvant le système (3.80)  $N$  fois en considérant le marqueur  $A$  comme une classe et  $B$  comme toutes les autres classes. La fonction de marqueur  $L : V \rightarrow \{C_i\}_{i=1,\dots,N}$  associant à chaque sommet une classe est définie telle que :

$$L(u) = C_i | f_i(u) = \max_{j=1,\dots,N} f_j(u). \quad (3.82)$$

### Segmentation interactive d'image

Dans ce paragraphe, nous montrons le comportement du schéma d'interpolation dans le cadre de la segmentation d'image, ainsi que les avantages des schémas non-locaux comparés aux schémas locaux en utilisant différentes constructions de graphe.

La figure 3.3 illustre différents résultats de segmentation pour une image naturelle, avec deux constructions de graphes différentes (locale et non-locale), ainsi que différentes valeurs pour le paramètre  $p$ . Le graphe local est un graphe de 4-connexité où chaque pixel est caractérisé par sa couleur. Le graphe non-local est construit en utilisant les  $k$  plus proches voisins dans une fenêtre de voisinage de taille  $51 \times 51$  et chaque sommet est caractérisé par un patch couleur de taille  $5 \times 5$ .

### Classification de données

Nous illustrons dans ce paragraphe le comportement du  $p$ -Laplacien normalisé  $\Delta_{w,p}^N$  en utilisant l'équation (3.80) pour la classification de données. Nous avons utilisé ici un sous ensemble de 300 imagerie (représentant des zéros, uns et six) de la base de données USPS [Hul94]. Afin de simplifier la construction du graphe, nous considérons chaque imagerie comme un vecteur de taille 256 (le nombre de pixels de chaque imagerie). La métrique utilisée est la distance Euclidienne. Nous construisons un graphe des  $k$  plus proches voisins sur cet ensemble de données, où chaque sommet représente une imagerie. Nous avons initialisé un germe par classe. La figure 3.4 montre le graphe avec les germes initiaux et le résultat de la classification.

### 3.6.3 Inpainting non-local

L'inpainting est un problème fondamental en traitement d'image et est utilisé dans différents champs d'applications. On peut résumer l'inpainting comme la reconstruction d'une partie endommagée ou incomplète d'une image. Au cours

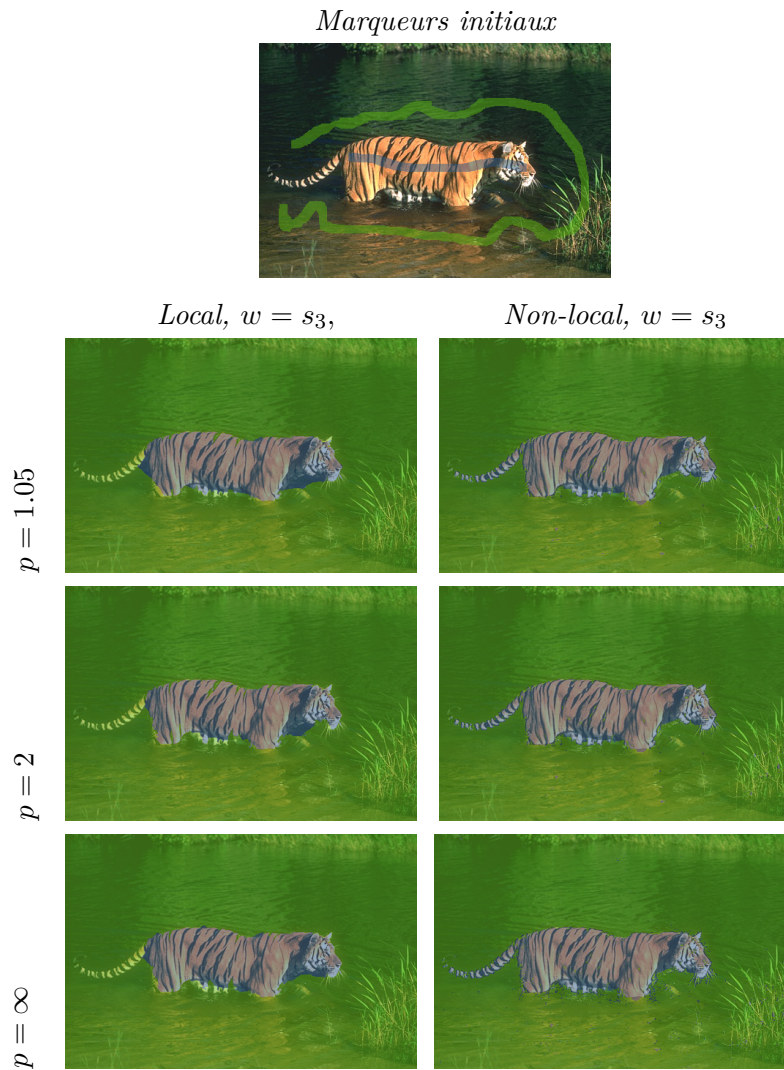


FIGURE 3.3 – Segmentation interactive d’image en utilisant  $\Delta_{w,p}^N$ . Voir le texte pour plus de détails.

des dernières années plusieurs méthodes ont été développées afin d’interpoler la géométrie, la texture, ou les deux simultanément. Parmi les méthodes qui ont été proposées, plusieurs sont basées sur les EDPs ou les méthodes variationnelles, voir [AFCS11, SB11] et les références citées. Depuis les travaux de Buades [BCM08] sur le filtrage non-local, beaucoup de méthodes non-locales pour l’inpainting d’image ont été développées. En effet ces méthodes ont une meilleure capacité à reconstruire des zones texturées que les méthodes purement locales.

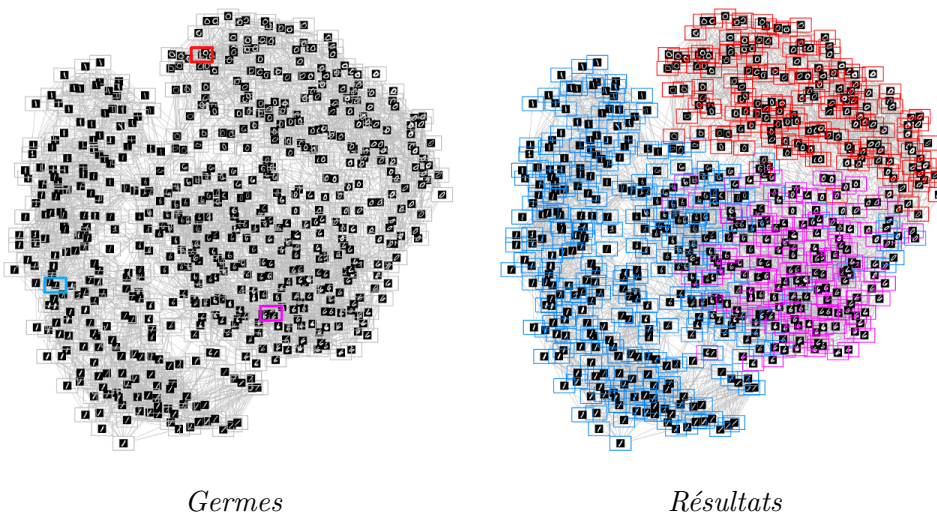


FIGURE 3.4 – Classification semi-supervisée de données. La figure de gauche montre un graphe construit sur un échantillon d’une base de données de chiffres manuscrits (0, 1 et 6) avec quelques germes initiaux dans chaque classe. La figure de droite montre le résultat de la classification à partir de ces germes.

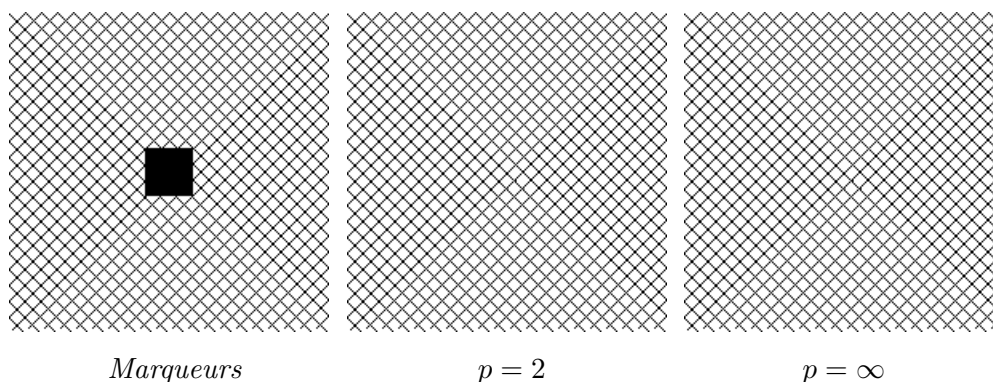


FIGURE 3.5 – Inpainting de texture. Voir le texte pour plus de détails.

Plusieurs travaux récents sont consacrés à unifier les approches d’interpolation locales et non-locales [GO07]. Nous pouvons citer [AFCS11] qui a présenté un cadre variationnel pour l’inpainting non-local d’image, ou encore les travaux [GEL11] qui a présenté un cadre de régularisation discrète non-local pour le traitement d’images et de variétés. Ce cadre a été utilisé pour présenter une approche qui unifie les méthodes géométriques locales et les méthodes non-locales basées motif pour l’inpainting vidéo non-local.

Le problème d’inpainting peut s’écrire comme le problème d’interpolation (3.80), avec  $V_0$  l’ensemble des pixels où l’information est manquante,  $g : V \rightarrow$

### 3.6. Expérimentations

---

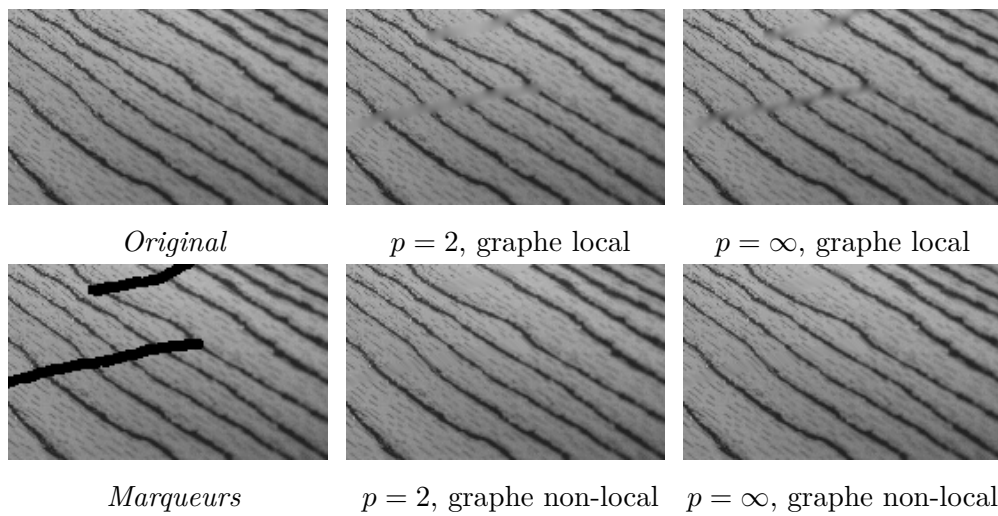


FIGURE 3.6 – Comparaison entre l’inpainting local et non-local, en utilisant une construction de graphe différente. Voir le texte pour plus de détails.

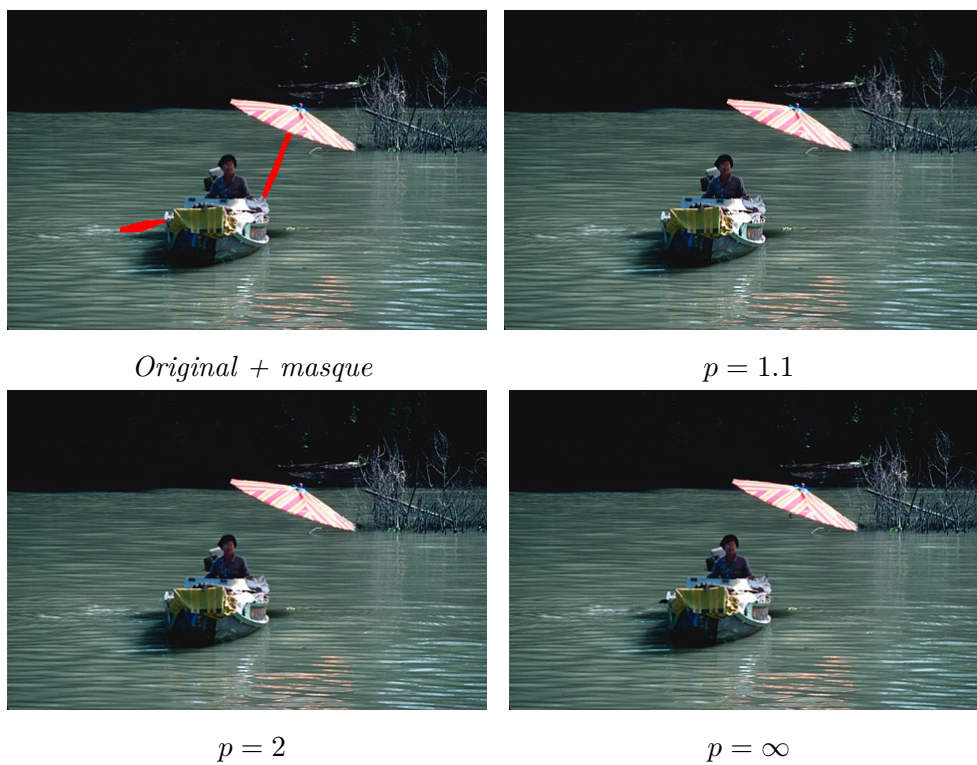


FIGURE 3.7 – Inpating d’images naturelles en utilisant une construction non-locale de graphe. Voir le texte pour plus de détails.



$\mathbb{R}^c$  représente l'information connue et  $f : V \rightarrow \mathbb{R}^c$  est l'image à reconstruire. Nous illustrons le comportement de cette méthode avec les figures 3.5, 3.6 et 3.7. La première (figure 3.5) montre le comportement de l'algorithme sur une image texturée, en utilisant une construction de graphe non-locale : nous utilisons ici un graphe des  $k$  plus proches voisins avec une fenêtre de voisinage de taille  $31 \times 31$  et des patches de tailles  $15 \times 15$ . La fonction de poids dépend de la similarité entre patches. La figure 3.6 compare le comportement de l'algorithme en utilisant deux constructions de graphes : locale et non-locale. Le graphe local est un graphe de 8-connexité, le graphe non-local est construit de la même manière que pour la figure 3.5. À la figure 3.7 nous illustrons le comportement sur une image naturelle, en utilisant la même construction de graphe non-locale que pour la figure 3.5 et en faisant varier le paramètre  $p$  de  $\Delta_{w,p}^N$ .

### 3.7 Conclusion

Dans ce chapitre, nous avons introduit une nouvelle classe de  $p$ -Laplacien normalisé. Nous l'avons présenté comme une adaptation discrète du  $p$ -Laplacien des jeux sur graphe pondéré. Cette classe d'opérateur est basée sur de nouveaux opérateurs différentiels interpolant entre le Laplacien normalisé, le 1-Laplacien et le Laplacien infini, sur graphe. Cet opérateur est aussi lié à des opérateurs statistiques non-locaux tels que les moyennes, médian et midrange non-locaux. Il généralise le  $p$ -Laplacien normalisé sur graphe pour  $1 \leq p \leq \infty$ . Nous avons aussi montré les liens entre notre opérateur et certaines EDPs locales et non-locales impliquant différents opérateurs de Laplace.

Le  $p$ -Laplacien normalisé étant lié avec différents jeux stochastiques de Tug-of-War, nous avons rappelé le principe des jeux à tour aléatoire, ainsi que celui du Tug-of-War et du Tug-of-War non-local afin de faire le lien entre ces jeux et l'opérateur que nous proposons. Comme beaucoup de problèmes de traitement d'images et de données sont formulés comme des problèmes d'interpolations, nous avons étudié le problème de Dirichlet associé à notre opérateur et prouvé l'existence et l'unicité de la solution à ce problème. Nous avons étudié l'équation de diffusion impliquant cet opérateur et montré des propriétés mathématiques importantes. Enfin nous avons illustré l'intérêt et le comportement de cet opérateur à travers différents problèmes inverses en traitement d'image et en classification.

# Chapitre 4

## $p$ -Laplacien et Laplacien infini avec termes de gradient

### Sommaire

---

<b>4.1</b>	<b>Introduction</b>	<b>82</b>
<b>4.2</b>	<b><math>p</math>-Laplacien non-local</b>	<b>83</b>
<b>4.3</b>	<b>Une nouvelle classe de <math>p</math>- et <math>\infty</math>- Laplacien sur graphe</b>	<b>84</b>
4.3.1	Définition	85
4.3.2	Connexion avec différents opérateurs différentiels	87
4.3.3	Relations avec certains jeux de Tug-of-War	90
<b>4.4</b>	<b>Équations aux différences partielles associées à l'opérateur proposé</b>	<b>91</b>
4.4.1	Équation de diffusion	92
4.4.2	Problème de Dirichlet	99
<b>4.5</b>	<b>Application aux problèmes inverses sur graphes pondérés</b>	<b>102</b>
4.5.1	Restauration et simplification d'image	102
4.5.2	Contours actifs	111
4.5.3	Interpolation	114
<b>4.6</b>	<b>Conclusion</b>	<b>119</b>

---



## Publications associées à ce chapitre

- [ETTar] Abderrahim ELMOATAZ, Matthieu TOUTAIN et Daniel TENBRINCK :  
On the  $p$ -Laplacian and  $\infty$ -Laplacian on graphs  
with Applications in Image and Data Processing. *SIAM Journal on  
Imaging Sciences (SIIMS)*, pages 1–37, to appear.
- [SET15] Ahcene SADI, Abderrahim ELMOATAZ et Matthieu TOUTAIN :  
Nonlocal pde morphology : a generalized shock operator on graph.  
*Signal, Image and Video Processing*, pages 1–8, 2015.

### 4.1 Introduction

Le but principal de ce chapitre est d'introduire une nouvelle classe de  $p$ -Laplacien et de Laplacien infini sur graphes avec termes de gradients, basée sur les opérateurs aux différences partielles. Une caractéristique intéressante de la classe d'opérateur que nous proposons est le fait que l'on puisse interpoler adaptativement entre des termes qui correspondent à des filtres de diffusion non-locale et des termes liés à la morphologie mathématique non-locale. Par conséquent, en utilisant cet opérateur nous pouvons combiner les avantages des deux formulations dans un même cadre. De plus, nous pouvons montrer que la nouvelle classe de  $p$ -Laplacien et Laplacien infini sur graphe que nous proposons permet de retrouver différents schémas de discrétisation existants, dans les cas locaux et non-locaux, mais aussi que nous pouvons retrouver des formulations d'opérateurs sur graphes introduites précédemment au chapitre 2. Par conséquent, l'opérateur que nous proposons est une formulation discrète unifiée du  $p$ -Laplacien et du Laplacien infini, du  $p$ -Laplacien des jeux (pour  $p = \infty$ ), du  $p$ -Laplacien non-local, ainsi que des gradients morphologiques, sur graphes pondérés, aussi bien pour des domaines discrets réguliers ou irréguliers.

Les principales contributions de ce chapitre sont les suivantes. D'abord, nous présentons le  $p$ -Laplacien non-local dans le domaine continu, afin de compléter le passage en revue des opérateurs continus que nous avons commencé au début du chapitre 3. Nous proposons ensuite une nouvelle classe de  $p$ -Laplacien et de Laplacien infini sur graphes unifiant différents schémas de discrétisation existants, locaux ou non-locaux. La formulation que nous proposons peut notamment être exprimée comme une interpolation convexe entre deux termes de gradients directionnels. Qui plus est, dans cette représentation les coefficients pondérant les termes de gradients peuvent être choisis dynamiquement. Ceci ouvre donc la voie à des filtrages adaptatifs appliquant des effets de type diffusion non-locale ou morphologique au gré des

caractéristiques des régions considérées. Par la suite, nous nous intéressons aux connexions avec des EDPs locales et non-locales, ainsi qu’avec le jeu du Tug-of-War. Comme au chapitre précédent, nous montrons que la formulation unifiée que nous proposons, en utilisant  $p = \infty$ , conduit à des EdPs qui coïncident avec les fonctions de valeurs de plusieurs Tug-of-War. Nous appliquons cette nouvelle classe de  $p$ -Laplacien et Laplacien infini à travers deux EdPs sur graphes pondérés liées à deux EDPs classiques dans le domaine continu et prouvons des propriétés mathématiques importantes, telles que l’existence et l’unicité de la solution. Nous étudions d’abord une famille d’EdPs paraboliques avec des conditions temporelles initiales. Nous montrons que cette famille conduit à une généralisation des filtres de diffusions et de choc sur des domaines discrets réguliers et irréguliers. Ensuite, nous étudions une famille d’EdPs elliptiques avec des conditions aux bords de Dirichlet, généralisant les processus d’interpolations sur les domaines discrets et nous prouvons l’existence et l’unicité de la solution. Enfin, nous illustrons comment cette nouvelle classe de  $p$ -Laplacien et Laplacien infini avec termes de gradients peut être appliquée à travers différents exemples, *i.e.* en segmentation, débruitage, inpainting et clustering. Pour souligner la généralité de la formulation proposée, nous testons nos algorithmes sur des données variées, telles que des images, des maillages 3D, ou encore des données non organisées telles que les nuages de points ou les bases de données.

## 4.2 $p$ -Laplacien non-local

L’intérêt pour le Laplacien fractionnel et le Laplacien non-local n’a cessé de croître ces dernières années. Ces opérateurs sont utilisés à travers des applications diverses, telles que : la mécanique des milieux continus, les phénomènes de transition de phase, la dynamique de population, le traitement d’image, ou encore la théorie des jeux, voir [AVMRTM10, AMRT08]. En traitement d’images, la régularisation basée sur le  $p$ -Laplacien non-local [GO07] est liée aux travaux sur les méthodes non-locales en traitement d’images, initialement proposées par Buades [BCM08]. Les méthodes de régularisation non-locales ont montré leur supériorité par rapport aux modèles classiques, notamment en filtrage, débruitant tout en préservant les structures géométriques et répétitives (telles que les textures) présentes dans l’image.

Soient  $\Omega \subset \mathbb{R}^m$  un domaine borné et  $f : \Omega \rightarrow \mathbb{R}$ . Pour  $1 \leq p < \infty$ , le  $p$ -Laplacien non-local est défini tel que :

$$\mathcal{L}_p u(x) = \int_{\Omega} \mu(x-y) |u(y) - u(x)|^{p-2} (u(y) - u(x)) \, dy, \quad 1 \leq p < \infty, \quad (4.1)$$

où  $\mu: \Omega \rightarrow \mathbb{R}$ . L'équation d'évolution impliquant cet opérateur a été étudiée dans [AMRT08] avec des conditions sur la fonction  $\mu$ . Les auteurs de [AMRT08] montrent aussi que cet opérateur est dérivé de l'équation d'Euler-Lagrange associée à l'énergie suivante :

$$J_p(u) = \frac{1}{2p} \int_{\Omega} \int_{\Omega} \mu(x-y) |u(y) - u(x)|^p dx dy . \quad (4.2)$$

On peut aussi remarquer que pour :

$$\mu(x) = \frac{1}{|x|^{ap}}, \quad (4.3)$$

où  $a = n/p + s$ ,  $0 < s < 1$  et  $p \geq 1$ , on retrouve la formulation du  $p$ -Laplacien fractionnel :

$$\mathcal{L}_p u(x) = \int_{\Omega} \frac{1}{|x-y|^{ap}} |u(y) - u(x)|^{p-2} (u(y) - u(x)) dy. \quad (4.4)$$

L'équation d'évolution de cet opérateur, avec différentes conditions aux bords (Neumann, Dirichlet, Cauchy), est étudiée dans [MRT].

En utilisant (4.3), mais cette fois pour  $p = \infty$ , on retrouve une formulation du Laplacien infini non-local, appelé Laplacien infini de Hölder [CLM12] :

$$\mathcal{L}_{\infty} u(x) = \max_{y \in \Omega, y \neq x} \left( \frac{u(y) - u(x)}{|y-x|^a} \right) + \min_{y \in \Omega, y \neq x} \left( \frac{u(y) - u(x)}{|y-x|^a} \right), \quad (4.5)$$

qui peut être formellement dérivé comme la limite quand  $p \rightarrow \infty$  de la minimisation de la famille d'énergie suivante :

$$J_p(u) = \int_{\Omega} \int_{\Omega} \frac{|u(y) - u(x)|^p}{|x-y|^{ap}} dx dy . \quad (4.6)$$

### 4.3 Une nouvelle classe de $p$ - et $\infty$ - Laplacien sur graphe

Dans cette section nous proposons un nouvel opérateur discret sur graphes pondérés qui correspond à une nouvelle classe de  $p$ -Laplacien avec termes de gradients. Nous montrons que l'opérateur proposé est un opérateur aux différences partielles unifiant les opérateurs sur graphes présentés au chapitre 2, tout en les étendant, à travers une formulation interpolant entre le  $p$ -Laplacien et les opérateurs morphologiques sur graphes. Nous commençons dans la section 4.3.1 par la définition de l'opérateur proposé et nous montrons que des cas

particuliers de cet opérateur permettent de retrouver soit le  $p$ -Laplacien, soit les opérateurs morphologiques précédemment définis. Nous étudions ensuite les liens entre cet opérateur et différents opérateurs différentiels locaux et non-locaux dans la section 4.3.2, ainsi que le lien avec les jeux de type Tug-of-War en section 4.3.3.

### 4.3.1 Définition

En utilisant la discrétisation du  $p$ -Laplacien et du Laplacien infini sur graphes, telle qu'introduite au chapitre 2, nous introduisons maintenant une nouvelle expression de ces deux opérateurs :

**Définition 8.** Les Laplaciens  $\mathcal{L}_{w,p}$  et  $\mathcal{L}_{w,\infty}$  sont définis pour une fonction  $f \in \mathcal{H}(V)$  par :

$$\mathcal{L}_{w,p}(f)(u) = \begin{cases} \alpha(u)\|\nabla_w^+(f)(u)\|_{p-1}^{p-1} - \beta(u)\|\nabla_w^-(f)(u)\|_{p-1}^{p-1}, & 2 \leq p < \infty, \\ \alpha(u)\|\nabla_w^+(f)(u)\|_\infty - \beta(u)\|\nabla_w^-(f)(u)\|_\infty, & p = \infty, \end{cases} \quad (4.7)$$

où  $\alpha, \beta : V \rightarrow [0, 1]$  et  $\alpha(u) + \beta(u) = 1$ ,  $\nabla_w^+$  et  $\nabla_w^-$  les gradients directionnels définis aux équations (2.34) et (2.35) à la section 2.2.2.

Avec une simple factorisation, ces opérateurs peuvent être réécrits tels que :

$$\begin{aligned} \mathcal{L}_{w,p}(f)(u) &= 2 \min(\alpha(u), \beta(u)) \Delta_{w,p}^a(f)(u) \\ &\quad + (\alpha(u) - \beta(u))^+ \|\nabla_w^+(f)(u)\|_{p-1}^{p-1} \\ &\quad - (\alpha(u) - \beta(u))^- \|\nabla_w^-(f)(u)\|_{p-1}^{p-1}, \quad 2 \leq p < \infty, \\ \mathcal{L}_{w,\infty}(f)(u) &= 2 \min(\alpha(u), \beta(u)) \Delta_{w,\infty}(f)(u) \\ &\quad + (\alpha(u) - \beta(u))^+ \|\nabla_w^+(f)(u)\|_\infty \\ &\quad - (\alpha(u) - \beta(u))^- \|\nabla_w^-(f)(u)\|_\infty, \quad p = \infty, \end{aligned} \quad (4.8)$$

où  $\Delta_{w,p}^a$  est le  $p$ -Laplacien anisotrope défini à l'équation (2.23) en section 2.1.5 et  $\Delta_{w,\infty}$  le Laplacien infini défini à l'équation (2.52) en section 2.2.3.

En considérant les fonctions  $\alpha$  et  $\beta$  constantes ( $\alpha(u) = \alpha$  et  $\beta(u) = \beta$ ), cette expression permet de retrouver différentes expressions du Laplacien, du Laplacien infini, ou du  $p$ -Laplacien sur graphes ainsi que les opérateurs avec termes de gradients, en fonction des valeurs de  $\alpha, \beta$  :

- Dans le cas  $\alpha = \beta \neq 0$ , l'opérateur (4.8) s'écrit :

$$\begin{aligned}\mathcal{L}_{w,p}(f)(u) &= \Delta_{w,p}^a(f)(u), \\ \mathcal{L}_{w,\infty}(f)(u) &= \Delta_{w,\infty}(f)(u),\end{aligned}\tag{4.9}$$

on retrouve ainsi l'expression des  $p$ -Laplacien et Laplacien infini.

- Dans le cas  $\alpha = 1$  on retrouve :

$$\begin{aligned}\mathcal{L}_{w,p}(f)(u) &= \|\nabla_w^+(f)(u)\|_{p-1}^{p-1}, \\ \mathcal{L}_{w,\infty}(f)(u) &= \|\nabla_w^+(f)(u)\|_\infty,\end{aligned}\tag{4.10}$$

- pour  $\beta = 1$  :

$$\begin{aligned}\mathcal{L}_{w,p}(f)(u) &= -\|\nabla_w^-(f)(u)\|_{p-1}^{p-1}, \\ \mathcal{L}_{w,\infty}(f)(u) &= -\|\nabla_w^-(f)(u)\|_\infty.\end{aligned}\tag{4.11}$$

On peut reconnaître ici les opérateurs aux différences partielles morphologiques utilisant la discrétisation directionnelle [TEL11] présentés en section 2.2.2.

- Dans le cas où  $\alpha - \beta > 0$ , l'opérateur (4.8) devient :

$$\begin{aligned}\mathcal{L}_{w,p}(f)(u) &= 2\beta(u)\Delta_{w,p}^a f(u) \\ &\quad + (\alpha(u) - \beta(u))\|\nabla_w^+(f)(u)\|_{p-1}^{p-1}, \\ \mathcal{L}_{w,\infty}(f)(u) &= 2\beta(u)\Delta_{w,\infty} f(u) \\ &\quad + (\alpha(u) - \beta(u))\|\nabla_w^+(f)(u)\|_\infty.\end{aligned}\tag{4.12}$$

- Inversement, dans le cas  $\alpha - \beta < 0$ , l'opérateur (4.8) devient :

$$\begin{aligned}\mathcal{L}_{w,p}(f)(u) &= 2\alpha(u)\Delta_{w,p}^a f(u) \\ &\quad - (\beta(u) - \alpha(u))\|\nabla_w^-(f)(u)\|_{p-1}^{p-1}, \\ \mathcal{L}_{w,\infty}(f)(u) &= 2\alpha(u)\Delta_{w,\infty} f(u) \\ &\quad - (\beta(u) - \alpha(u))\|\nabla_w^-(f)(u)\|_\infty.\end{aligned}\tag{4.13}$$

On peut remarquer que pour les deux derniers cas, en utilisant les opérateurs dans une EdP parabolique (*e.g.*,  $\partial_t f(u) = \mathcal{L}_{w,p}(f)(u)$ ), on obtient des filtres combinant une diffusion (avec le terme  $\Delta_{w,p}^a$ ) et un filtrage de choc (avec les termes de gradient).

### 4.3.2 Connexion avec différents opérateurs différentiels

Dans cette section, nous montrons que l'opérateur de  $p$ -Laplace avec termes de gradients que nous proposons (4.7) permet de retrouver des schémas de discrétisation classiques lorsqu'il est appliqué aux sommets d'un graphe défini sur une grille, mais aussi qu'il permet de retrouver des opérateurs non-locaux.

Pour cette section, nous considérons  $\Omega$  un domaine ouvert et borné de  $\mathbb{R}^m$  et  $f : \Omega \rightarrow \mathbb{R}$  une fonction donnée.

#### Discrétisation d'opérateurs locaux

Nous considérons d'abord le  $p$ -Laplacien anisotrope et nous montrons que la discrétisation utilisant les différences centrées du deuxième ordre permet de retrouver l'opérateur que nous proposons sur graphe. Ensuite, nous analysons le cas  $p = \infty$  et nous montrons que notre opérateur permet aussi de retrouver le Laplacien infini en utilisant la discrétisation d'Oberman. Enfin, nous montrons les liens entre notre opérateur et différentes discrétisations de normes de gradients.

**$p$ -Laplacien anisotrope :** Le  $p$ -Laplacien anisotrope peut s'exprimer par :

$$\Delta_p^a f = \sum_{i=1}^m \frac{\partial}{\partial x_i} \left[ \left| \frac{\partial f}{\partial x_i} \right|^{p-2} \frac{\partial f}{\partial x_i} \right]. \quad (4.14)$$

Comme au chapitre précédent, nous utilisons les différences centrées pour discrétiser cette expression :

$$\frac{\partial}{\partial x_i} f(x) \approx D_i(f)(x) = \frac{f(x + h_i/2) - f(x - h_i/2)}{h_i}. \quad (4.15)$$

On obtient la discrétisation suivante du  $p$ -Laplacien anisotrope :

$$\begin{aligned} \Delta_p^a f(x) = \sum_{i=1}^m \frac{1}{h_i^p} & \left( |f(x_i + h_i) - f(x_i)|^{p-2} (f(x_i + h_i) - f(x_i)) \right. \\ & \left. + |f(x_i - h_i) - f(x_i)|^{p-2} (f(x_i - h_i) - f(x_i)) \right). \end{aligned} \quad (4.16)$$

Nous montrons maintenant comment retrouver cette formulation en utilisant l'opérateur sur graphes que nous proposons. Pour ce faire, nous utilisons la construction de graphe sur une grille ( $G_g$ ) définie en section 3.3.3. Pour plus de fluidité dans la lecture, nous la rappelons ici.

Soit  $G = (V, E, w)$  un graphe pondéré représentant une grille à  $m$  dimensions. Soit  $u$  un sommet associé à un vecteur de coordonnées spatiales :  $u = (i_1 h_1, \dots, i_m h_m)^T$  où  $i_j \in \mathbb{N}$ ,  $h_j$  le pas spatial de la grille et  $j = 1, \dots, m$ . Le voisinage de  $u$  peut être défini tel que  $N_g(u) = \{v : v = u \pm h_j e_j\}_{j=1, \dots, m}$  où  $e_j = (q_k)_{k=1, \dots, m}^T$  est le vecteur tel que  $q_k = 1$  si  $j = k$  et  $q_k = 0$  sinon.

Nous considérons le cas où  $\alpha(u) = \beta(u) = \frac{1}{2}$  pour tout  $u \in V$  et  $2 \leq p < \infty$ . En identifiant les sommets du graphe par leur coordonnées spatiales et en utilisant la fonction de poids :

$$w_5(u, v_i) = \frac{1}{h_i^2}, \quad (4.17)$$

nous obtenons la formulation suivante :

$$\begin{aligned} \mathcal{L}_{w_5, p}(f)(u) &= \Delta_{w_5, p}^a(f)(u) = \sum_{v \sim u} \sqrt{w_5(u, v)}^p |f(v) - f(u)|^{p-2} (f(v) - f(u)) \\ &= \sum_{i=1}^m \frac{1}{h_i^p} (|f(v_i^+) - f(u)|^{p-2} (f(v_i^+) - f(u)) \\ &\quad + |f(v_i^-) - f(u)|^{p-2} (f(v_i^-) - f(u))) , \end{aligned} \quad (4.18)$$

où  $v_i^\pm = u \pm h_i e_i$ . On peut voir ici que l'on retrouve avec notre opérateur la formulation discrète du  $p$ -Laplacien anisotrope (4.16).

**Laplacien infini** : Soit  $f(x)$  une fonction lisse avec un gradient non nul en  $x$ . Pour le cas  $p = \infty$ , on peut retrouver la discrétisation d'Oberman du Laplacien infini [Obe13]. La démonstration est la même que pour le lien entre la discrétisation d'Oberman du Laplacien infini et le Laplacien infini défini au chapitre 3, nous ne la rappelons donc pas ici.

**Schéma de discrétisation de gradient** : En utilisant aussi la construction de graphe représentant une grille  $n$ -dimensionnelle ( $G_g$ ) ainsi que la fonction de poids  $w_5$ , on peut discrétiser la norme  $\mathcal{L}^2$  des gradients directionnels :

$$\begin{aligned} \|\nabla_{w_5}^+(f)(u)\|_2^2 &= \sum_{v \sim u} \sqrt{w_5(u, v)}^2 ((f(v) - f(u))^+)^2 \\ &= \sum_{i=1}^m \max(D_i^+(f)(u), 0)^2 + \min(D_i^-(f)(u), 0)^2 , \end{aligned} \quad (4.19)$$

où  $D_i^+$  et  $D_i^-$  désignent les différences finies avant et arrière :

$$D_i^+(f)(u) = \frac{f(v_i^+) - f(u)}{h_i}, \quad (4.20)$$

$$D_i^-(f)(u) = \frac{f(u) - f(v_i^-)}{h_i}. \quad (4.21)$$

On peut voir ici que l'écriture de (4.19) correspond au schéma de discrétisation directionnel de Osher-Sethian utilisé dans [OS88] pour  $p = 2$ . Pour le cas  $p = \infty$ , toujours en utilisant  $w_5$  et la représentation de la grille par un graphe, on obtient :

$$\begin{aligned} \|\nabla_{w_5}^+(f)(u, t)\|_\infty &= \max_{v \sim u} (\sqrt{w_5(u, v)}(f(v, t) - f(u, t))^+) \\ &= \max_{i=1, \dots, m} (D_i^+(f)(u, t), -D_i^-(f)(u, t), 0), \end{aligned} \quad (4.22)$$

ce qui correspond au schéma de discrétisation de Godunov pour  $|(\nabla f)(x)|_\infty$ .

### Approximation d'opérateurs différentiels non-locaux

Dans la suite, nous analysons les connections entre l'opérateur que nous proposons et la discrétisation d'opérateurs différentiels non-locaux en utilisant une construction non-locale de graphe et une fonction de poids adaptée.

Soit un graphe Euclidien  $G = (V, E, w_4)$ , avec  $V = \Omega \subset \mathbb{R}^m$ ,  $E = \{(x, y) \in V \times V \mid w_4(x, y) > 0\}$ ,  $2 \leq p < \infty$  et  $w_4$  la fonction de poids définie à l'équation (3.36), que nous rappelons ici :

$$w_4(x, y) = \begin{cases} \frac{1}{|y-x|^{2s}}, & \text{si } y \in \Omega \text{ et } y \neq x \\ 0, & \text{sinon.} \end{cases} \quad (4.23)$$

**$p$ -Laplacien fractionnel :** En utilisant l'opérateur proposé (4.7), on peut approximer le  $p$ -Laplacien fractionnel non-local (4.4), avec  $\alpha = \beta = \frac{1}{2}$ , tel que :

$$\begin{aligned} \mathcal{L}_{w_4, p}(f)(x) &= \int_{\Omega} \sqrt{w_4(x, y)}^p |f(y) - f(x)|^{p-2} (f(y) - f(x)) dy \\ &= \int_{\Omega} \frac{1}{|x - y|^{s \times p}} |f(y) - f(x)|^{p-2} (f(y) - f(x)) dy. \end{aligned} \quad (4.24)$$

**Remarque :** Pour  $w(x, y)$  quelconque, on peut retrouver le  $p$ -Laplacien anisotrope non-local (4.1).

#### Laplacien infini de Hölder :

Pour le cas  $\alpha = \beta$  et  $p = \infty$  nous retrouvons le même opérateur qu'au chapitre 3, ou le lien entre cet opérateur et le Laplacien infini de Hölder a déjà été effectué en section 3.3.3, nous ne le refaisons donc pas ici.



### 4.3.3 Relations avec certains jeux de Tug-of-War

Nous montrons dans cette section la relation entre l'opérateur que nous proposons et certains jeux de type Tug-of-War, comme présenté en section 3.3.4 du chapitre précédent. Dans le cas où  $\alpha = \beta = 0.5$  et  $p = \infty$ , l'opérateur que nous proposons (4.7) est le Laplacien infini sur graphe et est équivalent au  $p$ -Laplacien normalisé sur graphe (3.12), avec  $p = \infty$ . La démonstration effectuée à la section 3.3.4 du lien entre cet opérateur et les Tug-of-War et Tug-of-War non-locaux tient donc ici aussi. Pour cette raison nous ne la répétons pas dans cette section. Nous faisons le lien, pour  $p = \infty$ , entre notre opérateur et le Tug-of-War biaisé, tel que défini dans [PPS10].

#### Tug-of-War biaisé

Le principe du Tug-of-War ayant déjà été énoncé au chapitre précédent en section 3.3.4, nous ne le rappelons pas ici. Nous spécifions cependant le domaine  $\Omega \subset \mathbb{R}^m$  sur lequel le jeu se déroule ainsi que la fonction de valeur du jeu  $f : \Omega \rightarrow \mathbb{R}$  :

$$f^\varepsilon(x) = \frac{1}{2} \left[ \max_{y \in B_\varepsilon(x)} f^\varepsilon(y) + \min_{y \in B_\varepsilon(x)} f^\varepsilon(y) \right], \quad \forall x \in \Omega. \quad (4.25)$$

Dans [PSSW09] les auteurs ont montré que pour  $\varepsilon \rightarrow 0$ ,  $f^\varepsilon \rightarrow f$ , qui est une solution de l'EDP suivante :

$$\begin{cases} \Delta_\infty^G(f)(x) = 0, & x \in \Omega \\ f(x) = g(x), & x \in \partial\Omega, \end{cases} \quad (4.26)$$

avec  $g$  la fonction de gain terminal. Nous avons montré le lien entre le Tug-of-War et l'opérateur que nous proposons en utilisant cette EDP (4.26) en section 3.3.4.

Nous montrons maintenant le lien entre l'opérateur que nous proposons (4.7) et une version modifiée du jeu : le Tug-of-War biaisé. Soient  $\alpha > 0$  et  $\beta > 0$ . On peut ajouter un biais au Tug-of-War en utilisant les mêmes règles, en changeant les probabilités  $P_I$  et  $P_{II}$  que le joueur I ou le joueur II joue tel que  $P_I = \alpha$  et  $P_{II} = \beta$ . Dans la suite nous utiliserons  $x_k^I$  le déplacement du jeton par le joueur I au tour  $k$  et  $x_k^{II}$  le déplacement du jeton par le joueur II au tour  $k$ . Si les stratégies des joueurs sont optimales, la fonction de valeur de jeu correspondante devient :

$$\begin{cases} f^\varepsilon(x) = \alpha \max_{y \in B_\varepsilon(x)} f^\varepsilon(y) + \beta \min_{y \in B_\varepsilon(x)} f^\varepsilon(y), & x \in \Omega, \\ f(x) = g(x), & x \in \partial\Omega. \end{cases} \quad (4.27)$$

Cette fonction de valeur est liée au Laplacien infini avec termes de gradients :  $c|\nabla u| + \Delta_\infty u(x) = 0$ , avec  $c$  dépendant des valeurs de  $\alpha$  et  $\beta$  [PPS10]. Ce type d'EDP ainsi que le jeu stochastique correspondant ont été étudiés dans [PPS10].

Considérons maintenant ce jeu sur un graphe Euclidien général. La probabilité de déplacer le jeton en  $x_k^I$  est donnée par la formule suivante :

$$P_I = \frac{\alpha\sqrt{w(x_{k-1}, x_k^I)}}{\alpha\sqrt{w(x_{k-1}, x_k^I)} + \beta\sqrt{w(x_{k-1}, x_k^{II})}}, \quad (4.28)$$

la probabilité pour que  $x_k = x_k^{II}$  est de  $1 - P_I$ .

Nous obtenons la relation suivante pour ce jeu :

$$\begin{cases} \mathcal{L}_{w,\infty}(f)(x) = 0, & x \in \Omega, \\ f(x) = g(x), & x \in \partial\Omega. \end{cases} \quad (4.29)$$

## 4.4 Équations aux différences partielles associées à l'opérateur proposé

Récemment, différentes approches non-locales ont été développées pour le traitement d'image. Ces approches sont appelées non-locales du fait que chaque pixel de l'image peut interagir directement avec d'autres pixels dans le domaine de l'image, sans les restrictions locales habituelles de 4 ou 8 voisins. Les modèles non-locaux ont montré un grand avantage par rapport aux modèles traditionnels, puisque la régularité locale n'est pas requise avec ces approches. Ils ont aussi montré leur habilité à préserver les structures géométriques et répétitives présentes dans les images (telles que les textures). Dans des travaux précédents sur graphes, les auteurs de [BEM09] ont montré que la régularisation utilisant les  $p$ -Laplaciens sur graphe unifie les approches locales et non-locales pour les filtres locaux et non-locaux. Les auteurs de [TEL11] ont montré que la transcription des opérateurs morphologiques basée sur les EdPs sur graphes permet d'obtenir des filtres morphologiques non-locaux (érosion et dilatation).

Dans cette section, nous étudions un problème de diffusion non-locale basé sur l'opérateur que nous proposons (4.7). Nous montrons que la discrétisation temporelle de ce problème de diffusion unifie le filtrage basé sur le  $p$ -Laplacien et les filtres morphologiques. En particulier, le problème de diffusion nous permet de dériver de nouveaux filtres, combinant adaptativement les filtres de diffusion et de choc.

Nous étudions aussi une famille d'EdPs elliptiques avec des conditions aux bords de Dirichlet, basée sur l'opérateur que nous proposons. Nous montrons l'existence et l'unicité de la solution à ce problème. L'équation correspondant à ce problème est une généralisation des processus d'interpolation sur les domaines discrets.

#### 4.4.1 Équation de diffusion

Soient un graphe  $G = (V, E, w)$  et une fonction  $f : V \times [0, T] \rightarrow \mathbb{R}$ . Nous considérons l'équation de diffusion suivante pour  $2 \leq p \leq \infty$  :

$$\begin{cases} \frac{\partial f(u, t)}{\partial t} &= \mathcal{L}_{w, p}(f)(u, t) , \\ f(u, t = 0) &= f_0(u) , \end{cases} \quad (4.30)$$

où  $f_0 : V \rightarrow \mathbb{R}$  est la valeur initiale de  $f$  au temps  $t = 0$ .

Comme au chapitre précédent, nous discrétisons la dérivée de  $f$  par rapport à la variable de temps  $t$  en utilisant le schéma explicite d'Euler :

$$\frac{\partial f(u, t)}{\partial t} = \frac{f^{n+1}(u) - f^n(u)}{\Delta t} , \quad (4.31)$$

où  $f^n(u) = f(u, n\Delta t)$ .

#### Cas général

Nous commençons par étudier la cas général où  $0 \neq \alpha(u) \neq \beta(u) \neq 0$ . Nous proposons un algorithme itératif pour résoudre l'équation (4.30) et nous montrons différentes propriétés suivant le choix de  $p$ . Afin de résoudre l'équation (4.30), nous utilisons la discrétisation temporelle (4.31) afin d'obtenir le schéma itératif général suivant :

$$f^{n+1}(u) = f^n(u) + \Delta t \mathcal{L}_{w, p}(f^n)(u) . \quad (4.32)$$

- Dans le cas où  $2 \leq p < \infty$  nous obtenons :

$$\begin{aligned} f^{n+1}(u) = f^n(u) + \Delta t & \left[ \alpha(u) \|\nabla_w^+(f^n)(u)\|_{p-1}^{p-1} \right. \\ & \left. - \beta(u) \|\nabla_w^-(f^n)(u)\|_{p-1}^{p-1} \right]. \end{aligned} \quad (4.33)$$

En considérant la réécriture de l'opérateur (4.8), on peut voir que l'on retrouve un filtre moyennneur général non-symétrique, interpolant entre un processus itératif morphologique (local ou non-local, suivant

la configuration du graphe) et un processus de filtrage moyenné, en fonction des valeurs de  $\alpha(u)$  et  $\beta(u)$ . Nous pouvons réécrire (4.33) telle que :

$$f^{n+1}(u) = f^n(u) + \Delta t \left[ \alpha(u) \sum_{v \overset{+}{\sim} u} A_{u,v,p}(f^n)(f^n(v) - f^n(u)) - \beta(u) \sum_{v \overset{-}{\sim} u} B_{u,v,p}(f^n)(f^n(u) - f^n(v)) \right] \quad (4.34)$$

où  $A_{u,v,p}(f) = \sqrt{w(u,v)}^{p-1} (f(v) - f(u))^{p-2}$ ,  $B_{u,v,p}(f) = \sqrt{w(u,v)}^{p-1} (f(u) - f(v))^{p-2}$ ,  $v \overset{+}{\sim} u = \{v \sim u \mid f(v) > f(u)\}$  et  $v \overset{-}{\sim} u = \{v \sim u \mid f(v) < f(u)\}$ .

Tous les coefficients de la partie de droite sont positifs si

$$1 \geq \Delta t (\alpha(u) \sum_{v \overset{+}{\sim} u} A_{u,v,p}(f^n) + \beta(u) \sum_{v \overset{-}{\sim} u} B_{u,v,p}(f^n)) .$$

Cette inégalité correspond à la condition CFL pour le pas de temps  $\Delta t$ . Elle conduit aussi au principe de maximum (que nous montrons plus tard dans la section 4.4.1) pour l'approximation de l'équation (4.30) par ce schéma numérique.

Pour le graphe entier, nous obtenons :

$$1 \geq \Delta t \max_{u \in V} \left( \alpha(u) \sum_{v \overset{+}{\sim} u} A_{u,v,p}(f^n) + \beta(u) \sum_{v \overset{-}{\sim} u} B_{u,v,p}(f^n) \right) . \quad (4.35)$$

Par conséquent, en considérant la borne supérieure du terme de droite de (4.35), nous obtenons une condition suffisante :

$$1 \geq \Delta t \max_{u \in V} (|N(u)| \max_{v \overset{-}{\sim} u} (|f(v) - f(u)|^{p-2}) .$$

En se basant sur l'inégalité précédente, nous pouvons définir la valeur maximum  $\tau$  de  $\Delta t$  telle que :

$$\tau = \frac{1}{\max_{u \in V} (|N(u)| \max_{v \overset{-}{\sim} u} (|f(v) - f(u)|^{p-2})} . \quad (4.36)$$

Afin d'interpréter l'opérateur proposé plus facilement, nous introduisons les deux opérateurs suivants :

$$\begin{aligned} NLD_p(f)(u) &= f(u) + \tau \|\nabla_w^+(f)(u)\|_{p-1}^{p-1}, \\ NLE_p(f)(u) &= f(u) - \tau \|\nabla_w^-(f)(u)\|_{p-1}^{p-1}, \end{aligned} \quad (4.37)$$

où  $NLD_p, NLE_p : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$  représentent respectivement la dilatation et l'érosion non-locale. En considérant l'identité  $\|\nabla_w^+(f)(u)\| = \|\nabla_w^-(-f)(u)\|$ , nous obtenons les propriétés suivantes pour ces opérateurs :

$$\begin{aligned} NLD_p(-f) &= -NLE_p(f), \\ NLE_p(-f) &= -NLD_p(f). \end{aligned} \quad (4.38)$$

Nous pouvons maintenant définir une itération du schéma (4.34) telle que :

$$f^{n+1}(u) = NLA_p(f^n)(u), \quad (4.39)$$

où l'opérateur  $NLA_p : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$  (Non-Local Averaging) est défini tel que :

$$\begin{aligned} NLA_p(f)(u) &= \alpha(u)NLD_p(f)(u) + \beta(u)NLE_p(f)(u) \\ &= \alpha(u)\left(f(u) + \tau \|\nabla_w^+(f)(u)\|_{p-1}^{p-1}\right) \\ &\quad + \beta(u)\left(f(u) - \tau \|\nabla_w^-(f)(u)\|_{p-1}^{p-1}\right) \\ &= (\alpha(u) + \beta(u))f(u) \\ &\quad + \tau\alpha(u)\|\nabla_w^+(f)(u)\|_{p-1}^{p-1} - \tau\beta(u)\|\nabla_w^-(f)(u)\|_{p-1}^{p-1} \\ &= f(u) + \tau\left[\alpha(u)\|\nabla_w^+(f)(u)\|_{p-1}^{p-1} - \beta(u)\|\nabla_w^-(f)(u)\|_{p-1}^{p-1}\right], \end{aligned} \quad (4.40)$$

correspondant au schéma itératif (4.32).

- Dans le cas où  $p = \infty$  nous obtenons :

$$f^{n+1}(u) = f^n(u) + \Delta t \left[ \alpha(u) \|\nabla_w^+(f^n)(u)\|_\infty - \beta(u) \|\nabla_w^-(f^n)(u)\|_\infty \right]. \quad (4.41)$$

Comme pour le cas  $2 \leq p < \infty$  décrit plus tôt, nous obtenons la condition de stabilité du schéma suivante en un sommet  $u$  :

$$1 \geq \Delta t \left( \alpha(u) \sqrt{w(u, v_0)} + \beta(u) \sqrt{w(u, v_1)} \right),$$

avec

$$\begin{aligned} v_0 &= \arg \max_{v \overset{+}{\sim} u} (\sqrt{w(u, v)} f(v) - f(u)) \quad \text{et} \\ v_1 &= \arg \max_{v \overset{-}{\sim} u} (\sqrt{w(u, v)} f(u) - f(v)) . \end{aligned}$$

Puisque la valeur maximum de la fonction de poids  $w$  est égale à 1 et que nous avons la condition  $\alpha(u) + \beta(u) = 1$ , nous obtenons la condition de stabilité :  $\Delta t \leq 1$ .

Nous pouvons maintenant réécrire une itération de (4.41) telle que :

$$f^{n+1}(u) = NLA_\infty(f^n)(u) , \quad (4.42)$$

avec  $NLA_\infty : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$  défini tel que :

$$\begin{aligned} NLA_\infty(f)(u) &= \alpha(u)NLD_\infty(f)(u) + \beta(u)NLE_\infty(f)(u) \\ &= \alpha(u)(f(u) + \|\nabla_w^+(f)(u)\|_\infty) + \beta(u)(f(u) - \|\nabla_w^-(f)(u)\|_\infty) \\ &= f(u) + \alpha(u)\|\nabla_w^+(f)(u)\|_\infty - \beta(u)\|\nabla_w^-(f)(u)\|_\infty , \end{aligned} \quad (4.43)$$

où  $NLD_\infty$  et  $NLE_\infty$  sont les opérateurs définis aux équations (2.63) et (2.64), ce qui correspond à notre schéma itératif (4.41).

### Cas spéciaux

Nous montrons maintenant que, en fonction des valeurs de  $\alpha(u)$  et  $\beta(u)$ , certains cas du schéma itératif (4.33) permettent de retrouver des filtres de traitement d'images non-locaux déjà introduit dans les précédents travaux sur les EdPs sur graphes. En effet, en utilisant l'opérateur  $p$ -Laplacien avec termes de gradients, nous fournissons une interprétation des opérateurs morphologiques comme une famille de filtres moyenneurs digitaux non-locaux, qui peuvent être exprimés en utilisant le schéma itératif précédemment décrit.

- Pour le cas  $\alpha(u) = 0$  ou  $\beta(u) = 0$ , on retrouve la transcription des filtres morphologiques sur graphes présentés au chapitre 2 en section 2.3.2 :

$$\begin{cases} \frac{\partial f(u, t)}{\partial t} = \pm \|\nabla_w^\pm(f)(u, t)\|_{p-1}^{p-1} , & \text{pour } 2 \leq p < \infty , \\ \frac{\partial f(u, t)}{\partial t} = \pm \|\nabla_w^\pm(f)(u, t)\|_\infty , & \text{pour } p = \infty . \end{cases} \quad (4.44)$$

Pour  $\beta(u) = 0$  nous obtenons les EdPs suivantes, qui correspondent à la dilatation sur graphes :

$$\begin{cases} \frac{\partial f(u,t)}{\partial t} = \|\nabla_w^+(f)(u,t)\|_{p-1}^{p-1}, & \text{pour } 2 \leq p < \infty, \\ \frac{\partial f(u,t)}{\partial t} = \|\nabla_w^+(f)(u,t)\|_\infty, & \text{pour } p = \infty. \end{cases} \quad (4.45)$$

En exprimant ces EdPs en utilisant l'opérateur  $NLD_p$  (4.37), nous obtenons le schéma itératif suivant :

$$\begin{cases} f^{n+1}(u) = NLD_p(f^n)(u), & \text{pour } 2 \leq p < \infty, \\ f^{n+1}(u) = NLD_\infty(f^n)(u), & \text{pour } p = \infty. \end{cases} \quad (4.46)$$

Pour  $\alpha(u) = 0$  nous obtenons des EdPs correspondant à l'érosion non-locale discrète sur graphes :

$$\begin{cases} \frac{\partial f(u,t)}{\partial t} = -\|\nabla_w^-(f)(u,t)\|_{p-1}^{p-1}, & \text{pour } 2 \leq p < \infty, \\ \frac{\partial f(u,t)}{\partial t} = -\|\nabla_w^-(f)(u,t)\|_\infty, & \text{pour } p = \infty. \end{cases} \quad (4.47)$$

Similairement au cas  $\beta(u) = 0$ , nous pouvons exprimer ces EdPs en utilisant l'opérateur  $NLE_p$  (4.37) afin d'obtenir le schéma itératif suivant :

$$\begin{cases} f^{n+1}(u) = NLE_p(f^n)(u), & \text{pour } 2 \leq p < \infty, \\ f^{n+1}(u) = NLE_\infty(f^n)(u), & \text{pour } p = \infty. \end{cases} \quad (4.48)$$

**Remarque :** Dans le cas où  $w(u,v) = 1, \forall (u,v) \in E$  et  $p = \infty$ , on retrouve les opérateurs morphologiques discrets sur une grille régulière [BM92] - érosion pour  $\alpha = 0$  et dilatation pour  $\beta = 0$ .

- Pour le cas  $\alpha(u) = \beta(u)$  nous pouvons exprimer (4.30) telle que :

$$\begin{cases} \frac{\partial f(u,t)}{\partial t} = \Delta_{w,p}^a(f)(u), & \text{pour } 2 \leq p < \infty, \\ \frac{\partial f(u,t)}{\partial t} = \Delta_{w,\infty}(f)(u), & \text{pour } p = \infty. \end{cases} \quad (4.49)$$

En utilisant le schéma itératif (4.39) avec  $2 \leq p < \infty$  nous obtenons :

$$f^{n+1}(u) = NLM_p(f^n)(u), \quad (4.50)$$

où l'opérateur  $NLM_p : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$  est défini tel que :

$$\begin{aligned}
 NLM_p(f)(u) &= \frac{1}{2}(NLD_p(f)(u) + NLE_p(f)) \\
 &= f(u) + \frac{\tau}{2}(\|\nabla_w^+(f)(u)\|_{p-1}^{p-1} - \|\nabla_w^-(f)(u)\|_{p-1}^{p-1}) \quad (4.51) \\
 &= f(u) + \tau\Delta_{w,p}^a f(u) ,
 \end{aligned}$$

Pour le cas  $p = \infty$  nous obtenons le schéma itératif suivant :

$$f^{n+1}(u) = NLM_\infty(f^n)(u) , \quad (4.52)$$

où l'opérateur  $NLM_\infty : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$  est défini tel que :

$$\begin{aligned}
 NLM_\infty(f)(u) &= \frac{1}{2}(NLD_\infty(f)(u) + NLE_\infty(f)) \\
 &= f(u) + \frac{1}{2}(\|\nabla_w^+(f)(u)\|_\infty - \|\nabla_w^-(f)(u)\|_\infty) \quad (4.53) \\
 &= f(u) + \Delta_{w,\infty}(f)(u) .
 \end{aligned}$$

On remarque ici que cet opérateur est défini de la même manière que l'opérateur  $NLMidrange$  au chapitre précédent, à l'équation (3.10) en section 3.3.1.

### Propriétés

Dans cette section nous montrons différentes propriétés importantes du processus de filtrage (4.34) introduit plus tôt en réécrivant les opérateurs sous forme de matrices. Ensuite, nous montrons que le schéma itératif satisfait le principe de minimum-maximum (PMM). Enfin, nous montrons que la convergence de ce processus de diffusion vers une fonction  $f^*$  entraîne  $\mathcal{L}_{w,p}(f^*)(u) = 0$ .

Soit une fonction donnée  $f : V \rightarrow \mathbb{R}$ , le processus itératif (4.39) appliqué à  $f$  peut s'écrire tel que :

$$F^{n+1} = \Phi(F^n)F^n, \quad (4.54)$$



où  $F^n = \left( f^n(u) \right)_{u \in V}^T \in \mathbb{R}^{|V|}$  et  $\Phi(F^n)$  la matrice des coefficients en ce qui concerne la fonction de poids, les paramètres  $\alpha$  et  $\beta$ , le paramètre  $p$  et le vecteur  $F$  à l'itération  $n$ . Cette matrice peut s'écrire :

$$\Phi(F^n)(u, v) = \begin{cases} 1 - \tau \left( \alpha(u) \sum_{z \overset{+}{\sim} u} A_{u,z,p}(f^n) + \beta(u) \sum_{z \overset{-}{\sim} u} B_{u,z,p}(f^n) \right), & \text{si } u = v \\ \tau \alpha(u) A_{u,v,p}(f^n), & \text{si } v \overset{+}{\sim} u \\ \tau \beta(u) B_{u,v,p}(f^n), & \text{si } v \overset{-}{\sim} u \\ 0, & \text{sinon.} \end{cases} \quad (4.55)$$

On peut montrer que la matrice  $\Phi$  a les propriétés suivantes :

- La somme de chaque ligne de  $\Phi$  est égale à 1,
- Cette matrice est à coefficients positifs :  $\Phi(F^n)(u, v) \geq 0, \forall (u, v) \in E$ ,
- À moins que  $\alpha(u) = \beta(u)$  pour tout  $u \in V$ , la valeur moyenne de  $f$  n'est pas préservée.

**Proposition 2.** *Le schéma itératif de filtrage (4.39) satisfait le PMM.*

*Démonstration.* Soient  $m = \min_{u \in V} (f^0(u))$  et  $M = \max_{u \in V} (f^0(u))$ . Par définition, nous savons que les opérateurs non-locaux  $NLE_p$  et  $NLD_p$  (eq. (4.37)) satisfont pour tout  $u \in V$  :

$$\begin{aligned} m &\leq NLE_p(f^0)(u) \leq M, \\ m &\leq NLD_p(f^0)(u) \leq M. \end{aligned} \quad (4.56)$$

À partir de ces inégalités et en rappelant que les paramètres  $\alpha(u) + \beta(u) = 1$ , nous pouvons écrire, pour tout  $u \in V$  :

$$m \leq \alpha(u) NLE_p(f^0)(u) + \beta(u) NLD_p(f^0)(u) \leq M, \quad (4.57)$$

et ainsi

$$m \leq f^1(u) = NLA(f^0)(u) \leq M. \quad (4.58)$$

Enfin, par récurrence cette relation peut être étendue aux itérations suivantes.

□

Nous pouvons conclure que le schéma itératif (4.39) est stable et correspond à un processus de filtrage non-local combinant des opérations de dilatation, d'érosion et de moyenneur non-local.

Soient un graphe  $G = (V, E, w)$  et une fonction  $f_0 \in \mathcal{H}(V)$ , nous pouvons écrire un processus de filtrage simple en utilisant l'algorithme suivant :

1. L'algorithme est initialisé avec  $f^0 = f_0$ .
2. Pour tous les  $k = 1, \dots, |V|$  faire  $f^{k+1}(u_k) = NLA_p(f^k)(u_k)$ .

**Proposition 3.** *Si le processus de filtrage itératif converge vers une fonction  $f^*$ , alors  $f^*$  satisfait  $\mathcal{L}_{w,p}(f^*)(u) = 0, \forall u \in V$ .*

*Démonstration.* Soit  $f^*$  la limite du schéma itératif (4.39). Nous avons alors :

$$\begin{aligned} f^*(u) &= NLA(f^*)(u) \\ &= f^*(u) + \tau[\alpha(u)\|\nabla_w^+(f^*)(u)\|_{p-1}^{p-1} - \beta(u)\|\nabla_w^-(f^*)(u)\|_{p-1}^{p-1}] \quad (4.59) \\ \Rightarrow 0 &= \tau\mathcal{L}_{w,p}(f^*)(u) \end{aligned}$$

Comme les graphes que nous considérons ont un ensemble fini de sommets et d'arêtes et  $\tau > 0$ , on peut directement déduire que  $\mathcal{L}_{w,p}(f^*)(u) = 0$ . □

#### 4.4.2 Problème de Dirichlet

Dans cette section nous analysons le problème de Dirichlet associé au  $p$ -Laplacien avec termes de gradients sur graphe  $\mathcal{L}_{w,p}$  et montrons que ce problème a une solution unique. D'une manière générale, le problème de Dirichlet est un problème aux limites du type suivant : trouver une fonction  $f \in \mathcal{H}(V)$  telle que  $\mathcal{L}_{w,p}(f) = 0$  sur un ensemble  $A \subset V$ , en connaissant la valeur de  $f$  aux frontières du domaine  $A$  ( $\partial A$ ).

Soient un graphe pondéré  $G = (V, E, w)$ , un ensemble de sommets  $A \subset V$  et  $g : \partial A \rightarrow \mathbb{R}$  une fonction définie aux frontières de  $A$ . Nous considérons l'équation suivante décrivant le problème de Dirichlet associé à l'opérateur  $\mathcal{L}_{w,p}$  :

$$\begin{cases} \mathcal{L}_{w,p}(f)(u) = 0, & \text{pour } u \in A, \\ f(u) = g(u), & \text{pour } u \in \partial A. \end{cases} \quad (4.60)$$

De nombreux problèmes en traitement d'image et en apprentissage peuvent être formulés de la sorte. Dans cette partie nous étudierons seulement le cas  $2 \leq p < \infty$ , le cas  $p = \infty$  ayant déjà été étudié dans [EDLL14].

### Étude de l'existence et unicité de la solution

**Théorème 2.** Soient un graphe connexe  $G = (V, E, w)$ , un ensemble  $A \subset V$  et une fonction  $g : \partial A \rightarrow \mathbb{R}$ . Il existe une fonction unique  $f \in \mathcal{H}(V)$  telle que  $f$  vérifie l'équation suivante :

$$\begin{cases} \alpha(u) \|\nabla_w^+(f)(u)\|_{p-1}^{p-1} - \beta(u) \|\nabla_w^-(f)(u)\|_{p-1}^{p-1} = 0, & \text{pour } u \in A, \\ f(u) = g(u), & \text{pour } u \in \partial A. \end{cases} \quad (4.61)$$

De plus la solution respecte le principe de maximum-minimum, selon lequel :

$$m \leq f(u) \leq M, \quad (4.62)$$

où  $m = \min_{\partial A}(g(u))$  et  $M = \max_{\partial A}(g(u))$ .

*Démonstration.* En premier lieu, nous remarquons que l'équation (4.61) peut être réécrite telle que  $f(u) = NLA_p(f)(u)$ . Nous prouvons d'abord l'unicité de la solution en utilisant le principe de comparaison. Nous prouvons ensuite son existence en utilisant le théorème du point fixe de Brouwer.

### Unicité de la solution

Soient deux fonctions  $f$  et  $h$  prenant les mêmes valeurs sur  $\partial A$  et telles que  $f = NLA_p(f)$  et  $h = NLA_p(h)$  sur  $A$ . Dans ces conditions, nous pouvons appliquer le lemme 2 entre  $f$  et  $h$ , ce qui nous dit que l'inégalité  $f \leq h$  sur  $\partial A$  s'étend à  $A$ . En interchangeant le rôle de  $f$  et  $h$  dans ce lemme, nous obtenons l'inégalité opposée. Nous pouvons donc conclure que  $f = h$  sur  $A$ .

### Existence de la solution et principe de maximum-minimum

Pour la preuve d'existence des solutions de l'équation (4.61) nous rappelons le théorème du point fixe de Brouwer. Il affirme qu'une fonction continue définie dans un sous ensemble convexe compact d'un espace Euclidien à valeur dans ce même compact a un point fixe. Nous identifions  $\mathcal{H}(V)$  comme  $\mathbb{R}^q$  et considérons l'ensemble  $K = \{f \in \mathcal{H}(V) \mid f(u) = g(u) \forall u \in \partial A, \text{ et } m \leq f(u) \leq M \forall u \in A\}$ , où  $m = \min_{\partial A}(g(u))$  et  $M = \max_{\partial A}(g(u))$ . Par définition,  $K$  est un sous ensemble convexe compact de  $\mathbb{R}^q$ . Il est aisé de montrer que  $f \rightarrow NLA_p(f)$  est continue et envoie  $K$  sur lui-même. En utilisant le théorème du point fixe de Brouwer, on peut déduire que  $NLA_p$  a un point fixe, qui est solution de

#### 4.4. Équations aux différences partielles associées à l'opérateur proposé

$NLA_p(f) = f$  et qui vérifie le principe de maximum-minimum, comme toutes les fonctions de  $K$ .

□

### Lemme 2. Principe de comparaison

Soient deux fonctions  $f$  et  $h$  telles que  $f = NLA_p(f)$  et  $h = NLA_p(h)$  où  $f \leq h$  sur  $\partial A$ , alors,  $f \leq h$  sur le domaine  $A$ .

*Démonstration.* Par contradiction, nous faisons l'hypothèse qu'il existe  $M \in \mathbb{R}$  tel que :

$$M = \max_V(f - h) > 0.$$

Soit  $B = \{u \in A : f(u) - h(u) = M\}$ . Par construction  $B \neq \emptyset$  et  $B \cap \partial A = \emptyset$ . Il existe un sommet  $u_0 \in B$  le plus proche de  $\partial A$  et un sommet  $v$  tel que  $v \notin B$ .

À partir de la définition de  $M$  et de  $v$  nous obtenons :

$$\begin{cases} f(u_0) - h(u_0) \geq f(u) - h(u) & \forall u \in N(u_0), \\ h(u) - h(u_0) \geq f(u) - f(u_0) & \forall u \in N(u_0), \\ h(u) - h(u_0) > f(u) - f(u_0) & \text{si } u = v. \end{cases}$$

À partir de ces inégalités, nous pouvons déduire :

$$\begin{aligned} \max_{u \sim u_0}(h(u) - h(u_0), 0) &\geq \max_{u \sim u_0}(f(u) - f(u_0), 0) \\ (\sqrt{w(u_0, u)} \max_{u \sim u_0}(h(u) - h(u_0), 0))^p &\geq (\sqrt{w(u_0, u)} \max_{u \sim u_0}(f(u) - f(u_0), 0))^p \\ \alpha(u_0) \sum_{u \sim u_0} (\sqrt{w(u_0, u)} \max_{u \sim u_0}(h(u) - h(u_0), 0))^p &\geq \alpha(u_0) \sum_{u \sim u_0} (\sqrt{w(u_0, u)} \max_{u \sim u_0}(f(u) - f(u_0), 0))^p \\ \alpha(u_0) \|\nabla_w^+(h)(u_0)\|_p^p &\geq \alpha(u_0) \|\nabla_w^+(f)(u_0)\|_p^p \end{aligned} \tag{4.63}$$

Cette inégalité devient stricte si  $f(v) \geq f(u_0)$ .

De même, on peut obtenir l'inégalité suivante :

$$\begin{aligned}
 h(u_0) - h(u) &\leq f(u_0) - f(u) \\
 \max(h(u) - h(u_0), 0) &\geq \max(f(u) - f(u_0), 0) \\
 \sum_{u \sim u_0} (\sqrt{w(u_0, u)} \max(h(u_0) - h(u), 0))^p &\leq \sum_{u \sim u_0} (\sqrt{w(u_0, u)} \max(f(u_0) - f(u), 0))^p \\
 \|\nabla_w^-(h)(u_0)\|_p^p &\leq \|\nabla_w^-(f)(u_0)\|_p^p \\
 -\beta(u_0) \|\nabla_w^-(h)(u_0)\|_p^p &\geq -\beta(u_0) \|\nabla_w^-(f)(u_0)\|_p^p
 \end{aligned} \tag{4.64}$$

Cette inégalité devient stricte si  $f(v) \leq f(u_0)$ .

L'une des deux inégalités précédente étant stricte, nous pouvons déduire l'inégalité suivante :

$$\begin{aligned}
 \alpha(u_0) \|\nabla_w^+(h)(u_0)\|_p^p - \beta(u_0) \|\nabla_w^-(h)(u_0)\|_p^p &> \alpha(u_0) \|\nabla_w^+(f)(u_0)\|_p^p - \beta(u_0) \|\nabla_w^-(f)(u_0)\|_p^p \\
 \mathcal{L}_{w,p}(h)(u_0) &> \mathcal{L}_{w,p}(f)(u_0) \\
 0 &> 0
 \end{aligned}$$

Cette dernière inégalité montre clairement une contradiction et conclue la preuve.  $\square$

## 4.5 Application aux problèmes inverses sur graphes pondérés

Dans cette section, nous illustrons le comportement de l'opérateur  $p$ -Laplacien avec termes de gradients  $\mathcal{L}_{w,p}$  appliqué à différents problèmes inverses de restauration ou d'interpolation sur graphes. Notre but ici n'est pas de comparer notre approche aux méthodes de l'état de l'art en mesurant ses performances, mais plutôt d'illustrer le potentiel de cette formulation universelle. Une étude comparative à l'état de l'art sera faite pour des problèmes de classification et de clustering au chapitre 6. Dans cette section, nous montrons en particulier l'applicabilité de cette méthode en construisant des graphes sur des données organisées (images) et non-organisées (nuages de points), ainsi que l'impact du choix des paramètres  $\alpha$  et  $\beta$ .

### 4.5.1 Restauration et simplification d'image

Nous commençons par illustrer le processus de diffusion basé sur le  $p$ -Laplacien avec termes de gradients en l'utilisant pour filtrer des images, définies aussi bien

sur une grille régulière 2D que sur un nuage de point 3D. Nous commençons par montrer qu'en fonction du choix des valeurs des fonctions  $\alpha$  et  $\beta$  le processus de diffusion (4.30) permet de retrouver des opérateurs classiques de la morphologie mathématique. Nous montrons ensuite le processus de diffusion appliqué au filtrage et à la simplification d'image pour  $\alpha = \beta = 0.5$ , puis nous proposons une manière de calculer  $\alpha$  et  $\beta$  en fonction des données à filtrer, afin d'obtenir un processus de filtrage adaptatif.

## Morphologie

Une image composée de  $J$  pixels peut être interprétée comme une fonction discrète  $f_0 : V \rightarrow \mathbb{R}^c$ , qui définit une application de l'ensemble des sommets  $V$  vers un espace couleur de dimension  $c$ . La figure 4.1 montre les processus morphologiques que l'on peut retrouver en utilisant le processus de diffusion (4.30) avec, soit  $\alpha = 0$  pour la première et la troisième ligne (ce qui correspond à une érosion), soit  $\beta = 0$  pour la deuxième et la quatrième ligne (correspondant à une dilatation). Pour cette figure nous avons construit un graphe en 8-connexité avec différentes fonctions de poids. Sur les première et deuxième lignes, nous montrons l'effet de la diffusion sur un graphe représentant la grille des points de l'image avec  $w(u, v) = 1$ . On peut voir que l'on retrouve les effets de l'érosion et la dilatation classique. Pour les troisième et quatrième lignes nous avons utilisé la fonction de poids  $w = s_3$ , où  $s_3$  est la similarité Gaussienne définie à l'équation (1.17), avec  $d = d_2$  la distance Euclidienne entre les couleurs des pixels. Comme on peut le voir, cela permet d'effectuer une érosion ou une dilatation tout en préservant certains détails de l'image.

## Filtrage et lissage morphologique adaptatif

Nous utilisons ici le processus de diffusion afin de filtrer des images et nuages de points. La figure 4.2 montre des résultats obtenus avec différentes fonctions de poids et constructions de graphes (local et non-local), ainsi que différentes valeurs des paramètres  $\alpha$  et  $\beta$ . La première colonne montre les résultats obtenus avec un graphe de 8-connexité avec la fonction de poids  $w(u, v) = 1$ , alors que la deuxième colonne utilise une fonction de poids dépendant de l'intensité des pixels ( $w = s_3$ ) avec  $d = d_2$  la distance Euclidienne dans l'espace couleur de l'image. La troisième colonne montre les résultats obtenus avec un graphe non-local, en utilisant une fenêtre de voisinage de taille  $15 \times 15$  où chaque sommet du graphe est caractérisé par un patch de taille  $5 \times 5$ . La fonction de similarité représente la similarité entre patches, avec  $w = s_3$  et  $d = d_2$  la distance Euclidienne entre les vecteurs de caractéristiques représentant les patches, comme défini à l'équation (1.23) en section 1.2.2. La première et la

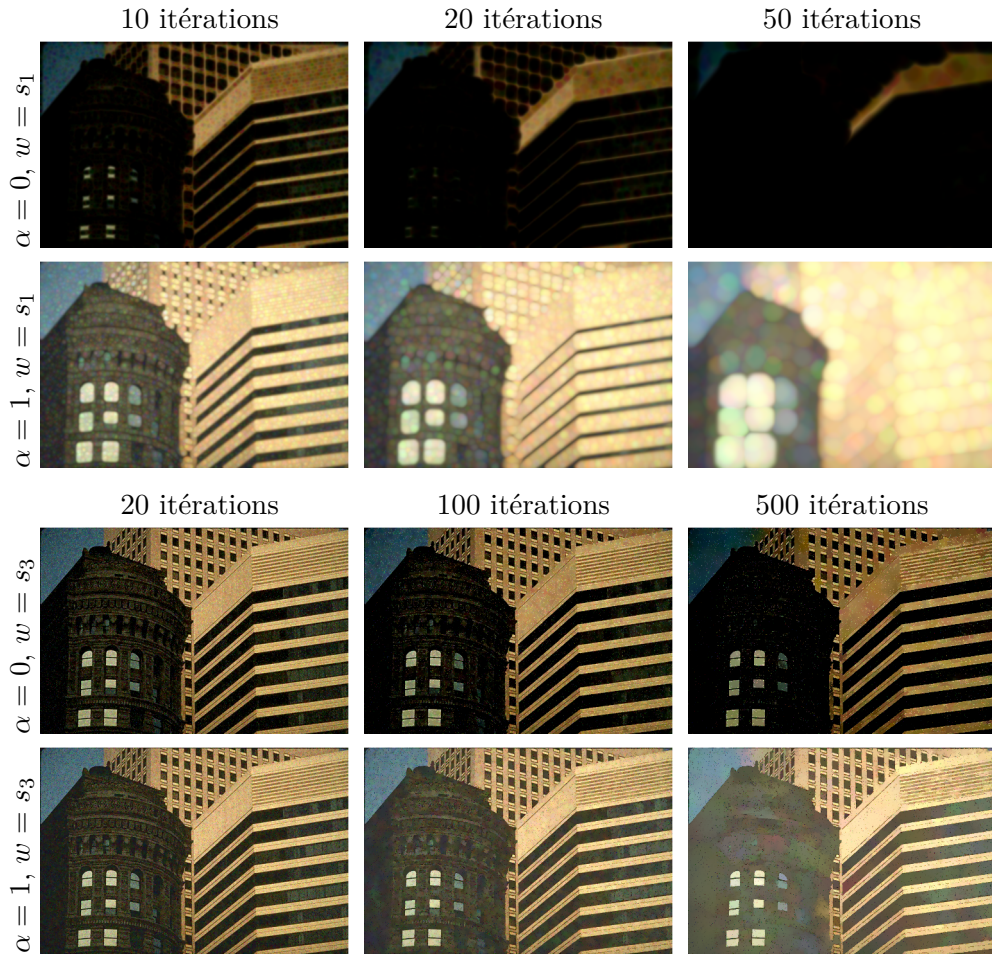


FIGURE 4.1 – Illustration des différentes opérations morphologiques que l’on peut retrouver en utilisant différentes valeurs de  $\alpha$  et  $\beta$ , en utilisant différentes fonctions de poids, avec  $p = 2$ . Voir le texte pour plus de détails.

troisième ligne montrent les résultats obtenus pour  $\alpha = \beta = 0.5$ , ce qui correspond au  $p$ -Laplacien anisotrope (pour le cas  $p = 2$  et  $p = \infty$ ).

Nous proposons maintenant de tirer parti des fonctions  $\alpha$  et  $\beta$  afin d’obtenir un compromis entre diffusion et lissage morphologique, en faisant varier les valeurs de  $\alpha$  et  $\beta$  dépendamment de la courbure moyenne sur graphe. Tout d’abord, nous rappelons la définition de la courbure moyenne d’une fonction sur un graphe, introduite par [CED14] :

$$\kappa_w(u, f) = \frac{\sum_{v \sim u} \sqrt{w(u, v)} \text{sign}(f(v) - f(u))}{\sum_{v \sim u} \sqrt{w(u, v)}}, \quad (4.65)$$





FIGURE 4.2 – Illustration du comportement du  $p$ -Laplacien avec termes de gradients pour le filtrage d'image. Les trois lignes représentent différentes constructions de graphes : 4-connexité avec  $w = s_1$  pour la première,  $w = s_3$  pour la deuxième en utilisant la similarité couleur et un  $k$ -ppv en utilisant la similarité entre patchs pour la troisième. La première et la troisième colonne montrent le comportement du filtrage avec  $\alpha = \beta = 0.5$ , ce qui correspond au  $p$ -Laplacien anisotrope sur graphe. La seconde et la quatrième colonne montrent l'utilisation des fonctions  $\alpha$  et  $\beta$  adaptatives, en utilisant la courbure moyenne d'une fonction sur graphe. Voir le texte pour plus de détails.



où la fonction sign est définie telle que :

$$\text{sign}(x) = \begin{cases} 1, & \text{si } x \geq 0, \\ -1, & \text{si } x < 0. \end{cases} \quad (4.66)$$

On peut réécrire (4.65) telle que :

$$\kappa_w(u, f) = \frac{\sum_{v \in \{v | f(v) \geq f(u)\}} \sqrt{w(u, v)} - \sum_{v \in \{v | f(v) < f(u)\}} \sqrt{w(u, v)}}{\sum_{v \sim u} \sqrt{w(u, v)}}. \quad (4.67)$$

Afin de pouvoir tirer parti de cette formulation de la courbure avec l'opérateur  $\mathcal{L}_{w,p}$  et de respecter la contrainte  $\alpha(u) + \beta(u) = 1$ , nous l'avons adaptée afin que la fonction  $\alpha$  représente la partie positive de la courbure et que  $\beta$  en représente la partie négative :

$$\alpha(u) = \frac{\sum_{v \in \{v | f(v) \geq f(u)\}} \sqrt{w(u, v)}}{\sum_{v \sim u} \sqrt{w(u, v)}}, \quad (4.68)$$

et

$$\beta(u) = \frac{\sum_{v \in \{v | f(v) < f(u)\}} \sqrt{w(u, v)}}{\sum_{v \sim u} \sqrt{w(u, v)}}. \quad (4.69)$$

Nous pouvons maintenant réécrire le processus de diffusion (4.30) en utilisant les équations (4.8) et (4.65) :

$$\begin{cases} \frac{\partial f(u, t)}{\partial t} = & 2 \min(\alpha(u), \beta(u)) \Delta_{w,p} f(u) \\ & + (\kappa_w(u, f))^+ \|\nabla_w^+(f)(u)\|_{p-1}^{p-1} \\ & - (\kappa_w(u, f))^- \|\nabla_w^-(f)(u)\|_{p-1}^{p-1} \\ f(u, t = 0) = & f_0(u). \end{cases} \quad (4.70)$$

Pour  $\kappa_w(u, f) > 0$  nous obtenons :

$$\frac{\partial f(u, t)}{\partial t} = 2\beta(u) \Delta_{w,p} f(u) + \kappa_w(u, f) \|\nabla_w^+(f)(u)\|_{p-1}^{p-1}, \quad (4.71)$$

pour  $\kappa_w(u, f) < 0$  nous obtenons :

$$\frac{\partial f(u, t)}{\partial t} = 2\alpha(u) \Delta_{w,p} f(u) + \kappa_w(u, f) \|\nabla_w^-(f)(u)\|_{p-1}^{p-1}. \quad (4.72)$$

Enfin, pour  $\kappa_w(u, f) = 0$ , nous obtenons :

$$\frac{\partial f(u, t)}{\partial t} = \Delta_{w,p} f(u). \quad (4.73)$$

Ceci nous permet d'ajuster adaptativement le processus de filtrage entre érosion et dilatation suivant la courbure moyenne de la fonction en chaque sommet du graphe, ce qui permet de combiner une opération de lissage (avec le terme Laplacien) et le filtrage de choc (avec soit la dilatation soit l'érosion) dans la même formulation. La figure 4.2 montre des résultats en utilisant cette adaptation de la courbure pour le filtrage d'image à la deuxième et quatrième colonne pour  $p = 2$  et  $p = \infty$  respectivement. Sur cette figure, on peut voir que pour  $\alpha$  et  $\beta$  dynamique, l'effet global est de lisser tout en préservant les contours des objets. La différence est surtout visible pour  $w(u, v) = 1$ , où le processus de diffusion classique lisse complètement l'image, que ce soit pour  $p = 2$  ou  $p = \infty$ , alors que l'utilisation de la courbure tend à conserver quelque peu les contours (beaucoup plus que la diffusion). Dans le cas des images où le graphe est pondéré avec la similarité couleur (deuxième ligne) l'effet de la courbure est beaucoup moins visible, cependant on remarque que les contrastes sont mieux préservés qu'en utilisant la diffusion seule, spécialement pour  $p = \infty$ . Pour les images où le graphe est non-local et pondéré avec la similarité entre patches, on peut voir que le processus de filtrage utilisant la courbure débruite globalement plus tout en conservant les contours des objets ainsi que les structures géométriques. Cet effet est assez visible au niveau des yeux et du nez, mais aussi à la frontière des cheveux et du front.

Afin de montrer l'adaptabilité de notre approche, nous montrons l'effet du processus de diffusion sur les couleurs d'un nuage de point 3D à la figure 4.3 avec  $p = 2$ . Pour la première et la deuxième colonne de cette figure, nous avons construit un graphe des  $k$  plus proches voisins à partir de l'ensemble  $\Omega \subset \mathbb{R}^3$  des coordonnées du nuage de points, en utilisant  $w = s_1$  et  $w = s_3$  pour la première et la deuxième colonne, respectivement. Le graphe utilisé pour générer les résultats présents sur la troisième colonne a été construit, toujours en utilisant un graphe des  $k$  plus proches voisins, mais en utilisant la définition des patches sur nuages de points, proposée par [LEL14a]. La première ligne montre les résultats en utilisant des valeurs pour  $\alpha$  et  $\beta$  constantes ( $\alpha(u) = \beta(u) = 0.5, \forall u \in V$ ), ce qui correspond avec cette construction de graphe à un processus de diffusion non-local. La deuxième ligne montre des résultats en utilisant les fonctions  $\alpha$  et  $\beta$  adaptatives définies précédemment (équations (4.68) et (4.69)). On observe ici les mêmes effets que pour le filtrage d'image à la figure 4.2 : Pour  $w(u, v) = 1$ , le processus de diffusion a complètement lissé les couleurs du nuage de point, ce qui n'est pas le cas en utilisant la courbure (les couleurs sont quand même lissées, mais les transitions sont plus franches qu'avec le processus de diffusion seul). On peut remarquer le même effet lors de l'utilisation de graphes pondérés (deuxième et troisième colonne), même s'il est beaucoup moins prononcé, les poids conservant déjà les frontières des objets.

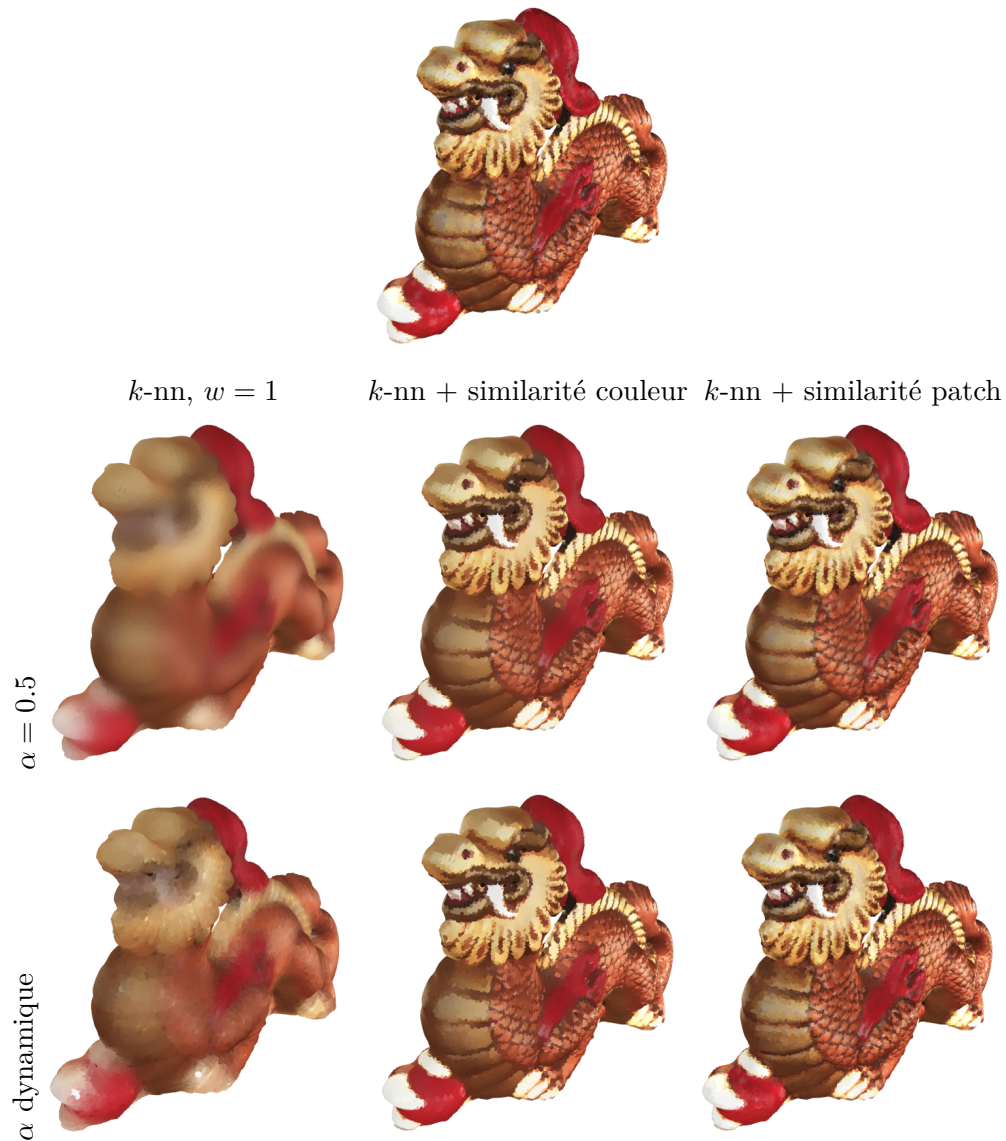


FIGURE 4.3 – Lissage des couleurs d’un nuage de point 3D en utilisant différentes fonctions de poids et  $p = 2$ . Voir le texte pour plus de détails

### Filtre de choc

La formulation de l’opérateur que nous proposons permet aussi de retrouver le filtre de choc que nous avons présenté dans [SET15]. Comme expliqué au chapitre 2, le filtre de choc combine érosion et dilatation en utilisant le signe du Laplacien, et s’exprime avec l’EDP suivante :

$$\begin{cases} \frac{\partial f}{\partial t}(x, t) = -\text{sign}(\Delta_p(f)(x, t))|\nabla(f)(x, t)|, \\ f(x, 0) = f_0(x), \end{cases} \quad (4.74)$$

où  $p \in \{2, \infty\}$ .

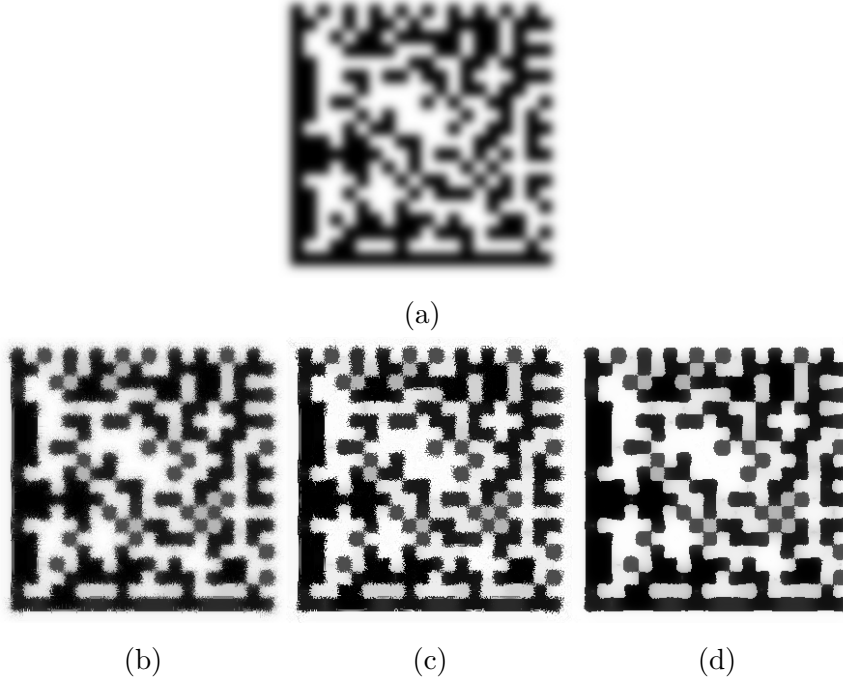


FIGURE 4.4 – Filtre de choc sur une image de QR code en utilisant  $p = 2$  avec  $w(u, v) = 1$ . Les images (b,c,d) présentent les résultats en utilisant 4, 8 et 48 voisins, respectivement. On remarque que le résultat en utilisant un plus grand voisinage est meilleur. Ceci est due à une meilleure approximation du Laplacien. Voir le texte pour plus de détails.

Dans [SET15], nous avons étendu ce filtre sur des graphes pondérés en utilisant les définitions du  $p$ -Laplacien et des gradients morphologiques sur graphes. La transcription de ce filtre sur graphe peut s'exprimer par l'EdP suivante :

$$\frac{\partial f}{\partial t} = k_{w,p}^+ \|\nabla_w^+ f\|_p - k_{w,p}^- \|\nabla_w^- f\|_p, \quad (4.75)$$

où  $k_{w,p} = -\text{sign}(\Delta_{w,p})$ ,  $k_{w,p}^+ = \max(k_{w,p}, 0)$ ,  $k_{w,p}^- = -\min(k_{w,p}, 0)$  avec  $p = 2$  ou  $p = \infty$ .

On peut voir aisément qu'en utilisant notre processus de diffusion (4.30), avec  $\alpha(u) = k_{w,p}^+$  et  $\beta(u) = k_{w,p}^-$ , on retrouve ici le filtre de choc sur graphe.

Nous illustrons ce processus sur des images aux figures 4.4 et 4.5. Il est intéressant de remarquer ici que le graphe utilisé pour calculer le signe du Laplacien et le graphe utilisé pour les termes de gradients peuvent être différents. Pour ces expérimentations, nous avons utilisé un graphe d' $\epsilon$ -voisinage afin d'estimer le Laplacien, avec  $w(u, v) = 1$ . Pour les figures 4.4(b) et 4.5(b), nous avons utilisé une distance de Manhattan avec  $\epsilon = 1$ . Pour les figures 4.4(c), 4.4(d), 4.5(c) et 4.5(d), nous avons utilisé une distance de Chebyshev avec  $\epsilon = 1$  pour les figures 4.4(c) et 4.5(c), et  $\epsilon = 3$  pour les figures 4.4(d) et 4.5(d). Le graphe utilisé avec les termes de gradients pour chacune de ces figures est un graphe de 8-connexité avec  $w(u, v) = 1$ . On remarque ici que l'utilisation d'un plus grand voisinage pour le Laplacien améliore sensiblement les résultats.



(a)



(b)

(c)

(d)

FIGURE 4.5 – Filtre de choc sur une image naturelle, avec  $p = 2$  et  $w(u, v) = 1$ . Comme pour la figure 4.4, nous avons utilisé 4, 8 et 48 voisins lors de la construction du graphe, pour les images (b,c,d), respectivement.



Nous avons aussi illustré le filtre de choc sur un nuage de point 3D à la figure 4.6. Ici nous avons construit un graphe des  $k$ -ppv spatial, avec  $w(u, v) = 1$ . On peut voir que l'effet du filtre est le même que sur les images, réhaussant les transitions.

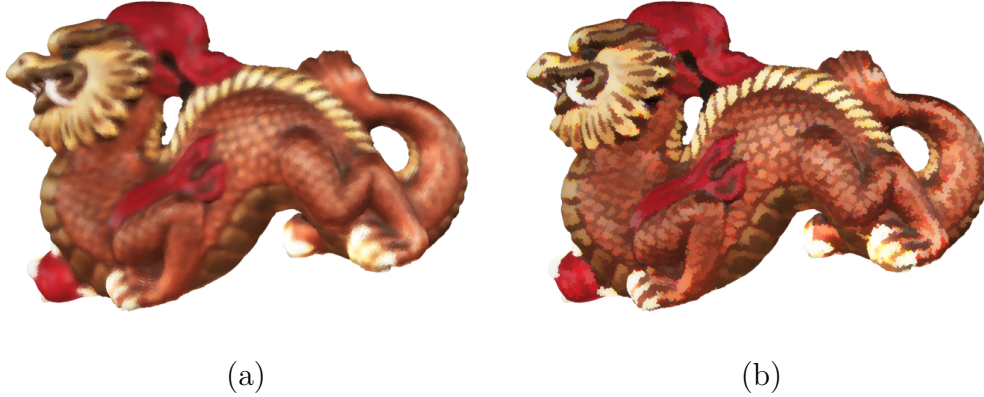


FIGURE 4.6 – Filtre de choc sur un nuage de point 3D avec  $p = 2$ . Le graphe est construit comme un  $k$ -ppv spatial. La figure (b) présente le résultat en utilisant  $w(u, v) = 1$  et  $k = 10$ .

À la figure 4.7, nous montrons le filtre de choc appliqué sur un graphe d'adjacence de région, en associant à chaque sommet du graphe la couleur moyenne de la région correspondante. Ce graphe n'est pas pondéré ( $w(u, v) = 1$ ). La figure 4.7(b) montre la carte de région lissée. Les figures 4.7(c) et 4.7(d) montrent l'application du filtre de choc sur ces données, en utilisant  $p = 2$  et  $p = \infty$ , respectivement. Comme pour l'expérimentation sur le nuage de point (figure 4.6), on retrouve les mêmes effets que sur les images.

## 4.5.2 Contours actifs

Dans cette section nous considérons la segmentation par contour actif sur graphe. Le contour est représenté implicitement par le niveau zéro d'une fonction d'ensemble de niveau, partitionnant le graphe en deux ensembles. Afin d'adapter cette idée à notre formulation, nous définissons la fonction d'ensemble de niveau initiale  $\phi_0$ , pour deux ensembles de sommets  $A$  et  $B$  avec  $A \cup B = V$  telle que :

$$\begin{cases} \phi_0(u) = -1, & \text{si } u \in A, \\ \phi_0(u) = 1, & \text{si } u \in B. \end{cases} \quad (4.76)$$

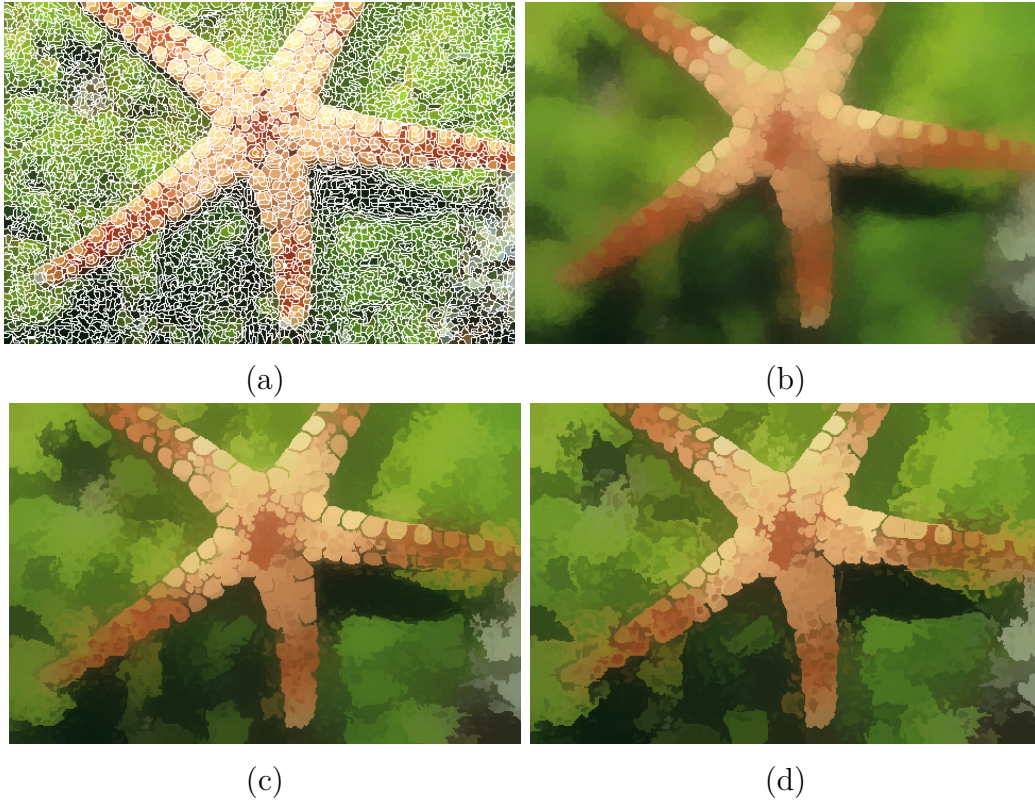


FIGURE 4.7 – Filtre de choc sur des données définies sur un graphe d’adjacence de région. La figure (a) montre la carte de région et la figure (b) montre la carte de région lissée. La donnée associée à chaque noeud est la couleur moyenne de la région. Les figures (c) et (d) présentent les résultats en utilisant  $w(u, v) = 1$ , avec  $p = 2$  et  $p = \infty$ , respectivement.

L’équation d’évolution du contour peut maintenant être donnée telle que :

$$\begin{cases} \frac{\partial \phi(u, t)}{\partial t} = \mathcal{L}_{w, p}(\phi)(u) , \\ \phi(u, 0) = \phi_0(u) . \end{cases} \quad (4.77)$$

Afin de formuler un algorithme de contours actifs sur graphe, nous utilisons (4.77) et nous définissons les valeurs des fonctions  $\alpha$ ,  $\beta$  comme la courbure moyenne sur graphe ((4.68), (4.69)). La figure 4.8 montre l’évolution d’un contour sur une image en utilisant cette méthode. Afin que la courbe puisse évoluer en fonction de l’information contenue sur les arêtes, nous avons construit le graphe comme un graphe de 8-connexité augmenté, *i.e.*, nous construisons d’abord le graphe de 8-connexité, puis nous ajoutons un certain nombre de voisins choisis aléatoirement dans une fenêtre autour du pixel

considéré. Dans cet exemple, la fonction de poids dépend de la similarité couleur entre les pixels ( $w = s_3$ ). Cette construction de graphe permet d'obtenir un graphe non-local sans ajouter trop d'arêtes, gardant sa taille relativement petite.

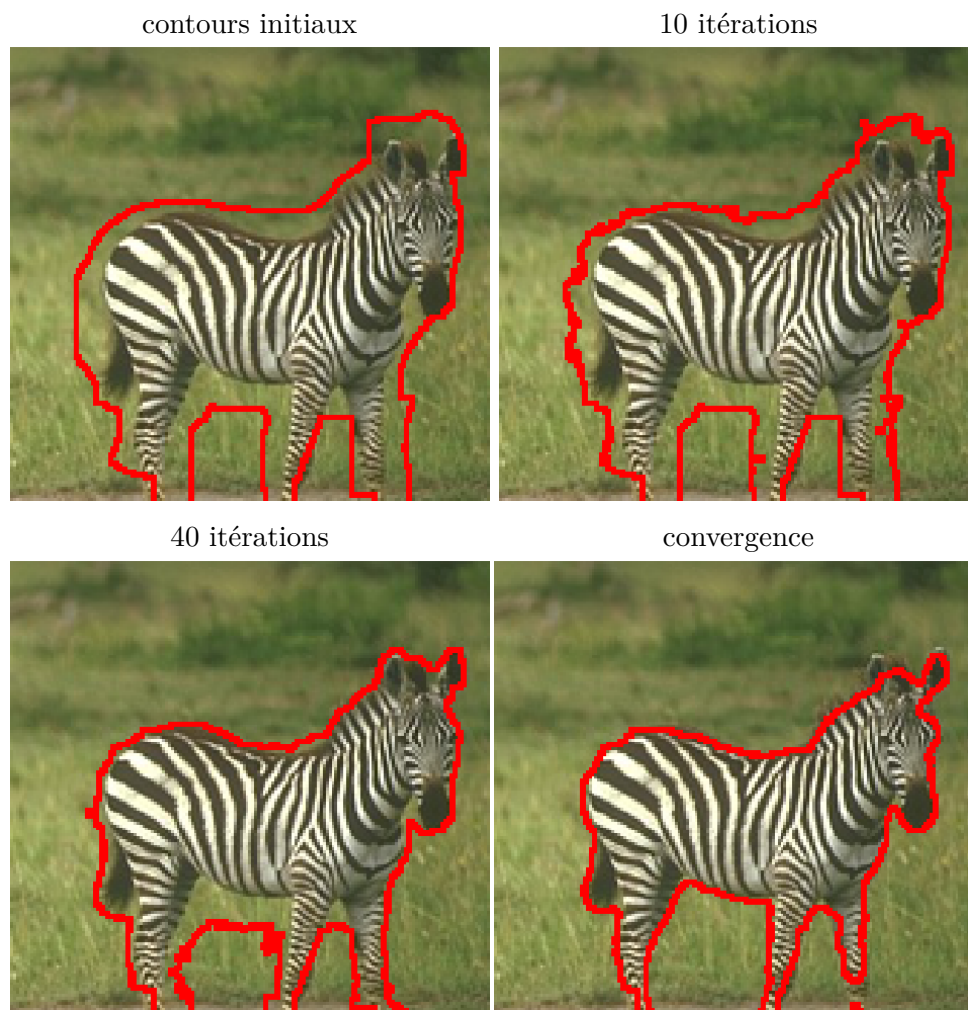


FIGURE 4.8 – Illustration du comportement de l'algorithme de contours actifs basé sur la courbure moyenne sur graphe utilisant le  $p$ -Laplacien avec termes de gradient proposé avec  $p = 2$ . La première image (en haut à gauche) montre les contours initiaux sur l'image. Les images suivantes montrent l'évolution du contour au fur et à mesure des itérations.



### 4.5.3 Interpolation

Différentes tâches en traitement d'images, vision par ordinateur et apprentissage peuvent être formulées comme des problèmes d'interpolations. La colorisation d'images et de vidéos, l'inpainting et la segmentation ou la classification semi-supervisée sont des exemples de ces problèmes d'interpolation. L'interpolation de données consiste à créer de nouvelles valeurs pour les données manquantes, en cohérence avec un ensemble de valeurs connues. Nous proposons ici d'illustrer le comportement du  $p$ -Laplacien avec termes de gradients pour ces problèmes d'interpolation, en les considérant comme solutions du problème de Dirichlet suivant :

$$\begin{cases} \mathcal{L}_{w,p}(f)(u) = 0, & u \in V_0, \\ f(u) = g(u), & u \in \partial^+ V_0 \end{cases} \quad (4.78)$$

où  $V_0 \subset V$  est le sous ensemble de sommets représentant l'information manquante. La valeur de la fonction  $g$  représente l'information connue et est dépendante de l'application. Celle-ci sera spécifiée pour chaque application dans la suite.

#### Segmentation interactive d'image

Dans cette section nous illustrons l'interpolation de germes dans le cas de la segmentation semi-supervisée d'image.

Comme au chapitre précédent, dans le cas de la segmentation semi-supervisée d'image, nous considérons ce problème comme un problème d'interpolation, où la fonction à interpoler est la fonction des marqueurs initiaux. Nous rappelons ici la formulation utilisée à la section 3.6.2. En utilisant l'équation (4.78) et en considérant deux classes  $A$  et  $B$  ( $A \cup B$  est l'ensemble de germes initiaux donnés par l'utilisateur), la fonction de marqueur initiale  $g$  est définie telle que :

$$\begin{cases} g(u) = -1, & \text{si } u \in A, \\ g(u) = 1, & \text{si } u \in B, \\ g(u) = 0, & \text{sinon.} \end{cases} \quad (4.79)$$

À convergence, l'appartenance d'un sommet à une classe peut être calculé simplement en seillant le signe de  $f$ .

*Remarque :* Dans le cas où il y a plus de deux classes ( $N$  classes), on peut effectuer une segmentation multi-classes en résolvant le système (4.78)  $N$  fois en considérant le marqueur  $A$  comme une classe et  $B$  comme toutes les autres

classes. Dans ce cas, la fonction de marqueur  $L : V \rightarrow \{C_i\}_{i=1,\dots,N}$  associant à chaque sommet une classe est définie telle que :

$$L(u) = C_i | f_i(u) = \max_{j=1,\dots,N} f_j(u). \quad (4.80)$$

Pour la figure 4.9, le graphe est construit de la même manière que pour la méthode des contours actifs précédemment décrite (section 4.5.2) et la fonction de poids dépend aussi de la similarité couleur entre pixels. La fonction  $f_0 : V \rightarrow \mathbb{R}$  à interpoler est initialisée à partir de germes dessinés par l'utilisateur. La figure 4.9 présente une image initiale avec des germes (bleu et verts) et le résultat de l'interpolation pour  $p = 2$ .



FIGURE 4.9 – Segmentation interactive utilisant le  $p$ -Laplacien avec termes de gradient, avec  $p = 2$ . Voir le texte pour plus de détails.

### Segmentation de graphe d'adjacence de région

Dans cette section, nous discutons de l'interpolation de germes / marqueurs initiaux pour la segmentation semi-supervisée en utilisant des graphes d'adjacence de régions. Ceci illustre l'adaptabilité de l'approche proposée pour traiter des données définies sur un graphe irrégulier, ainsi qu'une manière efficace d'extraire des objets similaires non connectés avec très peu de germes fournis par l'utilisateur. Nous avons construit le graphe d'adjacence de régions à partir de l'image initiale en utilisant l'approche par super sommet présentée dans [DEL13] afin de créer la carte de région initiale. Ce graphe est étendu en utilisant les  $k$  plus proches voisins afin d'ajouter des arêtes supplémentaires entre chaque région et les  $k$  régions qui lui sont les plus similaires, au sens de la couleur moyenne de chaque régions. L'un des avantages principal de cette construction de graphe est tout d'abord le fait de travailler sur une version réduite de l'image, augmentant considérablement la vitesse de traitement. Le deuxième avantage est que les arêtes additionnelles permettent de connecter

des régions qui ne l'étaient pas dans la représentation initiale de l'image. La fonction de poids dépend pour notre illustration de la similarité couleur entre sommet (chaque région est représentée par sa couleur moyenne). La fonction  $f_0 : V \rightarrow \mathbb{R}$  à interpoler est initialisée en fonction des germes fournis par l'utilisateur. La figure 4.10 présente l'image initiale avec les germes à interpoler dessinés par l'utilisateur (rouge et bleu), la carte de région et le résultat de l'interpolation avec les paramètres  $\alpha = \beta$ ,  $p = 2$  et  $w = s_3$  (1.17).

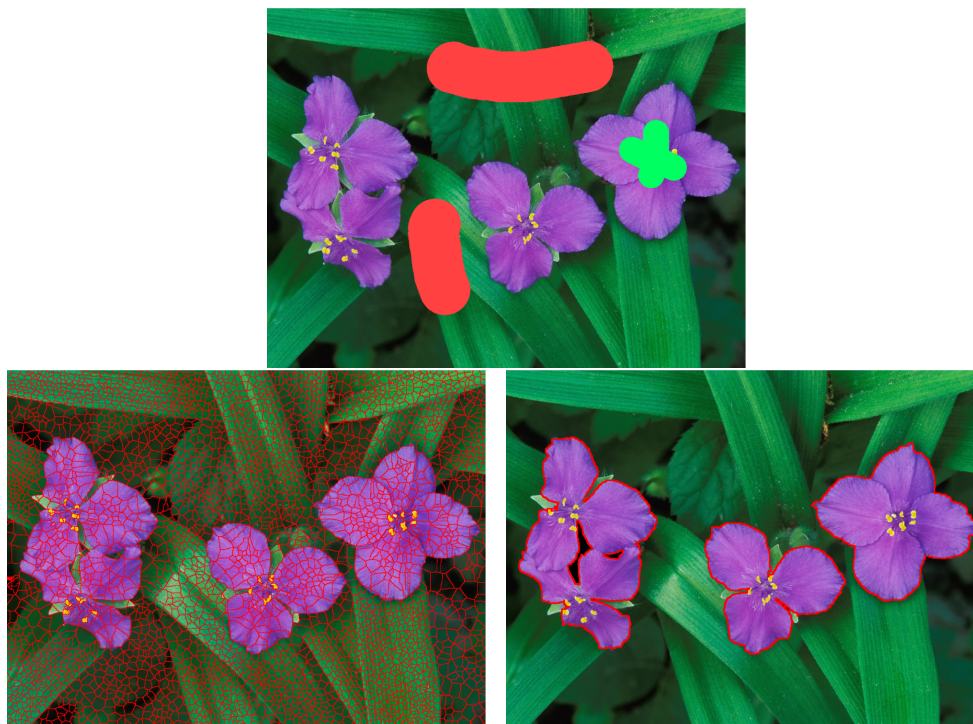


FIGURE 4.10 – Segmentation interactive d'une image en utilisant un graphe d'adjacence de régions, étendue avec des arêtes additionnelles, afin de connecter les régions similaires au sens de la couleur. Cette construction permet de diffuser les germes initiaux à travers des objets non connectés spatialement (*e.g.* les fleurs dans cette image).

### Classification de données

Notre méthode étant définie sur graphe, nous pouvons utiliser directement l'équation (4.78) afin de partitionner des données. Nous illustrons ici le comportement de la méthode pour  $p = \infty$ . Nous avons utilisé ici un sous-ensemble composé de 200 imageries représentant des chiffres (zéro et un) de la

base de données USPS [Hul94]. Le but est de partitionner le graphe en deux classes (les zéros et les uns). Afin de simplifier la construction du graphe, nous considérons chaque imagerie comme un vecteur de taille 256 (le nombre de pixels de chaque imagerie). La métrique utilisée est la distance Euclidienne. Nous construisons un graphe des  $k$  plus proches voisins sur cet ensemble de données, où chaque sommet représente une imagerie. Nous avons placé un germe par classe afin d'initialiser le processus d'interpolation de label. La figure 4.11 montre le graphe construit avec les germes initiaux, ainsi que le résultat de partitionnement.

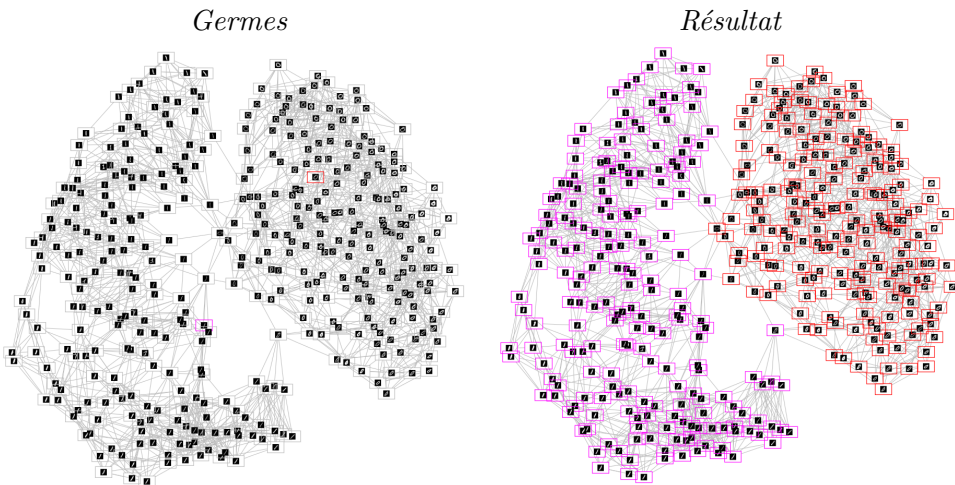


FIGURE 4.11 – Classification de données par interpolation sur un échantillon de données provenant de la base USPS. L'image de gauche montre le graphe construit sur cet échantillon, avec des germes initiaux. L'image de droite montre le résultat de la classification.

### Inpainting non-local

Nous illustrons tout d'abord l'inpainting sur des images à la figure 4.12 avec  $\alpha = \beta = 0.5$  et  $p = 2$ , ce qui correspond à  $\Delta_{w,2}$ . Ici la fonction à interpoler est l'image elle-même, avec  $V_0$  la portion d'image manquante. Le graphe est soit un graphe local (le voisinage est défini comme une 8-connexité) pour la troisième ligne, soit un graphe non-local construit en utilisant une fenêtre de voisinage de taille  $31 \times 31$  et des patches de taille  $15 \times 15$ , en utilisant la similarité entre patches comme fonction de poids (quatrième ligne).

Nous illustrons aussi l'inpainting sur graphe d'un nuage de point 3D à la figure 4.13. Dans ce cas, le graphe est construit comme un graphe non-local



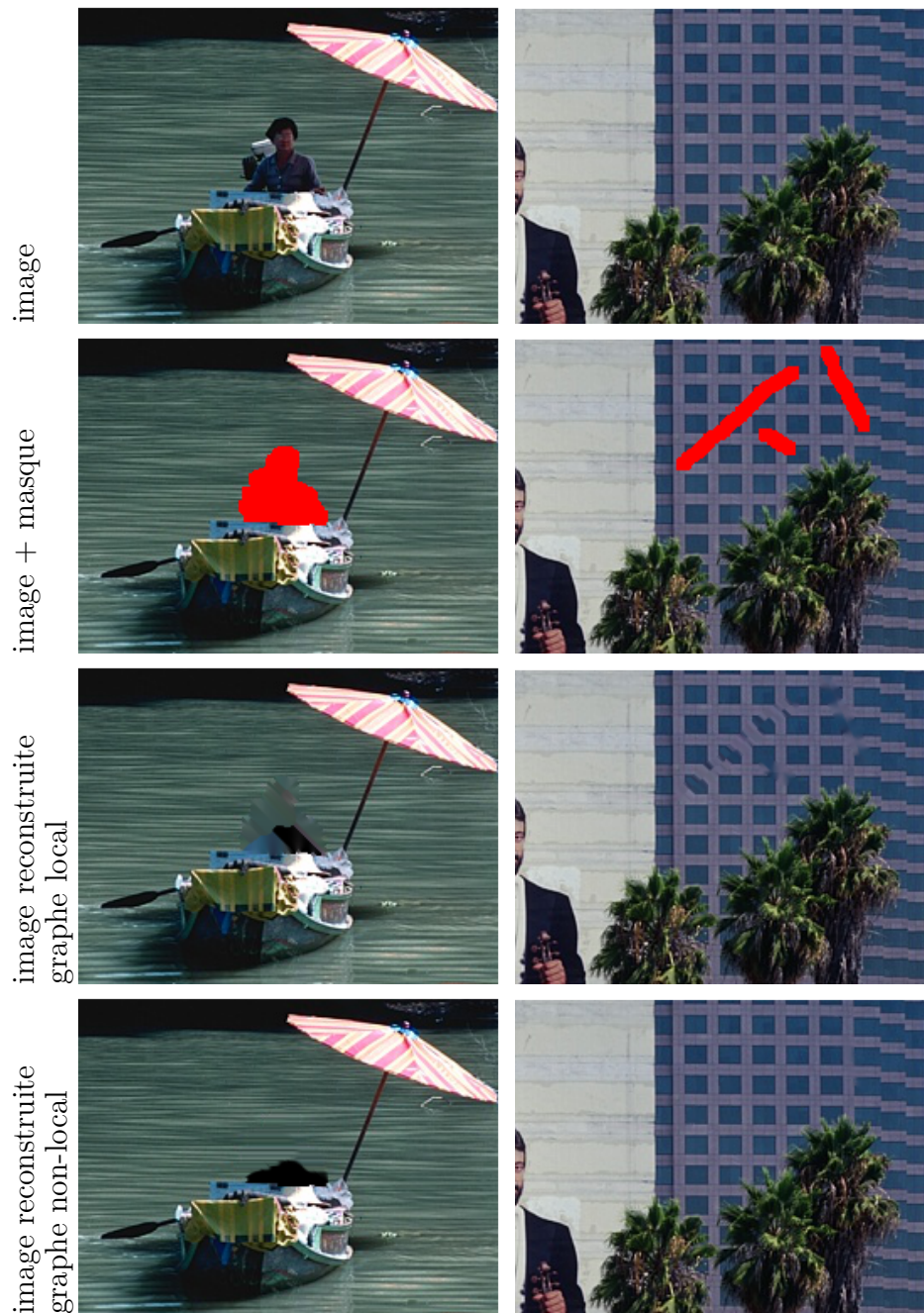


FIGURE 4.12 – Inpainting d’images naturelles en utilisant un graphe local et non-local. La troisième ligne montre le résultat en utilisant un graphe de 8-connexité. La quatrième ligne montre le résultat en utilisant une fenêtre de voisinage de taille  $31 \times 31$  et des patches de tailles  $15 \times 15$ .

à partir du nuage de points en utilisant la définition de patches sur nuage de points proposée par [LEL13]. La fonction  $f$  à interpoler associe un vecteur couleur à chaque sommet du graphe. Les résultats sont présentés pour plusieurs valeurs des paramètres  $\alpha, \beta$  et  $p$ , correspondant à  $\Delta_{w,\infty}, \Delta_{w,2}$ , un processus de dilatation et un processus d'érosion.

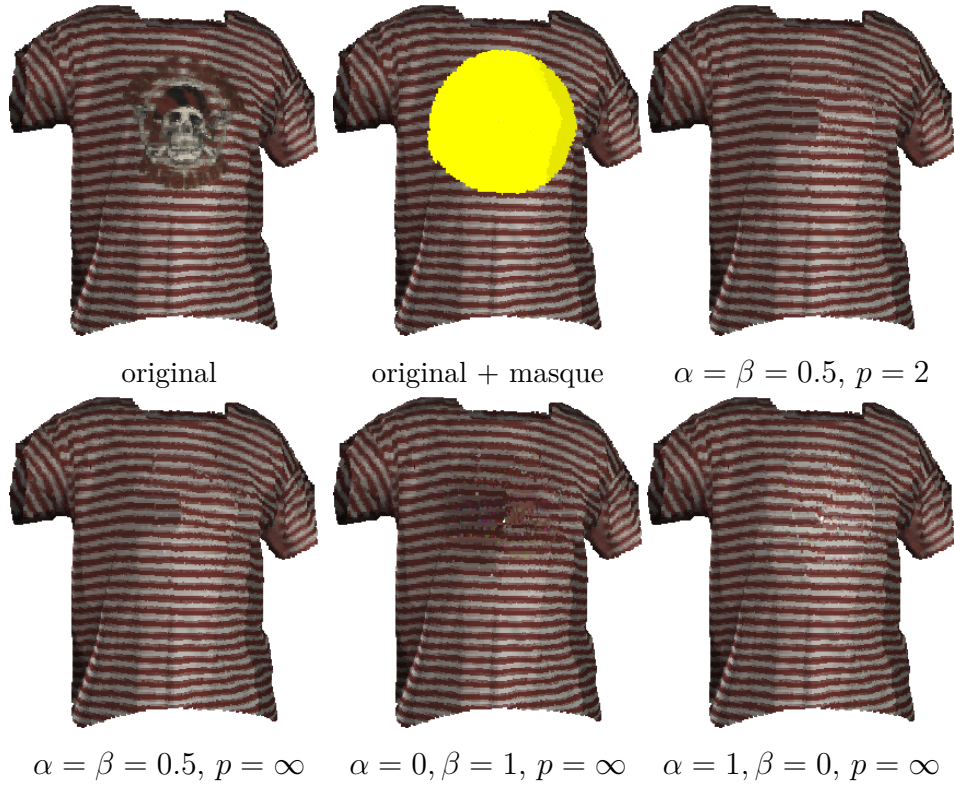


FIGURE 4.13 – Illustration d'inpainting sur un nuage de points 3D. Les résultats sont calculés ici pour  $\alpha = \beta = 0.5$  avec  $p = 2$  et  $p = \infty$ , et  $\alpha = 0, \beta = 1, \alpha = 1, \beta = 0$  avec  $p = \infty$ .

## 4.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle classe de  $p$ -Laplaciens et de Laplaciens infinis sur graphes, qui peut être exprimée comme une combinaison convexe entre deux termes de gradients directionnels. Une des caractéristiques intéressantes de cette représentation tient au fait que les coefficients devant les termes de gradients puissent être choisis dynamiquement en fonction des données, conduisant à des effets de filtrages adaptatifs (diffusion non-locale

et filtrage morphologique) dans différentes régions des même données. Nous nous sommes intéressées aux connexions entre l'opérateur que nous proposons et des EDPs locales et non-locales de la littérature, ainsi qu'avec le jeu du Tug-of-War. Nous avons montré que cet opérateur, en utilisant la norme  $\mathcal{L}_\infty$ , coïncide avec les fonctions de valeurs de différents Tug-of-War. Nous avons aussi appliqué cette nouvelle classe de  $p$ -Laplacien et Laplacien infini à travers deux EdPs sur graphes pondérés : une équation parabolique ainsi qu'une équation elliptique en utilisant des conditions aux bords de Dirichlet. Nous avons aussi prouvé des propriétés mathématiques importantes, telles que l'existence et l'unicité de la solution à l'équation elliptique. Nous avons montré que l'équation parabolique conduit à une généralisation de la diffusion et de la morphologie mathématique sur graphe et que l'équation elliptique avec des conditions aux bords de Dirichlet généralise les processus d'interpolation sur des domaines discrets. Enfin, nous avons illustré la façon dont cette nouvelle classe d'opérateurs sur graphe peut être appliquée à travers plusieurs exemples, *i.e.*, en débruitage, segmentation, inpainting et clustering. Afin de souligner l'applicabilité universelle de la formulation proposée, nous avons testé nos algorithmes sur une large variété de données, *i.e.*, des images ainsi que des nuages de points 3D ou de plus hautes dimensions tel que les bases de données.

# Chapitre 5

## Équation de Poisson et de Hamilton-Jacobi sur graphe

### Sommaire

---

<b>5.1</b>	<b>Introduction</b>	<b>122</b>
<b>5.2</b>	<b>Tug-of-War et équation de Poisson infini</b>	<b>123</b>
<b>5.3</b>	<b>Équation de Poisson infini non-locale avec termes de gradients</b>	<b>125</b>
5.3.1	Du Tug-of-War à l'équation de Poisson sur graphe	125
5.3.2	Calcul de distance généralisé sur graphe	128
<b>5.4</b>	<b>Applications</b>	<b>129</b>
5.4.1	Distances géodésiques pondérées	129
5.4.2	Segmentation et classification de données	132
5.4.3	Efficacité du schéma de résolution	139
<b>5.5</b>	<b>Conclusion</b>	<b>139</b>

---

### Publication associée à ce chapitre

[TELM15] Matthieu TOUTAIN, Abderrahim ELMOATAZ, François LOZES et Alamin MANSOURI : Non-local discrete  $\infty$ -poisson and hamilton jacobi equations : From Stochastic Game to Generalized Distances on Images, Meshes, and Point Clouds *Journal of Mathematical Imaging and Vision*, pages 1–13, 2015.



## 5.1 Introduction

Le but principal de ce chapitre est d'adapter et de résoudre une équation hybride de Poisson infini et de Hamilton-Jacobi sur un domaine discret général : un graphe pondéré de topologie arbitraire. Nous introduisons cette adaptation comme une EdP utilisant le Laplacien infini avec termes de gradient introduit au chapitre précédent (à l'équation (4.7)), pouvant être interprété comme une combinaison de gradients directionnels.

Notre motivation principale est de proposer un schéma de résolution numérique simple et unifié afin d'approximer des distances généralisées sur images, maillages, nuages de points, ou toutes données pouvant être représentées par des graphes pondérés.

Le calcul de fonction de distance peut être appliqué dans de nombreux domaines, tel que le traitement d'image, l'infographie, la robotique, ou encore la géométrie algorithmique. De plus, en utilisant la fonction de distance d'un germe à une cible, on peut calculer le chemin géodésique correspondant, utilisé dans de nombreuses applications, *e.g.* pour calculer le squelette d'une forme, les diagrammes de Voronoï, ou faire de l'édition de maillage.

Dans le contexte d'une grille régulière, résoudre numériquement ces équations est simple. La régularité de la grille fournit un domaine où la discrétisation des équations est directe. Dans le cas de l'équation de Poisson, on peut utiliser différentes méthodes afin de calculer efficacement une solution, via la transformée de Fourier par exemple, ou les méthodes multigrid [GGS<sup>+</sup>06]. Pour l'équation de Poisson infini ou l'équation eikonale, on peut utiliser la méthode de différences finies [Obe13], ou encore celle des éléments finis.

Dans le cas des maillages ou des surfaces courbes générales, résoudre ces équations est plus difficile. Le traitement numérique de ces EDPs requiert une représentation adaptée de la géométrie de la surface. On peut utiliser la paramétrisation de surface, comme les maillages triangulés, et utiliser soit une représentation explicite pour définir des opérateurs différentiels sur celle-ci, ou encore utiliser la géométrie intrinsèque pour définir des opérateurs différentiels directement sur les triangles. Une autre manière serait de représenter implicitement la surface par des ensembles de niveaux ou en utilisant la méthode des plus proches voisins [RM08]. De plus, dans le cas des nuages de points 3D, la connectivité des points n'est pas fournie, ce qui ajoute des difficultés supplémentaires dans le traitement de ces données.

Dans ce chapitre, nous proposons une méthode numérique simple et unifiée pour résoudre et adapter une équation hybride de Poisson infini et de Hamilton-Jacobi sur des domaines discrets organisés ou non.

En utilisant le framework des EdPs sur graphes ainsi que l'opérateur Laplacien infini avec termes de gradient introduit dans le chapitre précédent, nous interprétons d'abord le jeu du Tug-of-War relatif à l'équation de Poisson infini et de Hamilton-Jacobi comme une EdP sur un graphe Euclidien particulier. Ensuite, en étendant ces mêmes équations aux graphes pondérés, nous proposons une EdP générale à coefficients variables.

En fixant ces coefficients différemment, nous pouvons adapter ce schéma général à différentes applications en traitement d'image, de maillage, de nuage de point, etc.

Les principales contributions de ce chapitre sont les suivantes :

- Nous proposons une interprétation à la fois de l'équation de Poisson infini continue et d'une équation hybride Poisson-infini-Hamilton-Jacobi continue, comme une EdP sur des graphes particuliers.
- Nous proposons une extension de ces EdPs sur graphes pondérés de topologie arbitraire et nous montrons la relation entre cette EdP générale et le jeu du Tug-of-War non-local.
- Enfin, nous proposons de résoudre ces équations à l'aide d'un schéma morphologique simple, en utilisant les opérateurs morphologiques de dilatation (2.63) et d'érosion (2.64) sur graphes, définis au chapitre 2.

## 5.2 Tug-of-War et équation de Poisson infini

Soient  $\Omega \subset \mathbb{R}^m$  et  $\partial\Omega$  sa frontière. Nous notons  $d(x)$  la distance minimale de  $x \in \Omega$  à  $\partial\Omega$ . Une approche générale pour approximer simplement une fonction de distance lisse consiste à résoudre l'équation de Poisson avec des conditions aux bords de Dirichlet, définie telle que :

$$\begin{cases} \Delta u(x) = h(x) & , x \in \Omega, \\ u(x) = g(x) & , x \in \partial\Omega, \end{cases} \quad (5.1)$$

où  $u$ ,  $h$  et  $g$  sont des fonctions à valeurs réelles sur le domaine  $\Omega$ , et  $\Delta$  est l'opérateur de Laplace. Cette équation est classiquement utilisée dans des champs variés, tels que l'électrostatique, la gravité Newtonienne et plus récemment dans la reconstruction de surface [KBH06, CT11].

Pour calculer une estimation de la distance d'un point  $x \in \Omega$  à  $\partial\Omega$ , une approche commune est de fixer  $h(x) = -1$  et  $g(x) = 0$ . En traitement d'image, cette équation a été utilisée, *e.g.*, pour représenter des formes [GGS<sup>+</sup>06]. Dans

ce cas, le domaine est une grille en deux dimensions et résoudre cette équation peut être interprété comme le temps moyen que mettrait un marcheur aléatoire pour aller jusqu'au bord  $\partial\Omega$ , en partant d'un point  $x$  de  $\Omega$ .

On peut aussi considérer la  $p$ -équation de Poisson, qui est une généralisation naturelle de l'équation (5.1) :

$$\begin{cases} \Delta_p u(x) = -1 & , x \in \Omega, \\ u(x) = 0 & , x \in \partial\Omega, \end{cases} \quad (5.2)$$

où  $\Delta_p$  est le  $p$ -Laplacien. On peut remarquer que lorsque  $p = 2$ , on retrouve l'équation de Poisson. À mesure que  $p \rightarrow \infty$ , il a été montré que  $u(x) \rightarrow d(x)$  [Kaw90], ce qui correspond à l'équation suivante :

$$\begin{cases} \Delta_\infty u(x) = -1 & , x \in \Omega, \\ u(x) = 0 & , x \in \partial\Omega, \end{cases} \quad (5.3)$$

où  $\Delta_\infty$  est le Laplacien infini.

On peut faire le lien entre cette équation et le jeu du Tug-of-War dans un espace métrique. Nous ne rappelons pas le principe du Tug-of-War, énoncé au chapitre 3. Nous rappelons cependant la fonction de gain du jeu, qui correspond dans ce contexte à  $g(x_k) + \varepsilon^2 \sum_{j=0}^{k-1} h(x_j)$ , où  $x_k$  correspond à la position du jeton au tour  $k$ . Si les deux joueurs utilisent une stratégie optimale, selon le principe de programmation dynamique, la fonction de valeur du jeu satisfait la relation :

$$\begin{cases} u^\varepsilon(x) = \frac{1}{2} \left[ \sup_{y \in B_\varepsilon(x)} u^\varepsilon(y) + \inf_{y \in B_\varepsilon(x)} u^\varepsilon(y) \right] + \varepsilon^2 h(x), & x \in \Omega, \\ u(x) = g(x), & x \in \partial\Omega. \end{cases} \quad (5.4)$$

Les auteurs de [PSSW09] ont montré que quand  $h$  est nulle ou ne change pas de signe, la fonction de valeur du jeu  $u^\varepsilon$  converge vers la solution de l'équation de Poisson infini normalisée :

$$\begin{cases} -\Delta_\infty^N u(x) = h(x) & , x \in \Omega, \\ u(x) = g(x) & , x \in \partial\Omega, \end{cases} \quad (5.5)$$

où  $\Delta_\infty^N u = |\nabla u|^{-2} \Delta_\infty$  est le Laplacien infini normalisé.

Nous pouvons aussi considérer le cas où le jeu est biaisé, tel que nous l'avons fait à la section 4.3.3 : Nous considérons deux réels fixés  $\alpha > 0$ ,  $\beta > 0$  et  $\alpha + \beta = 1$ . Nous pouvons ajouter un biais au jeu du Tug-of-War en utilisant

la même dynamique, en fixant la probabilité de choisir  $x_k^I$  comme  $\alpha$  et  $x_k^{II}$  comme  $\beta$ . Quand le jeu est optimal, sa fonction de valeur satisfait [PPS10] :

$$\begin{cases} u^\varepsilon(x) = \alpha \sup_{y \in B_\varepsilon(x)} u^\varepsilon(y) + \beta \inf_{y \in B_\varepsilon(x)} u^\varepsilon(y) + \varepsilon^2 h(x), & x \in \Omega, \\ u(x) = g(x), & x \in \partial\Omega. \end{cases} \quad (5.6)$$

Cette fonction de valeur est relative au Laplacien infini avec termes de gradients :  $-\Delta_\infty u(x) + c|\nabla u| = h$ , où  $c$  dépend des valeurs des coefficients  $\alpha$  et  $\beta$ . Ce type d'EDP ainsi que le jeu qui y est lié ont été étudiés par les auteurs de [PPS10].

## 5.3 Équation de Poisson infini non-locale avec termes de gradients

Dans cette section, nous réécrivons la fonction de valeur 5.4 dans le contexte des EdPs sur graphes, afin de montrer le lien entre l'équation de Poisson infini sur graphe et le Tug-of-War.

### 5.3.1 Du Tug-of-War à l'équation de Poisson sur graphe

Soit le graphe Euclidien  $G = (V, E, w_6)$  avec  $V = \Omega \subset \mathbb{R}^m$ ,  $E = \{(x, y) \in \Omega \times \Omega \mid w_6(x, y) > 0\}$  et

$$w_6(x, y) = \begin{cases} \frac{1}{\varepsilon^4}, & \text{si } |y - x| \leq \varepsilon, \\ 0, & \text{sinon.} \end{cases}$$

En utilisant  $w_6$  dans la formulation de la norme infini du gradient externe (2.37), nous obtenons :

$$\begin{aligned} \|\nabla_{w_6}^+(f)(x)\|_\infty &= \max_{y \in B_\varepsilon} (\sqrt{w_6(x, y)}(f(y) - f(x))) \\ &= \frac{1}{\varepsilon^2} (\max_{y \in B_\varepsilon} (f(y)) - f(x)) \end{aligned}$$

En appliquant la même simplification à  $\|\nabla_{w_6}^-(f)(x)\|_\infty$ , nous obtenons :

$$\|\nabla_{w_6}^-(f)(x)\|_\infty = \frac{1}{\varepsilon^2} (f(x) - \min_{y \in B_\varepsilon} (f(y))).$$

À l'image des opérateurs max et min redéfinis sur graphe (3.43) à l'aide de la norme infini des gradients directionnels, nous obtenons les expressions suivantes pour les opérateurs sup et inf :

$$\sup_{y \in B_\varepsilon(x)} f(y) = \varepsilon^2 \|\nabla_{w_6}^+(f)(x)\|_\infty + f(x),$$

et

$$\inf_{y \in B_\varepsilon(x)} f(y) = f(x) - \varepsilon^2 \|\nabla_{w_6}^-(f)(x)\|_\infty.$$

En les remplaçant dans l'équation (5.4), nous obtenons :

$$f(x) = \frac{\varepsilon^2}{2} \left[ \|\nabla_{w_6}^+(f)(x)\|_\infty - \|\nabla_{w_6}^-(f)(x)\|_\infty \right] + f(x) + \varepsilon^2 h(x),$$

qui peut être simplifié tel que :

$$\Delta_{w_6, \infty}(f)(x) = -h(x), \quad (5.7)$$

qui correspond à l'équation de Poisson infini discrète. En utilisant une fonction de poids générale, nous obtenons une équation de Poisson infini générale sur graphe :

$$\Delta_{w, \infty}(f)(u) = -h(u),$$

En utilisant l'opérateur  $\mathcal{L}_{w, \infty}$  défini au chapitre précédent (équation (4.8)), nous pouvons transcrire la fonction de valeur du Tug-of-War biaisé (5.6) telle que :

$$\alpha(x) \|\nabla_w^+(f)(x)\|_\infty - \beta(x) \|\nabla_w^-(f)(x)\|_\infty + h(x) = 0, \quad (5.8)$$

avec  $\alpha, \beta : \Omega \rightarrow [0, 1]$  et  $\alpha(x) + \beta(x) = 1$ .

Nous considérons l'équation de Poisson sur graphe suivante, avec des conditions aux bords de Dirichlet :

$$\begin{cases} -\mathcal{L}_{w, \infty}(f)(u) = h(u) & u \in A \\ f(u) = g(u) & u \in \partial^+ A, \end{cases} \quad (5.9)$$

où  $A$  est un ensemble de sommets connexes et  $\partial^+ A$  sa frontière externe.

Comme au chapitre précédent, en utilisant différentes valeurs pour  $\alpha$  et  $\beta$  dans  $\mathcal{L}_{w, \infty}(f)(u)$ , nous retrouvons différentes versions de l'équation (5.9). Dans ce chapitre, nous nous intéressons en particulier à trois d'entre elles :

- Dans le cas  $\alpha = \beta \neq 0$ , l'équation (5.9) devient

$$-\Delta_{w,\infty}(f)(u) = h(u), \quad (5.10)$$

recouvrant ainsi l'équation de Poisson infini sur graphes.

- Afin de faciliter la lecture, nous rappelons ici le cas  $\alpha - \beta < 0$ , ou l'expression de (4.8) devient

$$\begin{aligned} \mathcal{L}_{w,\infty}(f)(u) &= 2\alpha\Delta_{w,\infty}(f)(u) \\ &\quad - (\beta - \alpha)\|\nabla_w^-(f)(u)\|_\infty. \end{aligned} \quad (5.11)$$

L'équation (5.9) devient maintenant :

$$-2\alpha\Delta_{w,\infty}(f)(u) + (\beta - \alpha)\|\nabla_w^-(f)(u)\|_\infty = h(u) \quad (5.12)$$

- Dans le cas  $\beta = 1$ , l'opérateur (4.8) correspond à l'opérateur utilisé dans la transcription de la dilatation sur graphe (2.46). L'équation (5.9) devient alors :

$$\|\nabla_w^-(f)(u)\|_\infty = h(u). \quad (5.13)$$

Ce qui correspond à l'équation eikonale sur graphe [DEL13] en utilisant la norme  $\mathcal{L}_\infty$ , où  $h$  joue le rôle du potentiel  $P(u)$  et  $f$  du temps d'arrivée d'un front se propageant sur le graphe.

En fixant  $h(u) = -1$  et  $g(u) = 0$ , cette formulation recouvre l'équation eikonale suivante :

$$\begin{cases} \|\nabla_w^-(f)(u)\|_\infty = 1, & u \in A \\ f(u) = 0, & u \in \partial^+ A. \end{cases} \quad (5.14)$$

En utilisant différentes fonctions d'interactions (définissant la fonction de poids du graphe), nous pouvons calculer des distances généralisées, de tout sommet dans  $A$  à  $\partial A$ . Cette équation eikonale est un cas particulier d'une équation plus générale sur graphe pondéré :

$$\begin{cases} \|\nabla_w^-(f)(u)\|_p = P(u), & u \in A \\ f(u) = 0, & u \in \partial^+ A. \end{cases} \quad (5.15)$$

Cette famille d'équation a été étudiée par les auteurs de [DEL13]. En particulier, pour  $p = 2$ , il a été montré que, sur des graphes représentant des grilles, cette équation correspond au schéma de discrétisation d'Osher-Sethian [OS88].

### 5.3.2 Calcul de distance généralisé sur graphe

Comme pour le cas du marcheur aléatoire pour  $\Delta_2(f)(u)$ , nous nous intéressons ici au cas où  $h(u) = 1$  et  $g(u) = 0$  à l'équation (5.9). Nous obtenons alors l'équation suivante :

$$\begin{cases} \mathcal{L}_{w,\infty}(f)(u) = -1 & u \in A \\ f(u) = 0 & u \in \partial^+ A. \end{cases} \quad (5.16)$$

Afin de résoudre cette équation, nous la simplifions d'abord telle que  $\mathcal{L}_{w,\infty}f(u) + 1 = 0$  afin d'utiliser le schéma dynamique suivant :

$$\begin{cases} \frac{\partial f(u,t)}{\partial t} = \mathcal{L}_{w,\infty}f(u,t) + 1 & u \in A \\ f(u,t) = 0 & u \in \partial^+ A \\ f(u,t=0) = 0 & u \in A. \end{cases} \quad (5.17)$$

Afin de discrétiser la variable temporelle, nous utilisons la méthode explicite de discrétisation d'Euler :  $\frac{\partial f(u,t)}{\partial t} = \frac{f^{n+1}(u) - f^n(u)}{\Delta t}$ , où  $f^n(u) = f(u, n\Delta t)$ , afin d'obtenir le schéma itératif suivant :

$$\begin{cases} f^{n+1}(u) = f^n(u) + \Delta t(\mathcal{L}_{w,\infty}f(u,t) + 1) & u \in A \\ f^{n+1}(u) = 0 & u \in \partial^+ A \\ f^0(u) = 0 & u \in A. \end{cases} \quad (5.18)$$

Ce schéma itératif peut être interprété comme un schéma morphologique impliquant des filtres de type morphologique : en fixant  $\Delta t = 1$  et en réécrivant le schéma itératif (5.18) en utilisant les opérateurs  $NLD_\infty$  et  $NLE_\infty$  définis aux équations (2.63) et (2.64), nous obtenons l'algorithme itératif suivant :

$$\begin{cases} f^{n+1}(u) = \alpha NLD_\infty(f)(u) + \beta NLE_\infty(f)(u) + 1 & u \in A \\ f^{n+1}(u) = 0 & u \in \partial^+ A \\ f^0(u) = 0 & u \in A. \end{cases} \quad (5.19)$$

## 5.4 Applications

### 5.4.1 Distances géodésiques pondérées

#### Image synthétique

Afin d'illustrer le comportement de la résolution de l'équation de Poisson infini que nous proposons, nous avons expérimenté notre algorithme sur une image synthétique (première image de la figure 5.1). Les résultats ont été obtenus en utilisant un graphe de 8-connexité (défini en section 1.2.2), en utilisant deux fonctions de poids différentes et en faisant varier les valeurs de  $\alpha$  et  $\beta$ . L'image originale est de taille  $400 \times 400$  en niveaux de gris et la fonction de distance est calculée depuis le coin haut gauche de l'image. La première ligne montre les cartes de distances avec des lignes de niveaux. Ces distances ont été obtenues en utilisant la fonction de similarité  $s_1$  ( $w(u, v) = 1$ , définie à l'équation (1.15)). On peut remarquer qu'en utilisant  $\alpha = 0$ , la distance entre deux lignes de niveau est constante, décrivant une fonction de distance linéaire. On peut aussi observer que la distance calculée est anisotrope, donnant plus d'importance aux directions diagonales (ceci étant dû à la norme  $\mathcal{L}^\infty$ ). À mesure que le paramètre  $\alpha$  varie de 0 à 0.5, la distance calculée évolue d'une forme linéaire à une en racine carrée. La seconde ligne de la figure 5.1 illustre les effets en utilisant la fonction de similarité Gaussienne  $s_3$  (définie à l'équation (1.17)), où la fonction  $d$  utilisée est la distance Euclidienne  $d_2$  (1.11) entre les intensités de pixels de l'image, avec  $\sigma = 150$ . L'information de forme est représentée naturellement par cette fonction de poids, augmentant la valeur de la distance calculée lorsque la fonction atteint le bord d'un objet.

#### Formes

Nous avons aussi expérimenté notre algorithme sur une image de forme (figure 5.2). Le graphe est construit de la même façon que pour l'illustration sur l'image synthétique, mais nous avons utilisé ici la fonction de similarité inverse  $s_2$  (1.16), avec  $d$  la distance Euclidienne entre les coordonnées de pixels. Pour cette illustration, la distance est calculée à partir des frontières de la forme. Comme on peut le remarquer, en faisant varier le paramètre  $\alpha$ , on observe le même phénomène que pour l'image synthétique, la fonction évoluant d'une forme linéaire à une forme en racine carrée.



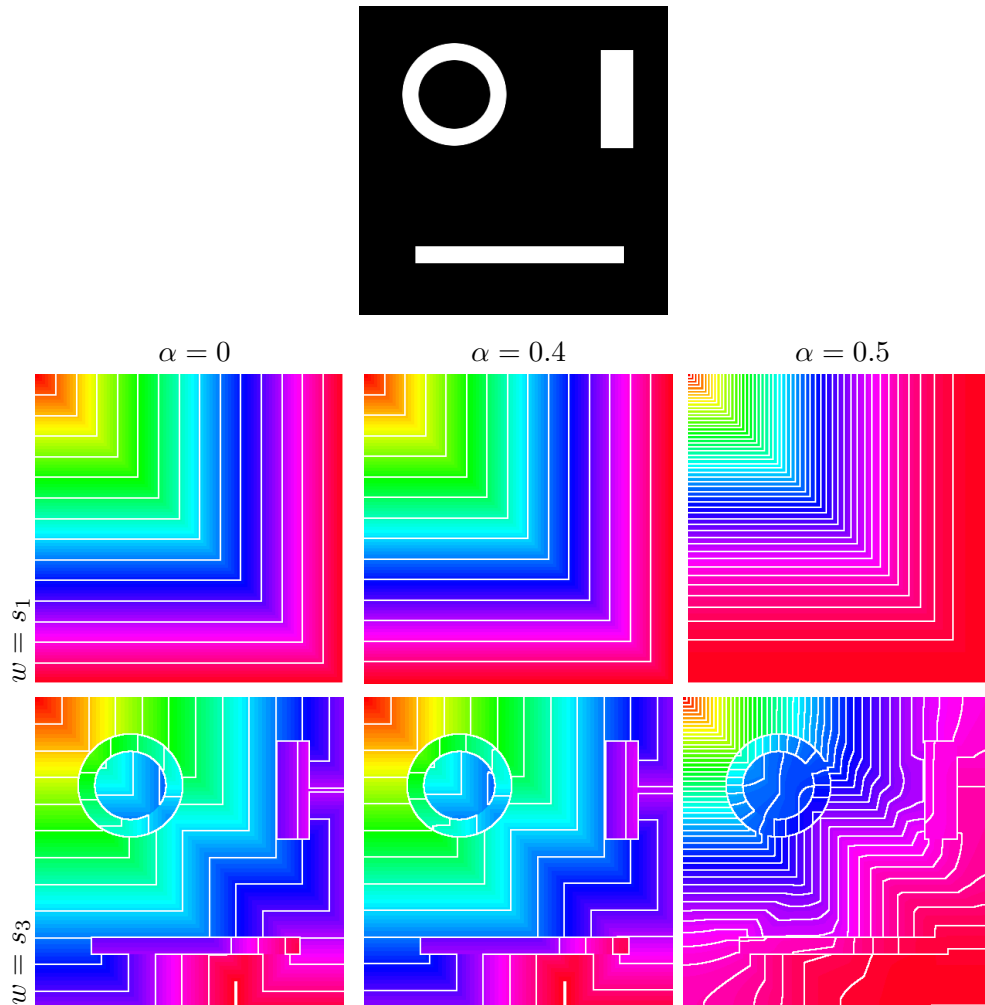


FIGURE 5.1 – Calcul de distance sur une image synthétique, en utilisant différentes valeurs pour les paramètres  $\alpha$  et  $\beta$  et différentes fonctions de poids. La distance est calculée à partir du coin haut gauche de l'image.

### Image naturelle

Afin d'illustrer les effets de la fonction de poids, nous avons éprouvé notre algorithme sur une image naturelle (image du haut de la figure 5.3).

Sur la première, deuxième et troisième ligne, comme pour l'expérimentation sur l'image de forme, nous avons utilisé la fonction de similarité inverse  $s_2$  (1.16), avec  $d$  la distance Euclidienne entre les coordonnées des pixels.

La première ligne montre la distance calculée en utilisant un graphe de 4-connexité, la deuxième ligne en utilisant un graphe de 8-connexité et la

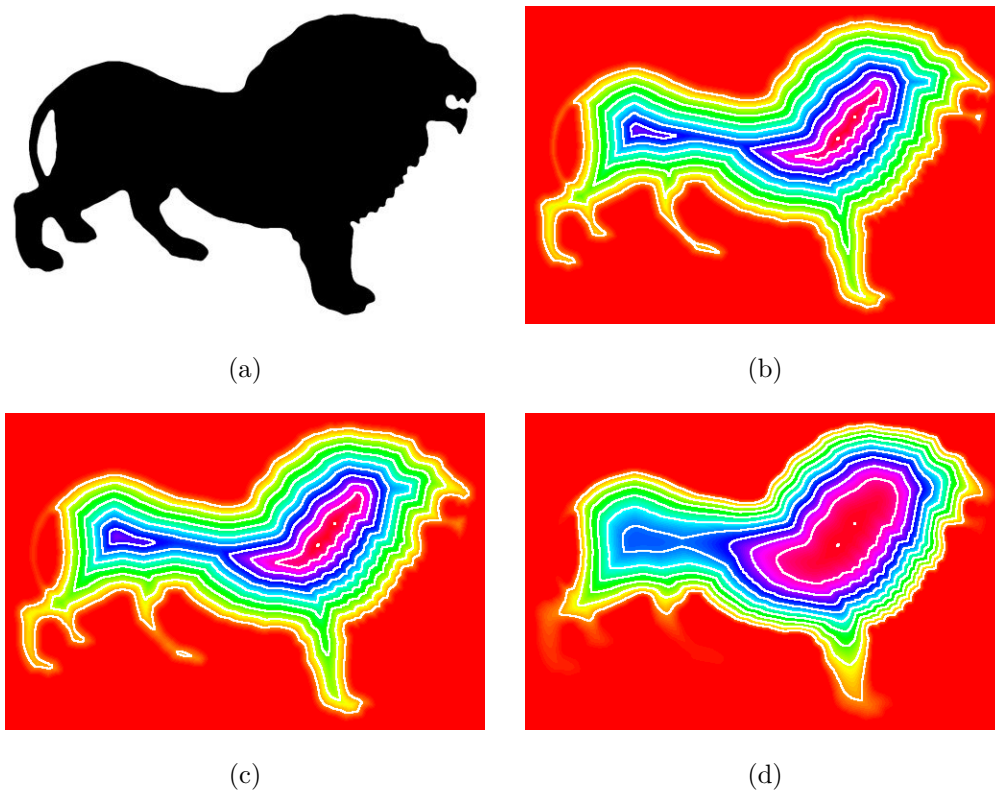


FIGURE 5.2 – Génération de distance en utilisant différentes valeurs pour le paramètre  $\alpha$  : la distance à la figure (b) est calculée avec  $\alpha = 0$ , à la figure (c) avec  $\alpha = 0.4$  et à la figure (d) avec  $\alpha = 0.5$ . La distance est calculée à partir des frontières de la forme.

troisième en utilisant un  $\varepsilon$ -voisinage avec  $\varepsilon = 3$  en utilisant une distance de Chebyshev, *i.e.* chaque sommet du graphe est connecté à tous les points autour de lui dans une fenêtre carrée de taille  $7 \times 7$ .

On peut remarquer qu'en ajoutant des voisins à un sommet, la distance calculée devient "moins anisotrope". En réalité, cette distance est toujours anisotrope, mais dans la direction des voisins. Au fur et à mesure que nous ajoutons des voisins à chaque sommet, il y a de plus en plus de directions prises en compte. Afin d'obtenir une distance isotrope, une solution serait d'ajouter une infinité de voisins dans une infinité de directions.

La quatrième ligne de la figure 5.3 montre la distance calculée en utilisant un graphe de 8-connexité, en utilisant  $s_3$  comme fonction de similarité et  $d = d_2$  la distance entre les couleurs des pixels (considérés comme des vecteurs). Comme dans le cas de l'image synthétique, on peut voir que cette fonction de poids

permet de faire ressortir les formes en fonction des différences d'intensités des pixels. Sur la cinquième ligne, nous avons construit un graphe des  $k$ -ppv (voir section 1.2.1) dans l'espace des patches de l'image. Nous avons choisi (arbitrairement)  $k = 5$ , en restreignant la fenêtre de recherche des plus proches voisins à une taille de  $15 \times 15$  pixels, centrée autour de chaque pixel. Nous avons utilisé des patches de tailles  $5 \times 5$  pixels, centrés aussi autour du pixel correspondant. Les patches sont représentés par une fonction d'attribut  $\mathcal{F}_W$  telle que définie en section 1.2.2. La métrique utilisée pour définir la proximité entre les patches est la distance Euclidienne. Afin de calculer la similarité entre patches, nous avons utilisé la similarité Gaussienne  $s_3$  avec  $d$  la somme des distances Euclidiennes normalisée entre les pixels des patches de la paire de sommets considérée. Comme on peut le voir, les objets sont mieux détournés, montrant de large zone où la fonction évolue lentement à l'intérieur des objets et une évolution rapide à la frontière d'un objet.

### Nuages de points 3D

Afin de montrer l'adaptativité de notre approche, nous illustrons le calcul de la distance généralisée sur différents nuages de points 3D aux figures 5.4 et 5.5. Nous avons construit un graphe des  $k$ -ppv avec  $k = 5$ , dans l'espace des coordonnées du nuage de point. Le pas de discrétisation spatial étant assez régulier sur ces données, nous avons utilisé une fonction de poids constante ( $w(u, v) = 1$ ). La ligne rouge en surimpression sur chaque figure représente le chemin le plus court entre le point source (le point à partir duquel la distance est calculée) et un autre point dans le nuage de point. Ce chemin est calculé à partir de la fonction de distance générée. Nous avons comparé notre algorithme avec l'adaptation de l'équation eikonale sur graphe [DEL13] (première colonne de la figure 5.4 et première ligne de la figure 5.5) en utilisant la norme  $\mathcal{L}^2$  du gradient.

#### 5.4.2 Segmentation et classification de données

Dans cette section, nous présentons le comportement de notre algorithme pour les tâches de segmentation d'image et de classification semi-supervisée. Nous l'illustrons en utilisant des configurations locales et non-locales sur différents types de données, à travers plusieurs exemples. Dans le cas de la segmentation d'image, plusieurs approches sont devenues populaires, telles que les graph cuts [BJ01], les marches aléatoires [Gra06], les plus courts chemins, le watershed ou encore des approches unifiant ces méthodes, telle que le powerwatershed [CGNT11, SG07].



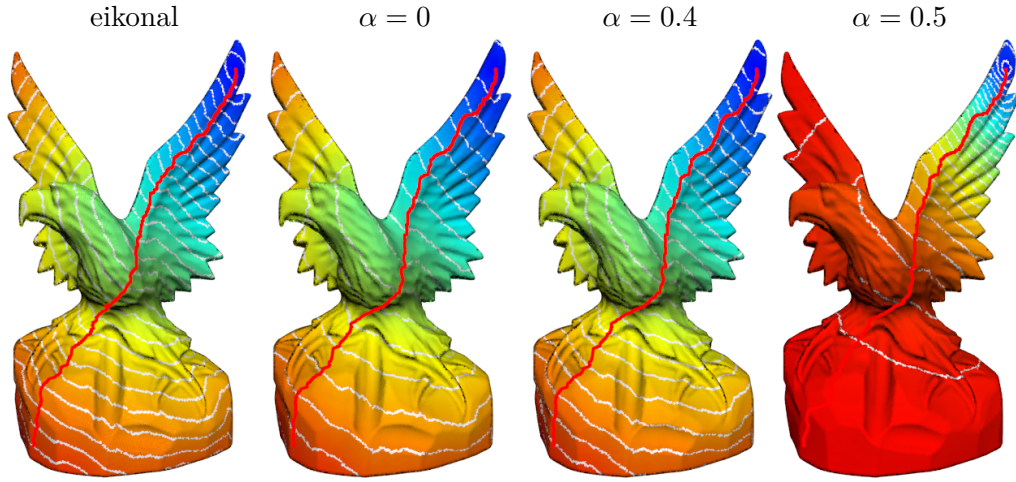


FIGURE 5.4 – Génération de distance sur un nuage de point 3D, en utilisant différentes valeurs pour le paramètre  $\alpha$  et un graphe des  $k$ -ppv. Voir le texte pour plus de détails.

semi-supervisée. On peut voir cette tâche comme un processus de diffusion de germes, ou marqueurs initiaux (marqueurs placés initialement sur l'image, renseignant sur la classe d'appartenance des points marqués). Afin d'adapter notre algorithme à la diffusion de marqueurs, nous le réécrivons comme suit. Soit  $V = \{u_1, \dots, u_n\}$  un ensemble fini de données, où chaque donnée  $u_i$  est un vecteur de  $\mathbb{R}^m$ . Soit  $G = (V, E, w)$  un graphe pondéré tel que l'ensemble des sommets représente les données.

La segmentation semi-supervisée de  $V$  consiste à partitionner l'ensemble  $V$  en  $k$  classes ( $k$  étant connu à l'avance) étant donné un sous ensemble de sommets de  $V$  initialement marqués. Le but est ensuite d'estimer la classe des données non marquées à partir de celles qui sont marquées. Soit  $C_l$  l'ensemble des sommets marqués appartenant à la classe  $l$ . Soient  $V_0 = \cup\{C_l\}_{l=1, \dots, k}$  l'ensemble des sommets *initialement marqués* et  $V \setminus V_0$  l'ensemble des sommets *initialement non marqués*. On peut ensuite étendre les marqueurs à l'ensemble des sommets en calculant  $k$  distances indépendantes, à partir des ensembles de sommets  $C_l$  initialement marqués correspondants :

$$\begin{cases} \mathcal{L}_{w, \infty} f_l(u) = -1 & u \in V \setminus V_0 \\ f_l(u) = 0 & u \in C_l. \end{cases} \quad (5.20)$$

Une fois les distances calculées à partir de chaque classe, l'appartenance d'un sommet  $u$  à une classe est donnée comme l'identifiant de la plus petite fonction de distance :  $\arg \min_{l \in 1, \dots, k} f_l(u)$ .

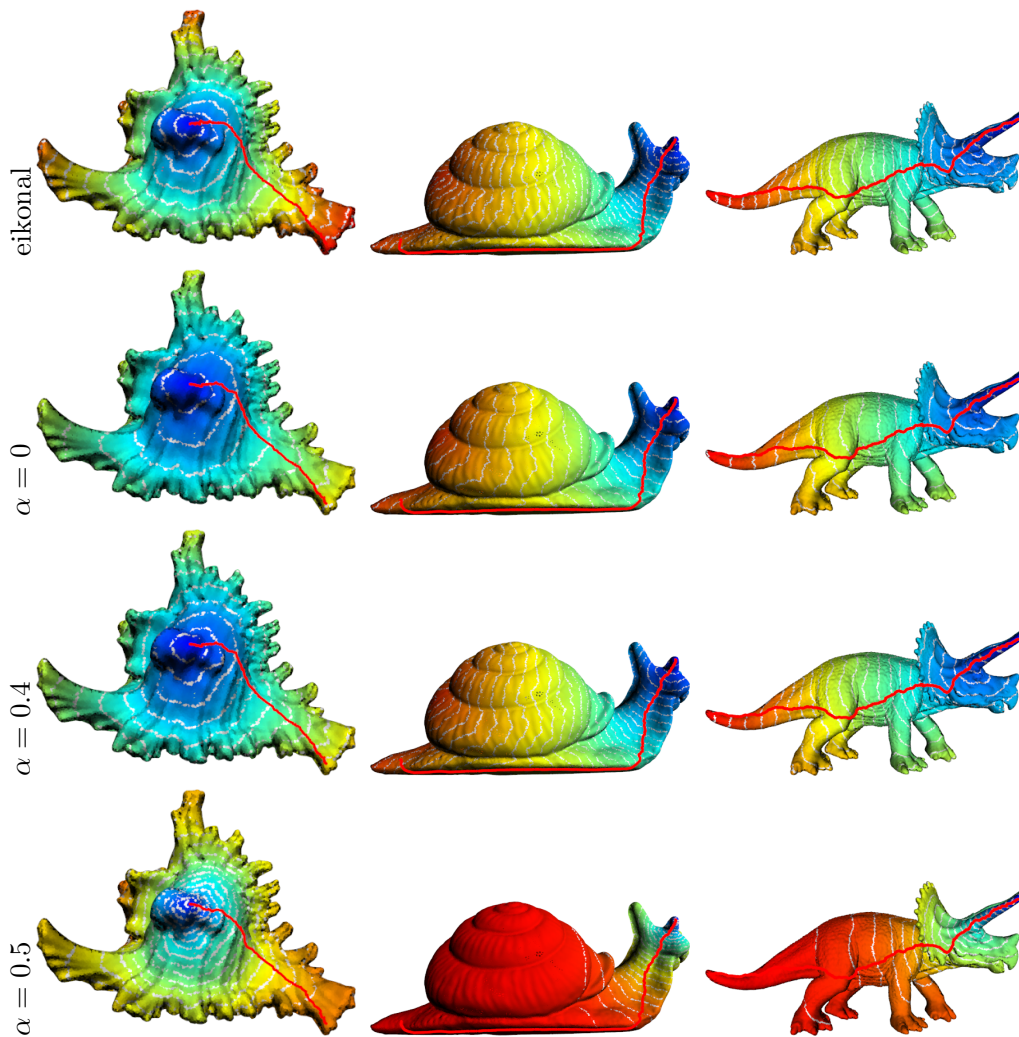


FIGURE 5.5 – Génération de distance sur des nuages de point 3D, en utilisant différentes valeurs pour le paramètre  $\alpha$  et un graphe des  $k$ -ppv. Voir le texte pour plus de détails.

### Segmentation d'images

Nous illustrons cette méthode sur une image à la figure 5.6. La première image (à gauche) est l'image initiale avec les germes dessinés par un utilisateur. Pour la deuxième image, nous avons construit un graphe de 8-connexité, en utilisant la similarité couleur ( $s_3$ ) comme fonction de poids du graphe. Pour la troisième image, nous avons construit un graphe non local des  $k$ -ppv, dans l'espace des patches de l'image, de la même manière que pour l'illustration du calcul de distance à la figure 5.3. Nous avons aussi illustré les effets de l'utilisation d'une





FIGURE 5.6 – Segmentation d’image en utilisant une construction locale et non-locale de graphe. Voir le texte pour plus de détails.

construction non-locale de graphe en utilisant la similarité entre patches sur une image de texture à la figure 5.7. Comme pour la figure 5.6, la première image est l’image initiale avec les marqueurs initiaux en surimpression. Dans la colonne de gauche, nous affichons le minimum des distances générées, sur celle de droite la segmentation correspondante. La première ligne montre les résultats obtenus avec une construction de graphe locale (graphe de 8-connectivité). La dernière ligne montre les résultats en utilisant un graphe des  $k$ -ppv avec une fenêtre de recherche de taille  $15 \times 15$  et des patches de taille  $9 \times 9$ . Ces résultats montrent le bénéfice de l’utilisation de configurations non-locales en utilisant des patches, particulièrement pour des images texturées, où les méthodes classiques donnent de moins bons résultats dans le détournement des objets désirés.

Afin d’évaluer quantitativement la méthode proposée, nous avons utilisé la base de données Grabcut de Microsoft [RKB04], disponible en ligne. Nous avons repris les résultats des évaluations effectuées par les auteurs de [CGNT11], ou ils comparent leur algorithme à ceux précédemment cités : les graph cuts [BJ01], les marches aléatoires [Gra06], les plus courts chemins, et les forêts couvrantes maximales (MSF). Nous avons évalué notre algorithme en quantifiant les erreurs de segmentation en utilisant des mesures utilisées dans [CGNT11], *i.e.* la Boundary Error (BE), le Rand Index (RI) et la Global Consistency Error (GCE). La Boundary Error (erreur de frontière en français) entre deux segmentations mesure la distance moyenne entre les pixels de frontières dans la première image et les plus proches dans la deuxième. Le Rand Index compte la proportion de paires de pixels dont les classes sont similaires entre le résultat de segmentation et la vérité terrain. Les valeurs

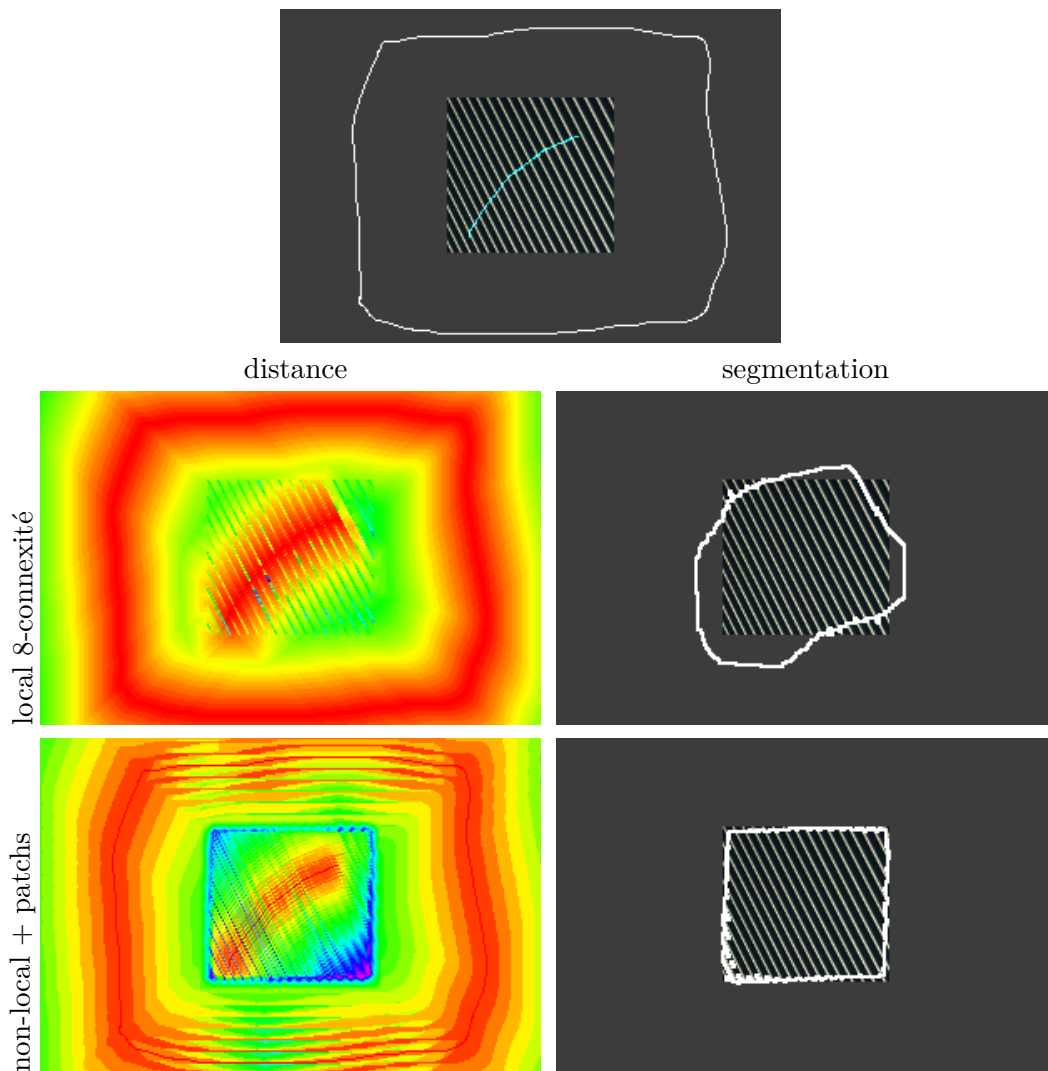


FIGURE 5.7 – Segmentation d’image texturée. La première image est l’image initiale avec les marqueurs initiaux en surimpression. La première ligne de résultat a été générée en utilisant un graphe de 8-connextité, la fonction de similarité utilisée est la similarité couleur. La deuxième ligne a été générée en utilisant un graphe des  $k$ -ppv avec  $k = 15$  dans l’espace des patches de l’image. Voir le texte pour plus de détails.

de cette mesure sont comprises dans l’intervalle  $[0, 1]$ . La Global Consistency Error mesure dans quelle proportion une segmentation peut être vue comme le raffinement d’une autre. Une bonne segmentation est caractérisée par une BE et une GCE aussi petite que possible et un RI aussi proche de 1 que possible. Pour cette expérimentation, nous avons construit un graphe de 8-connextité,



TABLE 5.1 – Évaluation sur la base Grabcut

	BE	RI	GCE
Shortest paths	2.82	0.972	0.0233
Random walker	2.96	0.971	0.0234
MSF	2.89	0.971	0.0244
Power wshed [CGNT11]	2.87	0.971	0.0245
Graph cuts	3.12	0.970	0.0249
$\infty$ -Poisson ( $\alpha = 0$ )	1.25	<b>0.977</b>	<b>0.0198</b>
$\infty$ -Poisson ( $\alpha = 0.4$ )	1.23	<b>0.977</b>	<b>0.0198</b>
$\infty$ -Poisson ( $\alpha = 0.48$ )	<b>1.21</b>	<b>0.977</b>	0.02

avec  $w(u, v) = s_3(u, v)$ . Les résultats de cette évaluation sont montrés au tableau 5.1. Comme on peut le remarquer, notre algorithme est légèrement meilleur sur chaque mesure d'erreur, désignant une meilleure segmentation sur ce jeu de données.

### Classification de données

Nous avons aussi expérimenté notre méthode dans le cadre de la classification de données sur un échantillon de la base de données MNIST [LC10], illustrée par la figure 5.8, afin de montrer l'adaptativité de l'algorithme proposé pour la classification de données non organisées de grandes dimensions. La méthode est similaire à celle employée pour la segmentation d'image et peut être adaptée directement. Pour construire le graphe, chaque sommet représente un objet de la base de données. Dans le cas de cette base de données, qui est composée d'images de chiffres manuscrits, nous représentons chaque objet par son vecteur de pixels correspondant. Pour calculer la similarité entre les objets, nous avons utilisé la fonction de similarité  $s_3$ , avec  $d$  la distance Euclidienne entre chaque vecteur de pixels. Nous avons ensuite construit un graphe des  $k$ -ppv en utilisant la même distance. Ce graphe est représenté à la figure 5.8(a). Pour obtenir la classification, deux sommets de chaque classes ont été choisis aléatoirement comme marqueurs initiaux. Le résultat de la classification est montré à la figure 5.8(b).

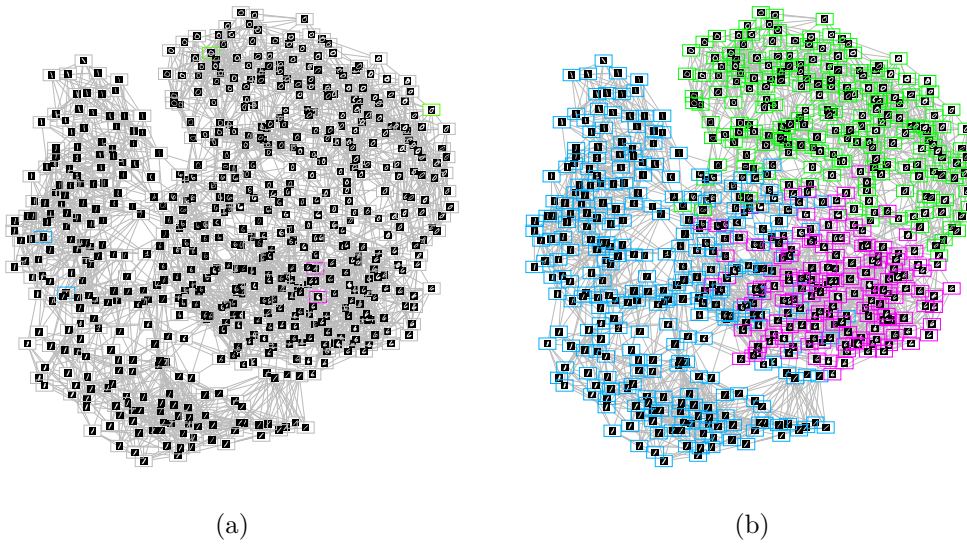


FIGURE 5.8 – Classification d’un échantillon de la base de données MNIST. La figure (a) représente le graphe avec les marqueurs initiaux. La figure (b) représente le résultat de la classification semi-supervisée en utilisant ces marqueurs initiaux.

### 5.4.3 Efficacité du schéma de résolution

Dans cette section nous traitons de l’efficacité en temps de calcul du schéma de résolution proposé. L’objectif principal de ce chapitre étant de proposer une adaptation de l’équation de Poisson infini sur graphe, dans le but de fournir un moyen de calculer des distances sur des graphes de topologies arbitraires, nous ne nous sommes pas concentrés sur la façon optimale de résoudre cette équation et avons utilisé une méthode de discrétisation temporelle simple afin d’obtenir un algorithme itératif. Pour résumer, en faisant varier le paramètre  $\alpha$ , le taux de convergence est superlinéaire en utilisant  $\alpha = 0$  (figure 5.9(a)) et sous linéaire en utilisant  $\alpha = 0.5$ (figure 5.9(c)).

## 5.5 Conclusion

Dans ce chapitre, en montrant le lien entre le jeu du Tug-of-War et l’équation de Poisson infini, nous avons adapté la formulation de cette équation sur graphe en utilisant l’expression du Laplacien infini avec termes de gradients proposé au chapitre précédent.

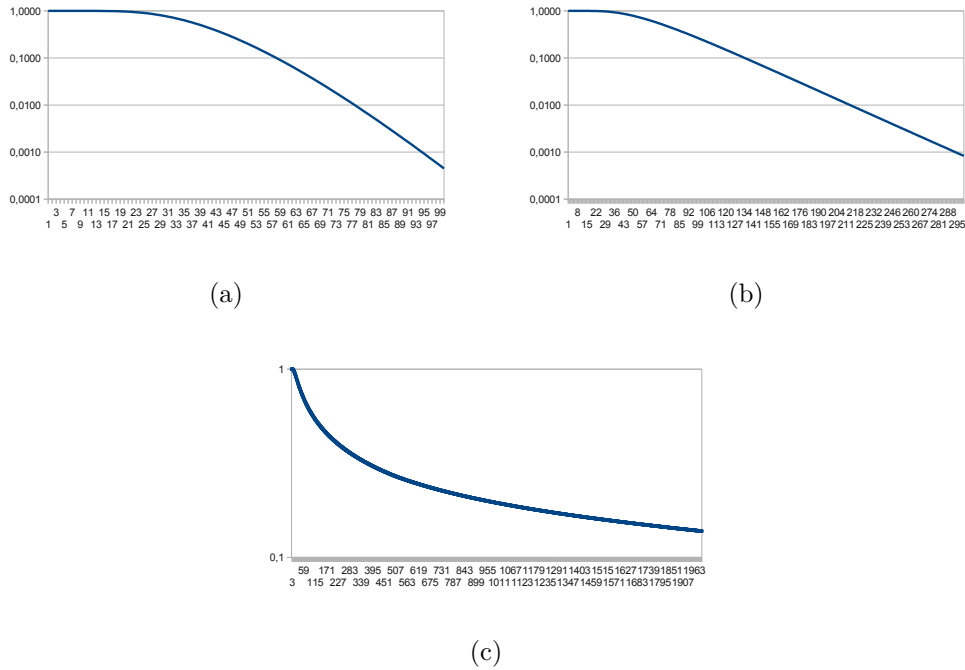


FIGURE 5.9 – Taux de convergence de l’algorithme proposé pour différentes valeurs du paramètre  $\alpha$  ( $\alpha = 0$  à la figure (a),  $0.25$  à la figure (b) et  $0.5$  à la figure (c)). L’axe des abscisses représente le nombre d’itérations de l’algorithme, les ordonnées représentent la convergence.

En utilisant cette formulation pour adapter l’équation de Poisson infini sur graphe, nous avons montré certains cas spéciaux de cette équation en faisant varier les coefficients devant les termes de gradients, montrant le lien entre cette version de l’équation de Poisson et l’équation eikonale. Nous avons utilisé cette extension sur graphe afin de calculer des distances sur des données pouvant être représentées par des graphes : des images, des nuages de points 3D, ainsi que des données non organisées de grande dimensions (pouvant être considérées comme des nuages de points à  $m$ -dimensions). Nous avons aussi montré que cette formulation peut être utilisée dans le cadre de la segmentation d’images et de la classification de données, obtenant ainsi des performances comparables voire supérieures à l’état de l’art.

## Troisième partie

Équations d'ensembles de  
niveaux pour le clustering et la  
classification de données



# Chapitre 6

## Équations d'ensembles de niveaux pour le clustering et la classification de données

### Sommaire

---

<b>6.1</b>	<b>Introduction</b>	<b>144</b>
<b>6.2</b>	<b>EDPs géométriques basées sur les ensembles de niveaux sur graphe</b>	<b>145</b>
6.2.1	Formulation continue	145
6.2.2	Transposition sur graphe	146
6.2.3	Lien avec l'équation eikonale	147
6.2.4	Algorithme de propagation de marqueurs utilisant l'équation eikonale sur graphe	147
<b>6.3</b>	<b>Processus d'évolution multi-classes</b>	<b>150</b>
6.3.1	Équation d'évolution de courbe multi-classes	150
6.3.2	Accélération de l'algorithme	151
<b>6.4</b>	<b>Classification semi-supervisée</b>	<b>156</b>
6.4.1	Méthode d'évaluation	156
6.4.2	Construction des graphes	157
6.4.3	Fonctions de potentiels	158
6.4.4	Influence du paramètre $\sigma$	158
6.4.5	Évaluation et comparaison	159
6.4.6	Temps d'exécutions	161

6.4.7 Applications en microscopie virtuelle . . . . .	163
<b>6.5 Clustering / Classification non-supervisée . . . . .</b>	<b>169</b>
6.5.1 Algorithme proposé . . . . .	169
6.5.2 Évaluation . . . . .	172
<b>6.6 Conclusion . . . . .</b>	<b>175</b>

---

## Publications associées à ce chapitre

- [TEDPar] Matthieu TOUTAIN, Abderrahim ELMOATAZ, Xavier DESQUESNES et Jean-Hugues PRUVOT : A unified geometric model for virtual slide image processing and classification. *Journal of Selected Topics in Signal Processing*, pages 1–11, to appear.
- [TEL14] Matthieu TOUTAIN, Abderrahim ELMOATAZ et Olivier LÉZORAY : Geometric pdes on weighted graphs for semi-supervised classification. *In Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 231–236. IEEE, 2014.
- [TDEL13] Matthieu TOUTAIN, Xavier DESQUESNES, Abderrahim ELMOATAZ et Olivier LEZORAY : A unified geometric model for virtual slide images processing. *In International Workshop on Adaptation and Learning in Control and Signal Processing (IFAC)*, pages 629–634, 2013.

## 6.1 Introduction

Dans ce chapitre, nous proposons tout d’abord une formulation, basée sur la transposition des ensembles de niveaux sur graphe, permettant de faire évoluer plusieurs courbes à la fois sur un graphe. Dans ce cadre, nous décrivons aussi l’algorithme de propagation de front sur graphe, basé sur l’équation eikonale et transposé sur graphe par [DEL13]. À partir de la formulation multi-classes des ensembles de niveaux que nous proposons, nous proposons d’utiliser le schéma de résolution proposé par [TEL11], et proposons un algorithme combinant la propagation de front sur graphe et la formulation proposée, ainsi qu’une implantation GPU, dans le cadre de la classification semi-supervisée, afin d’accélérer les temps de calcul et d’améliorer les résultats en termes de taux de classification.

Dans les chapitres précédents, nous avons proposé plusieurs formulations d'opérateurs de Laplace sur graphe ainsi qu'un algorithme, basé sur le problème de Dirichlet, afin d'effectuer de la propagation de germes sur graphes. Nous avons suggéré l'utilisation de cet algorithme pour la classification semi-supervisée au travers d'illustrations.

Dans ce chapitre nous proposons d'utiliser et d'évaluer ces différents algorithmes ( $p$ -Laplacien normalisé sur graphe,  $p$ -Laplacien sur graphe, propagation de front sur graphe, ensemble de niveaux, combinaison propagation de front / ensemble de niveaux) ainsi que l'algorithme MTV proposé par [BTUvB13] sur différentes bases de données de la littérature. Nous appliquons aussi ces algorithmes à la classification de données dans le cadre de la microscopie virtuelle.

Nous proposons ensuite un algorithme de clustering, inspiré par les récents travaux de [BHL<sup>+</sup>14]. Celui-ci permet d'utiliser des algorithmes de classification transductive dans un cadre non-supervisé. Nous proposons de reprendre son fonctionnement afin d'utiliser les algorithmes de classification semi-supervisée que nous proposons, permettant ainsi d'effectuer du clustering avec ceux-ci. Nous comparons les algorithmes que nous proposons avec l'algorithme INGRES proposé par [BHL<sup>+</sup>14] ainsi qu'à la méthode NMFR proposée par [YHD<sup>+</sup>12], en termes de pureté de cluster, sur vingt bases de données.

## 6.2 EDPs géométriques basées sur les ensembles de niveaux sur graphe

### 6.2.1 Formulation continue

Soit un front  $\Gamma$  évoluant sur un ouvert borné  $\Omega \subset \mathbb{R}^m$ . À un instant  $t \in [0, +\infty[$ , ce front peut être représenté comme une courbe paramétrique  $\gamma(t) : [0, 1] \rightarrow \Omega$ , évoluant sous l'effet d'une fonction de vitesse  $\mathcal{F} : \Omega \rightarrow \mathbb{R}$ . L'approche par ensemble de niveaux, introduite et popularisée par [OS88] est basée sur l'idée de représenter un front  $\Gamma$  implicitement par le niveau zéro d'une fonction d'ensembles de niveaux  $\phi(x, t)$ . En d'autres mots, en considérant un temps  $t$  et en utilisant cette approche, la position d'un front est donnée par l'ensemble des points  $x \in \Omega$  lorsque  $\phi(x, t) = 0$  :

$$\gamma(t) = \{x | \phi(x, t) = 0\} \tag{6.1}$$

La fonction d'ensemble de niveaux est une fonction lisse continue  $\phi : \mathbb{R}^m \times [0, \infty[ \rightarrow \mathbb{R}$  vérifiant les propriétés suivantes





et proposent un schéma itératif général pour calculer  $\phi$  :

$$\phi^{n+1}(u) = \begin{cases} \phi^n(u) + \Delta t \mathcal{F}(u) \|\nabla_w^+(\phi^n)(u)\|_p, \\ \text{si } \mathcal{F}(u) > 0, \\ \phi^n(u) + \Delta t \mathcal{F}(u) \|\nabla_w^-(\phi^n)(u)\|_p, \\ \text{si } \mathcal{F}(u) < 0, \end{cases} \quad (6.6)$$

où  $\phi^n(u) = \phi(u, n\Delta t)$ .

### 6.2.3 Lien avec l'équation eikonale

Dans le cas où le signe de  $\mathcal{F}$  est constant sur le domaine  $\Omega$ , l'équation utilisant les ensembles de niveaux peut être réécrite sous la forme d'une équation stationnaire :

$$\mathcal{F} \|\nabla_w^-(\mathcal{T})(u)\|_p = 1, \quad (6.7)$$

où  $\mathcal{T} : \Omega \rightarrow \mathbb{R}^+$  est la fonction associant chaque point du domaine  $\Omega$  au temps d'arrivée du front en ce point. Le lien entre  $\phi$  et  $\mathcal{T}$  est donné par la relation  $\mathcal{T}(u) = t - \phi(u, t)$ ,  $\forall t \in \mathbb{R}^+$ .

### 6.2.4 Algorithme de propagation de marqueurs utilisant l'équation eikonale sur graphe

Dans ce chapitre, nous proposons d'utiliser cet algorithme pour les tâches de classification semi-supervisée et de clustering. Nous l'introduisons donc ici. Les auteurs de [DEL13] ont proposé un algorithme de propagation de marqueurs sur graphe, basé sur la résolution de l'équation (6.7). La formulation de cette équation sur graphe est exprimée telle que :

$$\begin{cases} \|\nabla_w^-(f)(u)\|_p = P(u), & \forall u \in V, \\ f(u) = 0, & \forall u \in V_0, \end{cases} \quad (6.8)$$

où  $V_0 \subset V$  correspond à un ensemble initial de marqueurs et  $P$  un potentiel, correspondant à l'inverse de la vitesse  $\mathcal{F}$  dans l'équation (6.7). Cette adaptation est exprimée en utilisant l'érosion morphologique basée sur les EdPs et peut être liée avec l'équation géométrique générale (6.5).

Nous rappelons ici l'algorithme proposé par [DEL13] pour la propagation de marqueurs sur graphe, basé sur la résolution de l'équation eikonale sur graphe (6.8). Cet algorithme (algorithme 1) est basé sur la propagation simultanée de

---

**Algorithme 1:** Propagation de front sur graphe

---

**Données :**  $S^0$  : L'ensemble des marqueurs initiaux ;  
 $A$  : L'ensemble des sommets actifs ;  
 $NB$  : File de priorité des sommets de la bande mince, ordonné selon  $f$  ;  
 $FA$  : L'ensemble des sommets lointains ;  
 $lab$  : La fonction de marqueur.

```

1 début
2    $lab(u) = \text{Marqueur initial de } u$  ;
3    $f(u) = 0 \forall u \in S^0; f(u) = +\infty \forall u \in V \setminus S^0$  ;
4    $s(u) = +\infty \forall u \in V \setminus S^0$  ;
5    $A = S^0; NB = \{u \mid \exists v \in A \text{ et } v \in N(u)\}$  ;
6    $FA = V \setminus A \cup NB$  ;
7    $f(u) = \text{solution locale } \forall u \in NB$  ;
8 tant que } FA \neq \emptyset \text{ faire}
9    $u \leftarrow \text{premier élément de } NB$  ;
10  Enlever  $u$  de  $NB$  et ajouter  $u$  à  $A$  ;
11  pour chaque } v \in N(u) \setminus A \text{ faire}
12    Calculer la solution locale  $t \leftarrow f(v)$  ;
13    si } t < f(v) \text{ alors}
14       $f(v) = t$  ;
15      si } v \in FA \text{ alors}
16        Retirer  $v$  de  $FA$  et ajouter  $v$  à  $NB$  ;
17      sinon
18        Mettre à jour la priorité de  $v$  dans  $NB$  ;
19      si } f(u)/w_{uv} < s(v) \text{ alors}
20         $s(v) = f(u)/w_{uv}$  ;
21         $lab(v) = lab(u)$  ;

```

---

plusieurs fronts et marque le sommet atteint par un front en fonction de son marqueur initial.

La propagation est effectuée à partir d'un ensemble de sommets initial dont on connaît les classes (ensemble de marqueurs initiaux,  $S^0$  dans l'algorithme 1), jusqu'à ce que tous les sommets du graphe aient un marqueur. La complexité temporelle de cet algorithme de propagation de front est linéarithmique en termes de nombre de sommets et d'arêtes du graphe :  $\mathcal{O}(|E| + |V|\log(|V|))$ . Il peut être utilisé dans plusieurs applications sur graphes, telles que le calcul de distances géodésiques sur graphe, la segmentation d'image et, dans notre

cas, la classification semi-supervisée ainsi que le clustering. La solution locale  $t$  (ligne 11 de l'algorithme 1) est calculée en utilisant les équations (2.36) et (6.8) :

$$\left( \sum_{v \sim u} \sqrt{(w_{uv})^p} \max(0, (f(u) - f(v)))^p \right)^{\frac{1}{p}} = P(u). \quad (6.9)$$

Les auteurs de [DEL13] ont montré que cette équation possède une solution unique. Dans le cas où  $p = 1$ , la solution  $\bar{x}$  en un sommet  $u$  est caractérisée par :

$$\bar{x} = \frac{\sum_{i=1}^n h_i a_i + C}{\sum_{i=1}^n h_i}, \quad (6.10)$$

où on considère l'ensemble des voisins de  $u$  :  $\{v_1, v_2, \dots, v_n\}$ , avec  $n = |v \sim u|$ ,  $\bar{x} = f(u)$ ,  $C = P(u)$  et pour tout entier  $i$  compris entre 1 et  $n$ ,  $a_i = f(v_i)$ ,  $h_i = 1/\sqrt{w_{uv_i}}$ .

De même, dans le cas où  $p = 2$ , la solution de l'équation (6.9) est caractérisée par :

$$\bar{x} = \frac{\sum_{i=1}^n h_i^2 a_i + \sqrt{\sum_{i=1}^n (h_i^2 C^2 - \sum_{j>i} h_i^2 h_j^2 (a_i - a_j)^2)}}{\sum_{i=1}^n h_i^2}. \quad (6.11)$$

Enfin, pour le cas  $p = \infty$ , la solution est caractérisée par :

$$\bar{x} = \min_{i=1}^n \left( a_i + \frac{C}{h_i} \right). \quad (6.12)$$

Largement utilisée en traitement d'image, nous proposons d'utiliser cette formulation pour la classification de données. Classiquement, pour une image  $I : V \rightarrow \mathbb{R}^c$ , les fonctions de potentiel utilisées en traitement d'image sont : un potentiel constant ( $P = 1$ ), ou encore la norme du gradient ( $P(u) = \|\nabla(I)(u)\|_2$ ). Pour le cas de la classification, nous avons adapté la fonction de potentiel afin d'obtenir des clusters compacts, contenant des éléments proches les uns des autres (la distance calculée en utilisant l'équation eikonale sur graphe doit être petite) et séparant les éléments qui ne sont pas proches (la distance calculée doit être grande). Afin d'atteindre ces exigences, nous avons utilisé un potentiel qui mesure la distance à la moyenne des éléments d'un cluster. Soit  $F : V \rightarrow \mathbb{R}^c$  le descripteur associé à chaque point des données représentées par le graphe. Nous proposons d'utiliser le potentiel  $P$  défini tel que  $P(u) = \|F(u) - \mathcal{C}_l\|$ , où  $\mathcal{C}_l$  est la moyenne des éléments à l'intérieur du cluster  $l$ , autrement dit les éléments à l'intérieur du front  $\Gamma_l$ . La moyenne

d'un cluster est mise à jour chaque fois qu'un noeud lui est ajouté. Dans la section 6.4, nous utilisons ces trois fonctions de potentiel afin de réaliser nos expérimentations. Celles-ci seront détaillées et comparées en termes de taux de classification.

## 6.3 Processus d'évolution multi-classes

Dans cette section, nous proposons tout d'abord une méthode basée sur la définition de l'équation d'évolution de courbe sur graphe (6.4) afin de faire évoluer plusieurs courbes à la fois sur un graphe, en utilisant la définition de la courbure sur graphe. Nous proposons ensuite un algorithme hybride ainsi qu'une implantation utilisant la technologie des GPUs afin d'accélérer et d'améliorer la classification de données sur graphe.

### 6.3.1 Équation d'évolution de courbe multi-classes

En se basant sur la résolution de l'équation générale (6.3) et en remplaçant le terme de vitesse par la courbure, nous obtenons la spécialisation suivante :

$$\frac{\partial \phi}{\partial t} = \kappa |\nabla \phi|. \quad (6.13)$$

Cette équation permet de faire évoluer un front en régularisant sa forme, en fonction de sa courbure. Dans cette section, nous proposons son adaptation sur graphe afin d'obtenir un nouvel algorithme multi-classes basé sur la courbure moyenne pour la classification semi-supervisée de toutes données pouvant être représentées par un graphe. Afin de résoudre cette équation sur graphe, nous utilisons le schéma itératif général précédemment décrit (6.6), en remplaçant la vitesse  $\mathcal{F}$  par la courbure sur graphe  $\kappa_w(u, f)$  (4.65). Cette courbure étant définie en utilisant la définition du périmètre sur graphe, elle est aussi liée à la minimisation des graph-cuts (voir [CES13] et les références connexes pour plus de détails). Nous proposons maintenant une approche afin de gérer plusieurs classes en utilisant ce schéma itératif. Pour ce faire, nous ne représentons pas l'évolution d'un front par son ensemble de niveaux, mais par sa probabilité d'atteindre un sommet du graphe. Ceci peut être considéré comme une méthode d'étiquetage de sommets sur graphe. Considérons  $m$  fronts évoluant sur un graphe. Nous représentons chaque front par une fonction  $f_l : V \rightarrow [-1, 1]$ , avec  $l \in \{1, 2, \dots, m\}$  initialisée telle que :

$$f_l^0 = \mathcal{U}_l = \chi_{\Omega_l} - \sum_{i \in \{1, 2, \dots, m\} \setminus l} \chi_{\Omega_i}, \quad (6.14)$$

où  $\Omega_l \subset V$  est le sous-ensemble de sommets où le front  $\Gamma_l$  a initialement atteint les sommets. En d'autres termes,  $f_l^0(u) = 1$  si  $u$  est à l'intérieur du front  $\Gamma_l$ ,  $f_l^0(u) = -1$  si  $u$  est à l'intérieur d'un autre front et  $f_l^0(u) = 0$  sinon. Afin de faire évoluer  $f_l$  en fonction de la courbure, nous utilisons le schéma itératif général (6.6), en remplaçant  $\phi$  par  $f_l$ . L'étiquetage des sommets est ensuite effectué par  $m$  processus indépendants du schéma itératif général suivant :

$$f_l^{n+1}(u) = \begin{cases} f_l^n(u) + \Delta t \kappa_w(u, f_l^n) \|\nabla_w^+(f_l^n)(u)\|_p, & \text{si } \kappa_w(u, f_l^n)(u) > 0, \\ f_l^n(u) + \Delta t \kappa_w(u, f_l^n) \|\nabla_w^-(f_l^n)(u)\|_p, & \text{si } \kappa_w(u, f_l^n)(u) < 0. \end{cases} \quad (6.15)$$

À la fin de ce processus, chaque fonction  $f_l$  est normalisée, *i.e.*  $f_l(u) \in [0, 1]$ , afin d'estimer la probabilité que le sommet  $u$  appartienne à une classe. La classification finale peut être obtenue pour un sommet donné  $u \in V$  par la formulation suivante :

$$\arg \max_{l \in \{1, \dots, m\}} \{f_l^n(u)\} \quad (6.16)$$

Cette formulation permet de gérer à la fois le cas de la classification semi-supervisée (interpolation de marqueurs dans le graphe entier à partir d'un sous-ensemble de sommets initialement étiquetés) et le cas d'une partition complètement définie (*i.e.* tous les noeuds du graphe possèdent une étiquette) que l'on veut faire évoluer afin de la raffiner.

**Remarque :** pour  $p = \infty$ , cette formulation est très proche de celle du  $p$ -Laplacien avec termes de gradient proposée au chapitre 4, notamment les expérimentations utilisant la courbure moyenne sur graphe pour les fonctions  $\alpha$  et  $\beta$  aux sections 4.5.1 et 4.5.2.

### 6.3.2 Accélération de l'algorithme

#### Algorithme hybride

L'algorithme que nous proposons reposant sur un schéma itératif, le nombre d'itérations requises pour que la solution soit stable peut être très grand. Comme nous l'avons montré dans la section précédente, l'équation eikonale et l'EDP d'évolution par ensemble de niveaux sont liées par un changement de variable. De plus, l'algorithme utilisé pour résoudre l'équation eikonale sur un graphe possède une complexité algorithmique linéarithmique.

Nous proposons ici de combiner ces deux algorithmes afin d'accélérer la convergence de l'algorithme que nous proposons. Pour ce faire, nous calculons

une première partition en utilisant l'algorithme de propagation de front basé sur l'équation eikonale. Nous considérons ce résultat comme une estimation de la partition finale. Nous utilisons ensuite cette partition comme initialisation de l'algorithme d'évolution de courbe multi-classes que nous proposons. Ceci permet de n'effectuer que peu d'itérations de l'algorithme pour obtenir une bonne solution, rendant la combinaison de ces deux algorithmes rapide et efficace.

## **Implantation GPU**

Au cours des dernières années, on a pu observer que les architectures matérielles informatiques modernes ne se livrent plus à une course à la vitesse d'horloge, mais se dirigent davantage vers un matériel conçu pour exécuter plusieurs tâches en parallèle. Il est courant d'avoir un CPU à huit coeurs dans les ordinateurs personnels modernes. On peut aussi citer les smart-phones, où les processeurs gagnent de plus en plus de coeurs. Bien que cela ne soit pas nouveau, on peut également observer que la spécialisation du matériel est de plus en plus employée par les concepteurs de matériel. Les processeurs graphiques (GPU) en sont un bon exemple. Ils sont conçus pour le rendu 3D, ce qui implique généralement d'effectuer des calculs massivement parallélisés, ou, pour simplifier, effectuer principalement le même calcul sur une multitude d'objets, aussi vite que possible. Pour ce faire, les GPUs sont conçus pour exécuter ces tâches en parallèle, ils sont donc composés d'un grand nombre de petites unités de calcul, dédiant plus de transistors aux unités de calcul qu'à la mémoire cache par rapport aux CPUs. Comme on peut le constater, les CPUs suivent la même tendance, ceux-ci étant composés de plus en plus de coeurs. On peut considérer les GPUs comme précurseurs de cette tendance. Comme on peut le voir dans la documentation NVidia de la technologie CUDA [NVI11], il y a un énorme fossé entre les CPUs et les GPUs en termes d'opérations en virgules flottantes par seconde (FLOPS). Utiliser ce matériel précurseur pour l'implantation d'algorithmes afin d'améliorer les temps de calculs peut être assez intéressant, mais son architecture impose des contraintes sur la dépendance aux données de l'algorithme ainsi que sur les accès mémoires.

## **Représentation du graphe**

Il existe différentes façons de représenter un graphe en mémoire, telles que les matrices d'adjacences ou encore les listes chaînées pour représenter le voisinage d'un sommet. Les listes chaînées offrent l'avantages d'être assez petites en termes de mémoire. Cependant, afin de récupérer le voisin d'un noeud, le dérérérencement d'un pointeur est nécessaire, pouvant pointer vers n'importe

quelle adresse en mémoire, ce qui ne garanti pas la contiguïté des accès à la mémoire.

Les matrices d'adjacences pourraient aussi être intéressantes, mais sont coûteuses en termes de mémoires pour des graphes de grandes tailles. Pour pallier cet inconvénient, nous pouvons envisager l'utilisation de matrices creuses. Hélas cela conduit aux mêmes problèmes que l'utilisation des listes chaînées, c'est-à-dire au déréférencement multiple de pointeurs afin d'obtenir un élément de la matrice.

Nous proposons d'utiliser une structure de données simple, qui est un compromis entre les listes chaînées et les matrices d'adjacences. Nous représentons un graphe comme deux matrices contenant, pour la première, les indices des sommets voisins et pour la deuxième, les poids des arêtes associées à ces voisins. Chacune de ces deux matrices est donc de taille  $N \times |V|$ , où  $|V|$  est le nombre de sommets du graphe et  $N$  le degré maximum d'un sommet du graphe (figure de gauche de 6.1). Chaque colonne de la matrice de voisinage contient les indices des sommets voisins. De la même façon, chaque colonne de la matrice de poids contient les poids entre le sommet courant (représenté par l'indice de la colonne courante) et ses voisins. Dans la plupart des cas, les sommets du graphe n'ont pas les mêmes degrés. Comme les poids et indices sont toujours positifs, nous avons choisi de marquer l'espace inutilisé dans chaque matrice via une valeur négative ( $-1$ ).

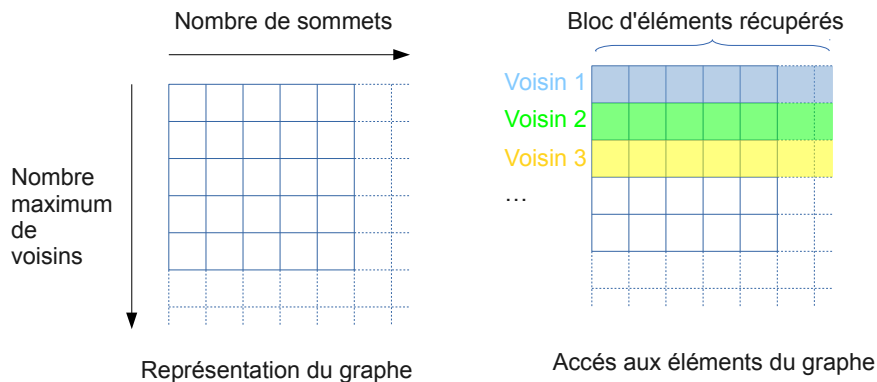


FIGURE 6.1 – De gauche à droite : représentation du graphe et accès aux éléments du graphe.

Afin de calculer la norme du gradient dans le schéma itératif général (6.6), on va accéder séquentiellement aux voisins d'un sommet ainsi qu'aux poids



d'un sommet. Cette structure de données permet de récupérer les poids et les indices par blocs (figure de droite de 6.1) à partir d'un warp (un warp est un ensemble de threads, 32 ou plus, suivant l'architecture GPU, exécutant la même instruction au même moment) en utilisant la contiguïté des données. On peut également discuter de l'utilisation des valeurs négatives pour le remplissage des valeurs inutilisées afin d'arrêter un thread arrivant à la fin du voisinage d'un sommet, créant différents chemins d'exécution. Ceci est en fait pertinent dans ce cas puisqu'un thread arrivant à la fin du voisinage n'a pas plus d'instruction à exécuter, laissant les autres threads continuer leurs exécutions.

### Avantages et inconvénients de cette représentation

L'avantage principal de cette structure de données est d'assurer la contiguïté des données lorsque l'on accède à un élément du graphe, ce qui réduit drastiquement le nombre d'accès à la mémoire globale, réduisant ainsi le temps d'exécution, les unités de calcul ne passant pas leur temps à attendre les données à traiter. L'inconvénient est que l'utilisation de cette structure de données peut induire l'allocation d'une grande quantité de mémoire inutilisée. Par exemple, si un sommet du graphe possède une centaine de voisins et que le reste des sommets ne possède chacun qu'un seul voisin,  $|V| \times 100$  éléments seront alloués en mémoire, alors que seulement  $(|V| - 1) + 100$  éléments sont réellement nécessaires. Comme nous avons utilisé des graphes des  $k$ -ppv pour nos expérimentations, nous n'avons pas rencontré ce problème.

### Implantation GPU de l'algorithme d'évolution de courbe multi-classe

Afin d'implanter l'algorithme d'évolution de courbe sur graphe (6.6) en utilisant CUDA, notre approche est de considérer qu'un thread correspond à un sommet du graphe. Ainsi, pour une itération, chaque thread calcule la solution pour un sommet. Étant donnée une itération de l'algorithme, la solution en chaque sommet est indépendante, l'implantation GPU de cette partie est quasiment directe par rapport à une implantation CPU classique. La seule contrainte ici est de pouvoir accéder aux éléments du graphe efficacement, comme discuté précédemment. L'algorithme 2 décrit globalement l'implantation du processus d'évolution de courbe multi-classes. La portion de code exécuté par le GPU est appelé ParallelLS dans l'algorithme 2, elle est appelée autant de fois que nécessaire (le nombre d'itérations) par le côté CPU de l'application. Celle-ci est lancée en parallèle à chaque itération, avec autant de threads que de sommets dans le graphe. Les détails sur le calcul de la solution pour une itération en un sommet du graphe sont données à l'algorithme 3, qui est la portion exécutée par le GPU.

---

**Algorithme 2:** Évolution de courbe multi-classes

---

**Données :**  $P_i$  : Partition initiale ;  
 $G$  : Graphe ;  
 $n$  : nombre d'itération  
**1 début**  
**2**   └ Copier le graphe  $G$  dans la mémoire du GPU :  $G_d$  ;  
**3 pour chaque classe  $l$  dans  $P$  faire**  
**4**   └ Construire la fonction initiale :  $f_l^0$  ;  
**5**   └ Allouer  $f_{ld}^1$  et  $f_{ld}^2$  dans la mémoire centrale du GPU ;  
**6**   └ Copier  $f_l^0$  dans  $f_{ld}^1$  ;  
**7**   **tant que  $i < n$  faire**  
**8**   └   **ParalleLS**( $G_d, f_{ld}^1, f_{ld}^2$ ) ;  
**9**   └   Synchronisation des threads (attente de la fin de tous les  
       threads) ;  
**10**  └   Swap( $f_{ld}^1, f_{ld}^2$ ) ;  
**11**  └ Copier  $f_{ld}^1$  vers la mémoire centrale CPU :  $F(l) = f_{ld}^1$  ;  
**12** Calculer la partition finale à partir de  $F$  :  $P_f$  ;

---



---

**Algorithme 3:** ParalleLS

---

**Données :**  $G$  : Graphe ;  
 fonctions :  $f_{ld}^1, f_{ld}^2$  ;  
**1 début**  
**2**   └ Récupérer l'indice du sommet courant à partir de l'indice du thread :  
        $u$  ;  
**3** Calculer la courbure  $\kappa_w(u, f_{ld}^1)$  ;  
**4 si  $\kappa_w(u, f_{ld}^1) > 0$  alors**  
**5**   └  $f_{ld}^2(u) = f_{ld}^1(u) + \Delta t \kappa_w(u, f_{ld}^1) \|\nabla_w^+(f_{ld}^1)(u)\|$   
**6 si  $\kappa_w(u, f_{ld}^1) < 0$  alors**  
**7**   └  $f_{ld}^2(u) = f_{ld}^1(u) + \Delta t \kappa_w(u, f_{ld}^1) \|\nabla_w^-(f_{ld}^1)(u)\|$

---

## 6.4 Classification semi-supervisée

Dans beaucoup de problèmes en classification, l'acquisition de données dont on connaît la classe afin d'entraîner un classifieur est coûteuse et/ou peut prendre beaucoup de temps. Acquérir des données dont on ne connaît pas la classe est par contre beaucoup plus facile. Dans ce contexte, les algorithmes de classifications semi-supervisées, apprenant à la fois des données étiquetées et non-étiquetées, sont très intéressants. La plupart des algorithmes de classification semi-supervisée sont basés sur l'hypothèse que les points qui sont assez proches au sens d'une métrique, ou dans un même cluster, devraient appartenir à la même classe.

Dans cette section, nous proposons d'évaluer la nouvelle famille d'opérateurs de  $p$ -Laplace proposée dans cette thèse, l'algorithme reposant sur la résolution de l'équation eikonale sur graphe, ainsi que l'algorithme d'évolution de courbe multi-classes que nous proposons, pour la tâche de classification semi-supervisée.

Nous proposons de les comparer au  $p$ -Laplacien variationnel sur graphe ainsi qu'à l'un des plus récents algorithmes de clustering et de classification appelé Multiclass Total Variation clustering (MTV) proposé par [BTUvB13]. Nous avons évalué ces méthodes en utilisant trois bases de données standard de la littérature : MNIST [LC10], OPTDIGITS [AK98] et PENDIGITS [AA98]. Ces bases de données sont composées de chiffres manuscrits, représentés sous forme d'images ou de coordonnées dans le plan. Elles sont découpées en deux ensembles : un ensemble d'apprentissage et un ensemble de test. Comme effectué dans [BTUvB13], nous avons fusionné ces deux ensembles afin d'effectuer la classification semi-supervisée sur les bases de données entières, produisant des ensembles de données de 70000, 5620 et 10992 éléments respectivement pour MNIST, OPTDIGITS et PENDIGITS. Dans le reste de cette section, le  $p$ -Laplacien normalisé proposé au chapitre 3 sera noté  $\mathbf{G}p\mathbf{LPL}$ , l'algorithme de propagation de front utilisant l'équation eikonale sera noté  $\mathbf{FM}$  (pour Fast Marching), l'algorithme d'évolution de courbe multi-classe sera noté  $\mathbf{LS}$  et le  $p$ -Laplacien variationnel sur graphe sera noté  $p\mathbf{LPL}$ . L'accélération proposée de l'algorithme  $\mathbf{LS}$ , raffinant les résultats de classification de  $\mathbf{FM}$  avec l'algorithme  $\mathbf{LS}$ , sera noté  $\mathbf{FM} + \mathbf{LS}$ .

### 6.4.1 Méthode d'évaluation

Afin d'évaluer nos algorithmes, nous les avons exécutés dix fois chacun, pour le même jeu de paramètres, en initialisant les marqueurs initiaux aléatoirement. Nous avons fait varier le pourcentage de marqueurs initiaux (1%, 2%, 5% et

10%) et avons gardé le meilleur taux de classification, le taux de classification moyen sur les dix exécutions ainsi que l'écart type du taux moyen de classification.

Pour les bases de données OPTDIGITS et PENDIGITS nous avons utilisé les versions pré-traitées, donnant des vecteurs d'attributs de tailles constantes ainsi que l'invariance aux petites distorsions (voir [AA98] et [AK98] pour plus de détails sur les routines de pré-traitements).

## 6.4.2 Construction des graphes

Pour ces bases de données, nous avons construits des graphes des  $k$ -ppv, avec  $k = 10$ , en utilisant différentes métriques. Pour la base de données MNIST nous avons construit le graphe en utilisant la "two-sided tangent distance" [KDTN00]. Cette métrique est particulièrement adaptée à cette base de données, fournissant une invariance aux transformations affines. Les deux autres bases de données étant déjà pré-traitées, nous avons utilisé la distance Euclidienne  $d_2$  (1.11). Nous avons calculé le poids entre les paires de sommets en utilisant la similarité Gaussienne  $s_3$  (1.17) avec  $d$  les métriques utilisées pour construire les  $k$ -ppv. Le paramètre  $\sigma$  étant assez difficile à régler pour cette fonction de similarité, nous considérons ici des stratégies pour estimer sa valeur automatiquement.

Nous avons considéré deux stratégies : soit, utiliser  $\sigma$  comme un paramètre d'échelle global, soit comme un paramètre d'échelle local, comme proposé par [ZMP04].

- **Estimation locale** : Dans ce cas, la fonction de similarité devient  $s(u, v) = \exp \frac{-d(u,v)^2}{\sigma_u \sigma_v}$ , où  $\sigma_u$  est un paramètre d'échelle local au sommet  $u$ . Nous avons calculé chaque  $\sigma_u$  comme la distance au  $M$ -ième plus proche sommet de  $u$ .
- **Estimation globale** : Pour estimer le paramètre  $\sigma$  globalement, nous avons utilisé la méthode décrite par [Ker04]. L'auteur de cet article propose une méthode robuste d'estimation globale et locale du paramètre  $\sigma$ . Ici nous n'utilisons que l'estimation globale, donnée par :

$$\hat{\sigma} = 1.4826 \operatorname{median}(|\mathcal{E}_S| - \operatorname{median}|\mathcal{E}_S|), \quad (6.17)$$

où  $\mathcal{E}_S$  est l'ensemble des résidus locaux du graphe, calculés tel que :

$$\mathcal{E}_i = \left( \sum_{v \sim u} f(u) - f(v) \right) / \sqrt{|v \sim u|^2 + |v \sim u|}, \quad (6.18)$$

pour un sommet  $u$ . Pour plus de détails et de justifications, voir [Ker04].

### 6.4.3 Fonctions de potentiels

Pour l'algorithme FM, basé sur la résolution de l'équation eikonale (algorithme 1, équation (6.8)), nous avons essayé trois fonctions de potentiels différentes (le potentiel est l'inverse de la fonction de vitesse  $\mathcal{F}$  à l'équation (6.7)). Soit  $F$  le vecteur d'attribut associé à chaque élément de la base de donnée. Nous avons utilisé un potentiel constant ( $P = 1$ ), la norme locale du gradient ( $P(u) = \|\nabla(F)(u)\|_2$ ) et la distance à la moyenne des sommets déjà dans les régions atteintes par le front ( $P(u) = \|F(u) - \mathcal{C}_m\|$ , où  $\mathcal{C}_m$  est la moyenne des éléments à l'intérieur du front  $\Gamma_m$ ). Afin d'utiliser efficacement ce potentiel, nous avons attribué une étiquette différente à chacun des marqueurs initiaux, considérant autant d'étiquettes différentes que de germes initiaux, afin que chaque moyenne soit indépendante par germe. Une fois la propagation de front terminée, l'étiquette initiale est réattribuée à chaque région. La figure 6.2 présente une comparaison des trois fonctions de potentiels que nous avons utilisées avec l'équation eikonale sur la base de données OPTDIGITS. Comme on peut le voir, le potentiel basé sur la distance à la moyenne des éléments à l'intérieur du front donne de meilleurs résultats en général, spécialement lorsque le taux de germes initiaux est bas.

### 6.4.4 Influence du paramètre $\sigma$

Afin d'avoir une intuition sur la manière de choisir la fonction de poids lors de la construction du graphe, nous avons évalué les algorithmes en utilisant les estimations globales et locales du paramètre  $\sigma$  décrites en section 6.4.2. Les figures 6.3 et 6.4 montrent respectivement le taux de classification moyen et l'écart type de ce taux sur dix exécutions, en utilisant les estimations globales et locales de  $\sigma$ . Comme on peut le voir, les résultats sont globalement meilleurs pour l'algorithme FM + LS en utilisant une estimation locale de  $\sigma$ . La figure 6.4 montre que l'écart type du taux de classification est bien plus petit lorsque l'on utilise une estimation locale de  $\sigma$ , montrant plus de robustesse à l'initialisation des germes. Lors de nos expérimentations, l'algorithme FM+LS a toujours donné les meilleurs résultats en utilisant l'estimation locale de  $\sigma$  et la meilleure robustesse à l'initialisation des germes. On peut aussi constater que l'estimation locale de  $\sigma$  avec FM + LS améliore les résultats lorsque le taux de germe initiaux est faible.

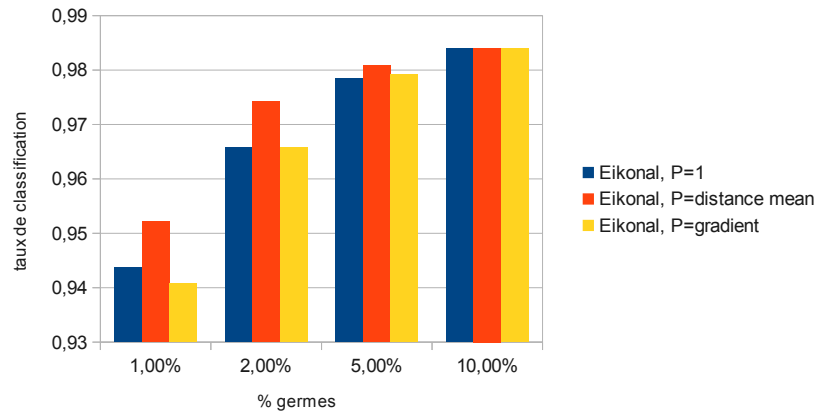


FIGURE 6.2 – Comparaison des trois fonctions de potentiels utilisées avec l'équation eikonale pour la classification semi-supervisée de la base de donnée OPTDIGITS. L'axe horizontal représente le pourcentage de germes utilisés et l'axe vertical le taux de classification. Voir le texte pour plus de détails sur les fonctions de potentiels.

### 6.4.5 Évaluation et comparaison

Dans cette section nous comparons quantitativement :

- le  $p$ -Laplacien normalisé proposé au chapitre 3 ( $GpLPL$ ),
- l'algorithme reposant sur la résolution de l'équation eikonale sur graphe (FM),
- l'algorithme d'évolution de courbe multi-classes que nous proposons dans ce chapitre (LS),
- le  $p$ -Laplacien variationnel sur graphe [ELB08, BEM09] ( $pLPL$ ),
- l'algorithme MTV [BTUvB13],

en utilisant les trois bases de données citées précédemment.

Nous avons effectué ces comparaisons en faisant varier différents paramètres : le nombre de germes initiaux, le paramètre d'échelle  $\sigma$  (global ou

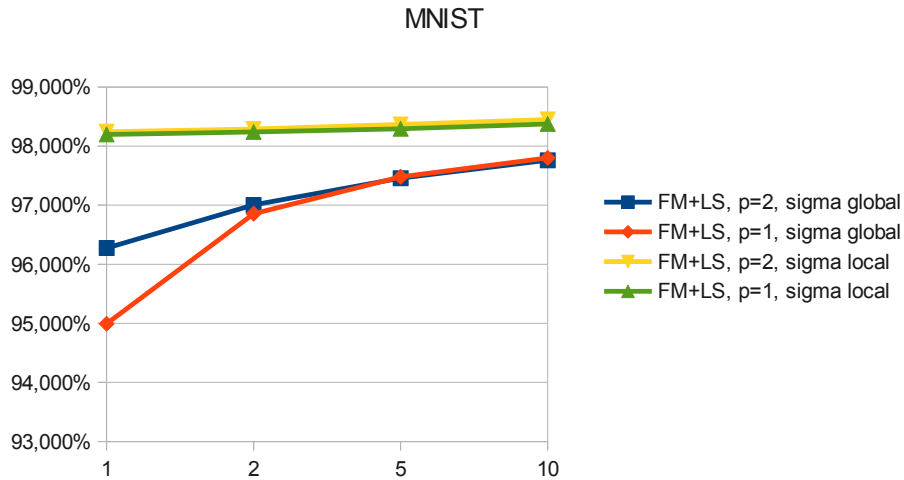


FIGURE 6.3 – Comparaison du taux de classification, en % sur la base de données MNIST en utilisant les estimations globales et locales du paramètre  $\sigma$ . L'axe horizontal représente le pourcentage de germe utilisé.

local), la fonction de potentiel pour l'algorithme FM, le paramètre  $p$  pour les  $p$ -Laplacien et la norme du gradient de l'équation eikonale. Comme expliqué précédemment, pour chaque jeu de paramètres, nous avons lancé dix fois nos algorithmes en faisant varier aléatoirement la position initiale des germes initiaux.

Les comparaisons des résultats sont visibles aux tableaux 6.1, 6.2, 6.3. Nous avons ici affiché le taux de classification moyen sur les dix exécutions en utilisant le jeu de paramètres donnant les meilleurs résultats. Comme on peut le voir, pour les bases de données MNIST et PENDIGITS, la combinaison des algorithmes FM + LS donne à chaque fois les meilleurs résultats, alors que pour OPTDIGITS, les résultats sont un peu moins bons mais restent comparables à ceux de l'algorithme MTV.

Afin d'illustrer le comportement du  $p$ -Laplacien normalisé que nous proposons au chapitre 3, nous avons fait varier le paramètre  $p$  tel que  $1 < p \leq 2$ , sans changer le reste du jeu de paramètre. Comme on peut le voir sur les figures 6.5 et 6.6, plus  $p$  est proche de 1, meilleur est le taux de classification.

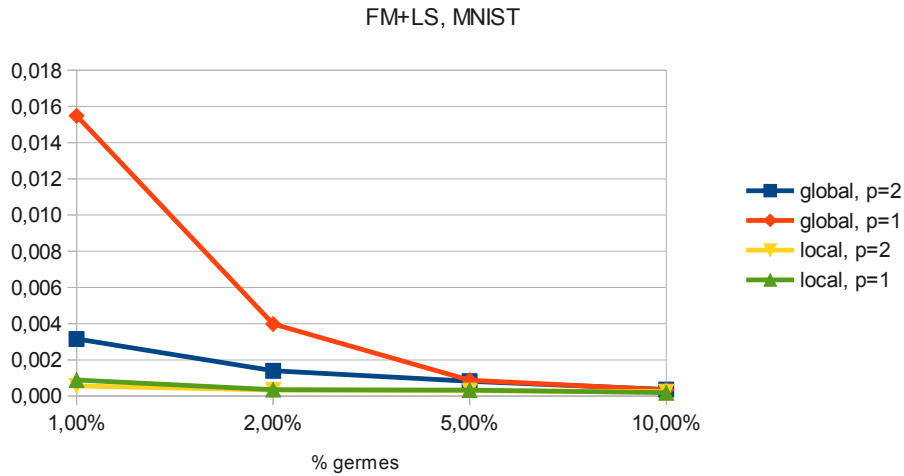


FIGURE 6.4 – Écart type du taux de classification.

% germes	$p$ LPL	$Gp$ LPL	FM	LS	FM+LS	MTV
1%	97.84%	97.97%	97.45%	98.20%	<b>98.24%</b>	97.59%
2%	97.88%	98.03%	97.64%	98.24%	<b>98.29%</b>	97.72%
5%	97.99%	98.05%	97.95%	98.33%	<b>98.37%</b>	97.79%
10%	98.02%	98.14%	98.19%	98.39%	<b>98.45%</b>	98.05%

TABLE 6.1 – Taux moyen de classification de la base de données MNIST.

### 6.4.6 Temps d'exécutions

Afin de présenter les temps d'exécution, nous avons exclu la construction de graphe, qui peut être effectuée en amont. Les résultats sont présentés aux tableaux 6.4 et 6.5. La machine que nous avons utilisée pour effectuer ces tests est équipée d'un Intel Core i7 3930K@3.2GHz, 16 GB de RAM en DDR3 et une NVidia GeForce GTX 580. Nous précisons ici que l'implantation CPU n'a pas été parallélisée.

On peut remarquer que la combinaison des algorithmes FM + LS est l'un des algorithmes les plus rapides. Ceci est due au fait qu'il ne nécessite que très peu d'itérations pour raffiner la partition. C'est aussi l'algorithme qui donne en moyenne les meilleurs résultats de classification.



% germes	$p$ LPL	$Gp$ LPL	FM	LS	FM+LS	MTV
1%	95%	96.77%	95.22%	96.82%	97.10%	<b>98.29%</b>
2%	97.53%	97.7%	97.41%	97.88%	97.92%	<b>98.35%</b>
5%	98.12%	97.9%	98.09%	98.38%	98.35%	<b>98.38%</b>
10%	98.05%	98.22%	98.41%	<b>98.64%</b>	98.51%	98.45%

TABLE 6.2 – Taux moyen de classification de la base de données OPTDIGITS.

% germes	$p$ LPL	$Gp$ LPL	FM	LS	FM+LS	MTV
1%	94.97%	96.08%	95.75%	95.71%	<b>96.25%</b>	93.73%
2%	96.81%	97.19%	97.38%	97.06%	<b>97.57%</b>	95.83%
5%	97.95%	98.34%	98.25%	98.30%	<b>98.56%</b>	97.98%
10%	98.61%	99%	98.94%	98.92%	<b>99.10%</b>	98.22%

TABLE 6.3 – Taux moyen de classification pour la base de données PENDIGITS.

Alg	MNIST	OPTDIGITS	PENDIGITS
FM	1.76	0.08	0.11
LS CPU	232.6	18.2	10.6
LS GPU	21	1.5	1.5
FM + LS CPU	25.02	1.19	1.09
FM + LS GPU	3.86	0.24	0.26

TABLE 6.4 – Comparaison des temps d'exécution, en seconde.

Alg	MNIST	OPTDIGITS	PENDIGITS
<b>accélération GPU LS</b>	11.08×	12.13×	7.1×
<b>accélération GPU FM + LS</b>	6.48×	4.96×	4.19×

TABLE 6.5 – Accélération GPU.

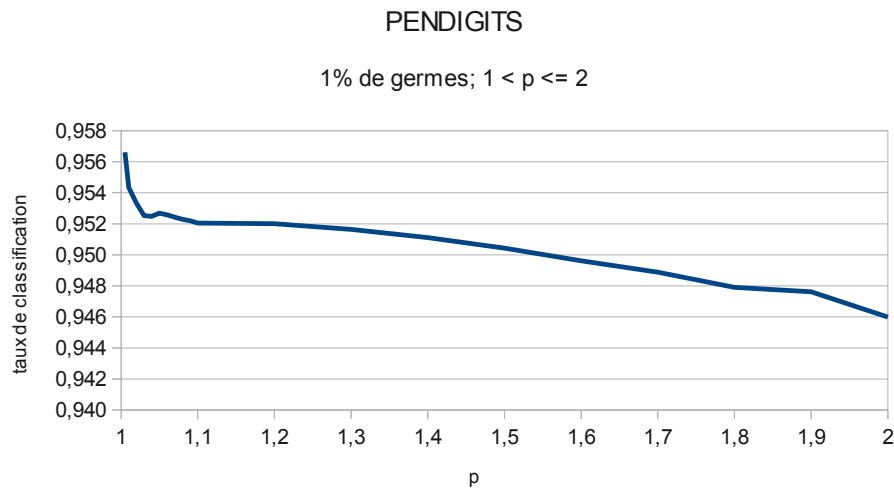


FIGURE 6.5 – Taux de classification obtenu en utilisant  $\Delta_{w,p}^N$  avec la base de données PENDIGITS, pour  $1 < p \leq 2$ .

### 6.4.7 Applications en microscopie virtuelle

Après avoir évalué nos algorithmes sur différentes bases de données de la littérature, nous proposons dans cette section de les utiliser pour la classification de noyaux de cellules dans le contexte de la microscopie virtuelle.

#### Contexte

Traditionnellement, les décisions de diagnostics prises par les cytopathologistes sont basées sur des caractéristiques de morphologies et de textures des composants cellulaires, vues à travers un microscope. Ce processus peut être très long, une lame de cytologie pouvant contenir des milliers, voir des millions de cellules, dans laquelle les cellules anormales sont très rares ou heureusement absentes. Durant la dernière décennie, l'avancée des scanners de lames hautes résolutions rapides et efficaces combinée avec le développement de la vision par ordinateur a ouvert la voie à l'utilisation de la pathologie numérique comme un outil de diagnostic. La figure 6.8 montre des exemples d'images cytologiques. Leur segmentation automatique peut être effectuée en utilisant l'équation eikonale combinée à des opérateurs de bases de traitement d'image, comme nous l'avons effectué dans [TDEL13].

Une fois les noyaux de cellules extraits, afin d'aider les pathologistes dans leur diagnostic, le but est de les classifier afin de savoir si des cellules anormales

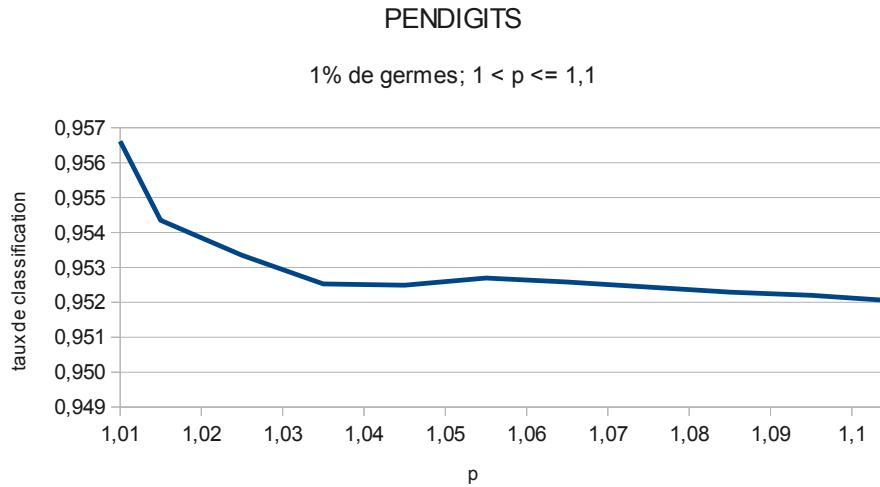
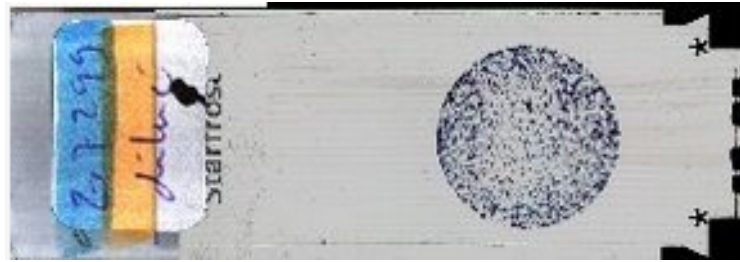
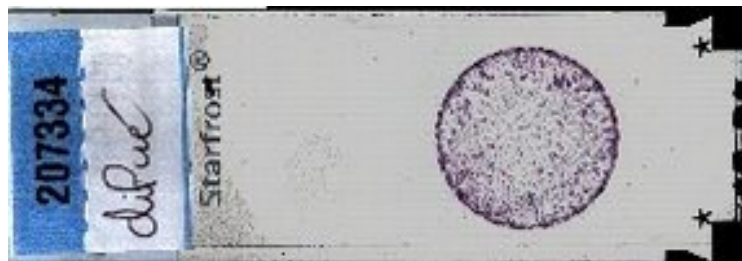


FIGURE 6.6 – Détails de la figure 6.5 pour  $1 < p \leq 1.1$ .

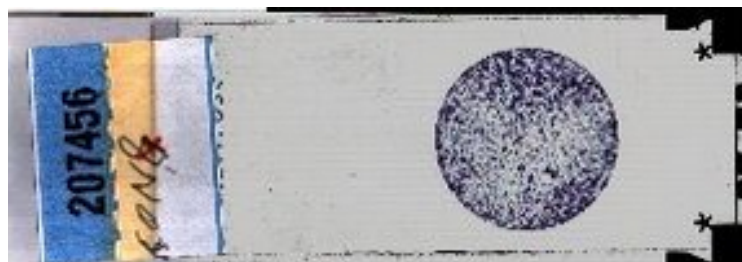
sont présentes sur la lames. Pour ce faire, il est commun d'utiliser des approches d'apprentissage automatique, utilisant des bases de données d'apprentissage. Afin de créer une base de données d'apprentissage, nous pourrions utiliser les images où l'extraction des noyaux a été effectuée : ce type de base de données peut contenir des millions de noyaux à classifier. Construire une base de données d'apprentissage contenant une telle quantité d'objets est évidemment infaisable par un être humain. Dans ce contexte, les techniques de classification semi-supervisée peuvent être un outil intéressant pour aider à étiqueter les bases de données de cellules. Nous proposons ici de construire une base de données d'apprentissage en utilisant les méthodes de classification semi-supervisées sur graphes et de l'évaluer sur une base de données de référence complètement étiquetée. Cette approche a trois principaux avantages. Le premier étant le fait qu'il y ait très peu d'objets à étiqueter manuellement. Deuxièmement, grâce à l'absence d'entraînement de classifieur, modifier la base de référence n'implique pas de réentraîner un classifieur, mais simplement d'ajouter une nouvelle donnée, sous la forme d'un sommet, au graphe représentant la base de données. Troisièmement, les graphes fournissent une représentation intrinsèque de l'organisation des différentes classes et la position de chaque noyau de cellule dans chaque classe. Une telle information peut être d'une importance cruciale pour les cythopathologistes dans le cas de cellules ambiguës.



(a)



(b)



(c)

FIGURE 6.7 – Exemples de lames cytologiques.

### Bases de données

Afin de pouvoir nous comparer aux méthodes de la littérature, nous avons utilisé la base de données de cellules séreuses provenant de [LEC03]. Elle est composée de 3956 objets, divisés en 18 classes, pouvant être agrégées en 4 classes : Polynucléaires (1117 instances), lymphocytes (1111 instances), macrophages / mésothéliales (1231 instances) et cellules anormales (497 instances). Nous pouvons considérer les 3 premières classes comme représentant les cellules normales. Chaque noyau est représenté par des caractéristiques (surface, forme, couleur, texture, ...), pour un total de 45 caractéristiques.

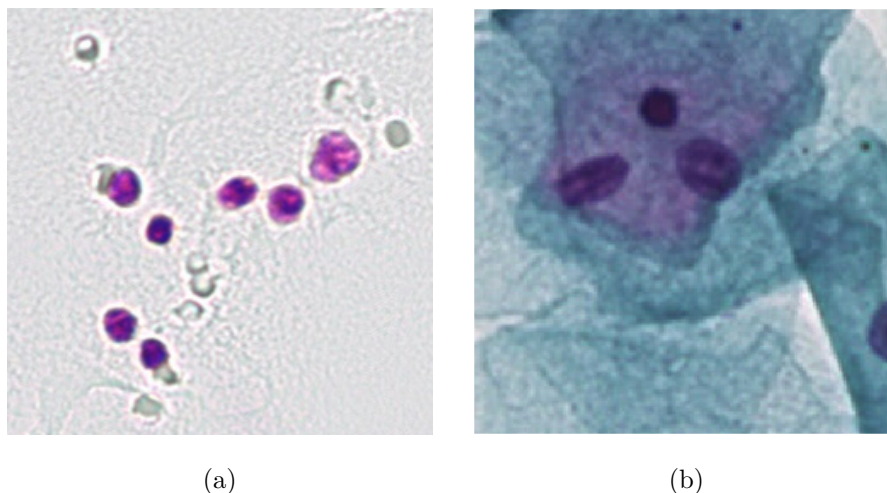


FIGURE 6.8 – Exemples d’images cytologiques. L’image (a) est issue d’une lame teintée avec la coloration Feulgen, où les noyaux apparaissent en rose. L’image (b) provient d’une lame teintée avec la coloration Papanicolaou (gynécologie), les noyaux apparaissent en bleu foncés et le cytoplasme apparaît en bleu clair.

Nous avons aussi construit notre propre base de données, en partenariat avec le service d’anatomopathologie du centre hospitalier public du Cotentin (CHPC) et la société Datexim. Cette base de données a été construite à partir d’images gynécologiques. Elle est composée de 3225 objets, divisés en 8 classes représentant les cellules : superficielles, intermédiaires, basales, endocervicales, métaplasie/dystrophie, polynucléaires, anormales et très anormales. De la même manière que pour la base de séreuses, nous pouvons considérer ici les 6 premières classes comme des cellules normales. Les noyaux sont aussi représentés de la même façon que pour la base de cellules séreuses, c’est-à-dire par des caractéristiques extraites des images dans lesquelles ils ont été segmentés, pour un total de 190 caractéristiques.

### Classification de noyaux

Pour la base de cellules séreuses, nous avons utilisé le même protocole d’évaluation que pour les expérimentations précédentes. Nous avons aussi utilisé la même méthode de construction de graphe, ainsi que les mêmes fonctions de pondérations et de potentiels pour l’équation eikonale. La figure 6.10(a) montre les résultats moyens obtenus en utilisant les 4 classes décrites plus haut. Nous obtenons des résultats comparables au réseau de neurones utilisé dans [LECO3].

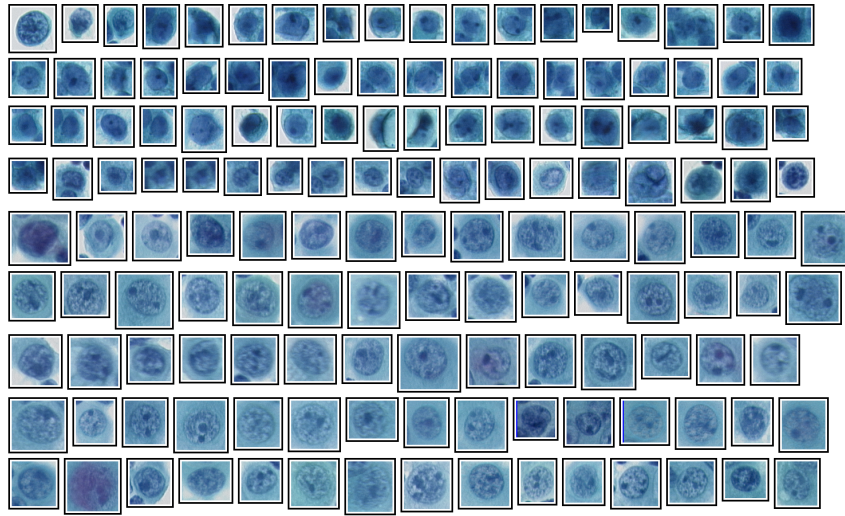
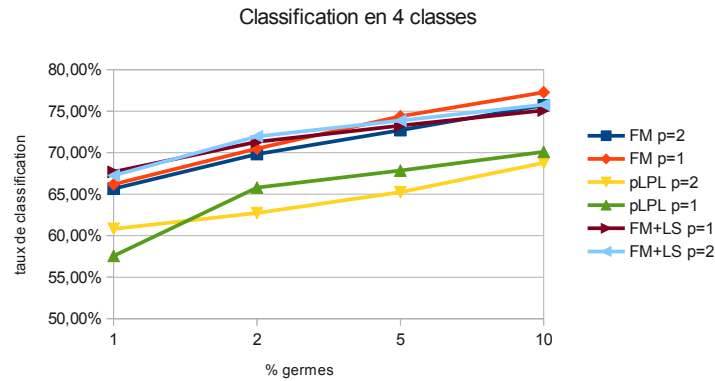


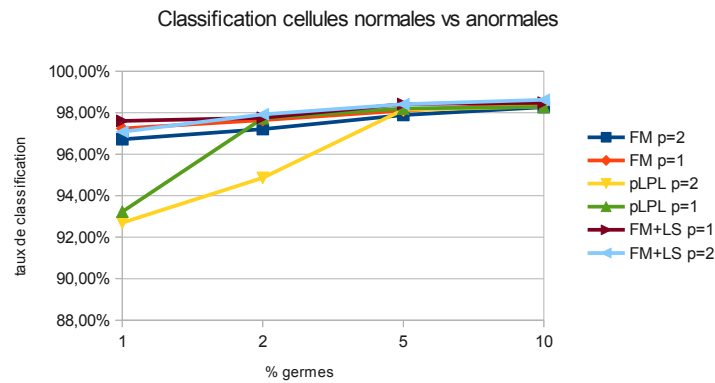
FIGURE 6.9 – Exemples de noyaux anormaux.

Il est important ici de mentionner le fait que lorsque l'on croise l'étiquetage de différents pathologistes sur cette base de données, le taux de recouvrement est de l'ordre de 60%, ce qui montre bien la difficulté du problème d'étiquetage. La façon d'étiqueter les cellules que nous proposons peut aussi être une manière de consensus entre les pathologistes. Ce taux de recouvrement est principalement dû à des différences lors de l'étiquetage des cellules normales. Comme on peut le voir à la figure 6.10(b), la classification de cette base de données en 2 classes (cellules normales vs anormales) donne de bien meilleurs résultats. Cette information est d'une grande importance lors de la prise de décisions de diagnostic.

Pour la base de cellules gynécologiques, nous avons tout d'abord sélectionné les caractéristiques les plus discriminantes en utilisant l'algorithme mRMR [PLD05]. À l'aide de cet algorithme, nous avons créé 3 bases de données contenant les 30, 40 et 50 caractéristiques sélectionnées. Nous avons ensuite procédé de la même manière que pour les expérimentations précédentes. Pour cette expérimentation, nous nous sommes comparés à un classifieur SVM. Nous avons procédé de la manière suivante : nous avons normalisé les caractéristiques, puis divisé la base de données en deux sous ensembles aléatoires (80% des éléments pour former la base d'apprentissage et 20% pour former la base de test). Pour l'apprentissage, nous avons utilisé une 5-fold cross validation. Afin de comparer cet algorithme avec les méthodes sur graphe, nous ne prenons plus 1%, 2%, 5% ou 10% de germes initiaux, mais 80%. Nous classifions ensuite les 20% de la base de donnée restante. Les résultats sont



(a)



(b)

FIGURE 6.10 – Taux de classification de la base de cellules séreuses en utilisant l'équation eikonale sur graphe, le  $p$ -Laplacien sur graphe et la combinaison FM+LS, en faisant varier le taux de germes initiaux. La figure (a) montre les taux de classifications obtenus en utilisant 4 classes. La figure (b) montre le taux de classification en utilisant l'agrégation de la base de données en 2 classes. L'axe horizontal représente le pourcentage de germes initial et l'axe vertical représente le taux de classification.

affichés au tableau 6.6. On peut remarquer que les algorithmes  $p$ LPL et  $Gp$ LPL ont exactement les mêmes résultats. Ceci est dû au fait qu'ici nous avons obtenu les meilleurs résultats en utilisant  $p = 2$ . Nous avons utilisé l'algorithme de Gauss-Jacobi pour résoudre le problème de Dirichlet avec le  $p$ -Laplacien sur graphe. Pour une itération de celui-ci, avec  $p = 2$ , la formulation est exactement

la même que pour le schéma itératif que nous avons utilisé pour le  $p$ -Laplacien normalisé au chapitre 3.

	FM	$p$ LPL	G $p$ LPL	SVM
8 classes	73.47%	73.37%	73.37%	<b>73.8%</b>
normales vs anormales	88.56%	88.44%	88.44%	<b>90.54%</b>

TABLE 6.6 – Taux de classification de la base de données de noyaux de cellules d’images gynécologiques.

On peut aussi remarquer le même phénomène que pour la base de données de cellules séreuses, à savoir un bien meilleur taux de classification pour la séparation des cellules normales et anormales.

## 6.5 Clustering / Classification non-supervisée

Le Clustering (ou regroupement, en français) consiste à partitionner automatiquement des données en clusters, en fonction d’une mesure de similarité. Dans la plupart des cas, les données sont représentées par des graphes pondérés, où chaque élément de l’ensemble de données à partitionner est représenté par un sommet du graphe. La similarité entre les points de données est représentée par l’ensemble des arêtes du graphe. Ce scénario correspond exactement à la façon dont nous utilisons les graphes, mais aussi à beaucoup d’algorithmes de clustering, tel que le clustering spectral [VL07].

Dans cette section, inspiré par les travaux de [BHL<sup>+</sup>14], nous proposons un algorithme de classification non-supervisée, utilisant les algorithmes de classifications transductifs que nous avons proposés aux sections et chapitres précédents. Nous évaluons ensuite cet algorithme en termes de pureté de cluster sur 20 bases de données et nous le comparons avec les approches de [YHD<sup>+</sup>12] et [BHL<sup>+</sup>14].

### 6.5.1 Algorithme proposé

L’idée principale de l’algorithme proposé par [BHL<sup>+</sup>14], appelé INGRES (Incremental Reseeding Strategy for Clustering), repose sur une propriété connue des marches aléatoires sur graphes : un marcheur aléatoire, commençant son parcours dans un cluster donné, ne va probablement pas en sortir rapidement. Cette propriété fournit la base des algorithmes de classifications



par propagation de germes, tels que ceux que nous avons utilisés et proposés. Dans le contexte semi-supervisé, l'appartenance de certains éléments des données à certaines classes est connue à l'avance, on peut donc placer ces éléments comme des germes et utiliser des algorithmes de propagation de germes afin de classifier l'ensemble des données. Ce n'est pas le cas dans le contexte non-supervisé, les auteurs de [BHL<sup>+</sup>14] proposent donc l'approche suivante afin d'utiliser des algorithmes de classification transductifs dans le contexte non-supervisé.

Partons du principe que le graphe construit à partir des données est composé de  $m$  clusters de tailles comparables. Afin de contourner le fait de ne pas connaître à l'avance où placer les germes, ils placent tout d'abord des germes aléatoirement. Ils appliquent ensuite sur ce graphe une matrice de random walk itérativement afin de propager les germes sur la totalité du graphe. Ils obtiennent ainsi une partition temporaire des données en assignant chaque sommet à chaque germe le représentant le mieux. Ils choisissent ensuite des germes aléatoirement dans la partition temporaire créée et répètent l'opération. Ils proposent aussi d'ajouter de plus en plus de germes au fur et à mesure des itérations de ce processus.

## Détails de l'algorithme INCRES

L'algorithme INCRES peut être formalisé de la façon suivante : Soit  $G = (V, W)$  un graphe pondéré composé d'un ensemble de  $N$  sommets  $V$  et d'arêtes pondérées  $W$  encodant la similarité entre chaque paire de sommets du graphe. Soit  $D$  la matrice diagonale des degrés pondérés de chaque sommet du graphe  $G$ . L'algorithme est initialisé avec une partition des sommets aléatoire  $P = (\mathcal{C}_1, \dots, \mathcal{C}_m)$ ,  $\mathcal{C}_1 \cup \dots \cup \mathcal{C}_m = V$ ,  $\mathcal{C}_r \cap \mathcal{C}_q = \emptyset (r \neq q)$ . Soit  $s = 1$  le nombre initial de germes par classe.

La première étape de cet algorithme consiste à sélectionner  $s$  germes dans chaque cluster (étape PLANT de l'algorithme 4). Ces germes sont ensuite propagés à travers tout le graphe en utilisant les marches aléatoires (étape GROW de l'algorithme 4) : L'étape PLANT initialise une matrice  $F$ , de taille  $N \times m$ , avec  $F_{i,j} = 1$  si le sommet  $i$  du graphe a été sélectionné comme germe du cluster  $j$  et  $F_{i,j} = 0$  sinon. L'étape GROW va ensuite appliquer itérativement à  $F$  une matrice de random walk :  $F^{n+1} = (WD^{-1})F^n$ , jusqu'à ce que toutes les entrées de la matrices soient non-nulles, calculant ainsi la probabilité pour chaque marcheur aléatoire d'atteindre chaque sommet. Une fois que tous les sommets du graphe sont atteints, chaque sommet est assigné à la classe à laquelle il est le plus susceptible d'appartenir (étape HARVEST de l'algorithme 4).

---

**Algorithme 4:** Algorithme INCRES

---

**Données :**  $W$  : Matrice de similarité;  
 $ds$  : incrément du nombre de germe;  
 $m$  : nombre de cluster;

- 1 **début**
- 2    $s = 1$ ;
- 3    $P$  : partition aléatoire;
- 4 **tant que faire**
- 5    $F = \text{PLANT}(P, s)$ ;
- 6    $F \leftarrow \text{GROW}(F, W)$ ;
- 7    $P \leftarrow \text{HARVEST}(F)$ ;
- 8    $s \leftarrow s + ds$ ;

---

Cet algorithme n'a qu'un seul paramètre,  $ds$ , contrôlant le taux d'augmentation du nombre de germes utilisés à chaque itération. Les auteurs de [BHL<sup>+</sup>14] proposent d'utiliser un taux d'accroissement linéaire  $ds = \mathbf{speed} \times 10^{-4} \times \frac{N}{m}$ , avec  $\mathbf{speed}$  compris entre un et dix. En utilisant cette formule, avec le paramètre de vitesse  $\mathbf{speed} = 1$ , le nombre total de germes serait égal à  $s = 0.1N/m$  après 1000 itérations, ce qui correspond à environ 10% des données pour des clusters de tailles équivalentes, ce qui correspond en moyenne au taux auquel cet algorithme se stabilise.

**Algorithme proposé**

Nous proposons ici de tirer parti du fonctionnement de cet algorithme, afin d'utiliser les méthodes que nous proposons, pour les tâches de clustering. Pour ce faire nous proposons de remplacer les étapes GROW et HARVEST de l'algorithme 4 par les méthodes que nous avons proposées et utilisées dans la section précédente pour la classification semi-supervisée.

En effet, on peut justifier l'utilisation des algorithmes que nous proposons de la même manière que l'ont fait les auteurs de [BHL<sup>+</sup>14] avec les marches aléatoires. Pour le  $p$ -Laplacien normalisé, nous utilisons le problème de Dirichlet associé à la formulation pour la classification proposée en section 3.6.2, celui-ci présentant les mêmes propriétés que les marches aléatoires.

Pour le fast marching sur graphe, l'algorithme repose sur le calcul des distances à partir d'un front initial, ici représenté par les germes initiaux. Le front va se déplacer très vite à l'intérieur des zones homogènes (*e.g.* dans un cluster de données similaires). Inversement, le front va se déplacer très lentement dès lors que la zone traversée n'est plus homogène (*e.g.* à la frontière

de clusters). Il est donc peu probable que le front issu d'un germe initialement placé au centre d'un cluster aille "plus vite" dans un autre cluster que le front initialement placé dans cet autre cluster.

## 6.5.2 Évaluation

Dans cette section nous évaluons l'algorithme proposé, en le comparant aux algorithmes NMFR [YHD<sup>+</sup>12] et INGRES en termes de pureté de cluster, sur vingt bases de données.

### Base de données

Les bases de données que nous avons utilisées correspondent à celles contenant le plus d'éléments utilisées par les auteurs de [YHD<sup>+</sup>12]. Nous avons ici directement utilisé les matrices de similarités disponibles à l'adresse <http://users.ics.aalto.fi/rozyang/nmfr/index.shtml>. Pour chaque expérimentation, nous avons construit le graphe à partir de la matrice correspondante, en utilisant directement les valeurs de similarité contenues dans les matrices.

### Comparaisons

Pour ces comparaisons, en lieu et place des étapes GROW et HARVEST de l'algorithme 4, nous avons utilisé le  $p$ -Laplacien normalisé sur graphe proposé au chapitre 3, ainsi que l'algorithme de propagation de front utilisant l'adaptation de l'équation eikonale sur graphe, proposé par [DEL13] et présenté en section 6.2.4. Nous les avons comparés avec l'algorithme INGRES décrit précédemment, ainsi que l'algorithme "nonnegative matrix factorization using graph random walk" (NMFR) [YHD<sup>+</sup>12]. Le principe des nonnegative matrix factorization est de factoriser une matrice  $A \in \mathbb{R}_+^{n \times m}$  en deux matrices  $B \in \mathbb{R}_+^{n \times p}$  et  $C \in \mathbb{R}_+^{p \times m}$ , *i.e.*  $A = BC$ , avec  $p$  significativement plus petit que  $m$  ou  $n$  et où  $A$ ,  $B$  et  $C$  n'ont pas d'entrées négatives. On approxime généralement cette factorisation en utilisant une fonction de coût mesurant l'erreur entre  $A$  et  $BC$ .

Cette méthode possède des propriétés de clustering inhérentes. Supposons que  $A$  soit une matrice représentant des données (chaque colonne représente un élément, les lignes représentant les caractéristiques de chaque élément). La décomposition de  $A$  en  $BC$  peut être interprétée comme le fait que la matrice  $B$  représente les centroïdes de chaque cluster. Chaque vecteur colonne de la matrice  $A$  peut alors être vu comme la combinaison linéaire de chaque centroïde, où les facteurs de chaque centroïde sont donnés par la colonne

correspondante dans la matrice  $C$ . On peut donc considérer que la matrice  $C$  indique la probabilité d'appartenance d'un élément à un cluster.

Bases de données	taille	RAND	NMFR	INCRES	GpLPL	FM
YEAST	1.5K	32%	<b>55%</b>	54%	54%	53%
SEMEION	1.6K	13%	<b>94%</b>	93%	93%	93%
FAULTS	1.9K	34%	39%	41%	<b>43%</b>	41%
SEG	2.3K	16%	73%	58%	<b>77%</b>	68%
CORA	2.7K	30%	<b>47%</b>	46%	44%	35%
MIREX	3.1K	13%	43%	<b>45%</b>	43%	44%
CITeseer	3.3K	21%	44%	47%	<b>48%</b>	37%
WEBKB4	4.2K	39%	<b>63%</b>	59%	61%	54%
7SECTORS	4.6K	24%	<b>34%</b>	32%	29%	29%
SPAM	4.6K	60%	<b>69%</b>	<b>69%</b>	<b>69%</b>	<b>69%</b>
CURETGREY	5.6K	5%	28%	19%	<b>30%</b>	<b>30%</b>
OPTDIGITS	5.6K	12%	<b>98%</b>	97%	96%	<b>98%</b>
GISETTE	7.0K	50%	<b>94%</b>	<b>94%</b>	<b>94%</b>	93%
REUTERS	8.3K	45%	<b>77%</b>	76%	<b>77%</b>	74%
RCV1	9.6K	30%	54%	55%	<b>56%</b>	50%
PENDIGITS	11K	12%	87%	<b>89%</b>	<b>89%</b>	88%
PROTEIN	18K	46%	<b>50%</b>	46%	46%	48%
20NEWS	20K	6%	63%	<b>64%</b>	62%	59%
MNIST	70K	11%	97%	96%	88%	<b>98%</b>
SEISMIC	99K	50%	<b>59%</b>	56%	<b>59%</b>	55%

TABLE 6.7 – Comparaison des algorithmes en termes de pureté de cluster.

On peut aussi utiliser cette technique dans le cas d'une matrice d'adjacence d'un graphe  $S \in \mathbb{R}_+^{n \times n}$ , représentant les similarités entre chaque paire de sommets du graphe. Pour  $m$  clusters, on peut approximer la factorisation de cette matrice par une matrice indicatrice de cluster  $U \in \{0, 1\}^{n \times m} : S \approx UU^T$ , où  $U_{ik} = 1$  si le  $i^{\text{eme}}$  élément est assigné au  $k^{\text{eme}}$  cluster et  $U_{ik} = 0$  sinon. Cependant, l'espace de solutions étant discret, ce problème d'optimisation est difficile. Une façon répandue de relaxer ce problème est d'utiliser des contraintes de non-négativité et d'orthogonalité. On peut alors remplacer  $U$

par une matrice  $W$  où  $W_{ik} \geq 0$  et  $W^T W = I$ . Ce choix est motivé par le fait que chaque ligne de  $U$  ne possède qu'une entrée non nulle, qui vaut 1.

On peut alors utiliser la norme de Frobenius afin de définir une fonction de coût à optimiser. La fonction objective de NMF devient alors :  $\|S - WW^T\|_F^2$ . La plupart des méthodes de résolution de ce problème utilisent l'erreur des moindres carrés. Lors de la minimisation de  $\|S - \hat{S}\|_F^2$ , la matrice approximée idéale  $\hat{S}$  devrait être diagonale par bloc et la matrice  $S$  en entrée devrait avoir une distribution Gaussienne, avec beaucoup d'entrées non-nulles. Ce n'est évidemment pas le cas lorsque l'on utilise des données réelles, où la matrice  $S$  donnée en entrée est souvent creuse et n'encode que les similarités locales. Pour ces raisons, les auteurs de [YHD+12] proposent d'utiliser une version moins creuse de la matrice d'adjacence  $S$ , encodant des similarités non-locales, en utilisant les marches aléatoires sur graphes pour la lisser (voir [YHD+12] pour plus de détails et de justifications).

Afin de pouvoir nous comparer quantitativement à ces algorithmes, nous avons utilisé la mesure de pureté de cluster, définie telle que :

$$\text{Pureté} = \frac{1}{N} \sum_{r=1}^m \max_{1 < i < m} n_{r,i}, \quad (6.19)$$

où  $n_{r,i}$  correspond au nombre de points du  $r^{\text{ième}}$  cluster appartenant à la  $i^{\text{ième}}$  classe de la vérité terrain. Cette mesure revient à affecter, pour chaque cluster, la classe de la vérité terrain qui le recouvre le plus (avec  $n_{r,i}$ ), puis à calculer le taux de bonne classification. Un bon clustering correspond à une pureté la plus proche de 1 possible.

Nous fournissons les résultats de nos algorithmes, en termes de pureté de cluster, ainsi que ceux des algorithmes INGRES et NMFR au tableau 6.7. La troisième colonne de ce tableau (RAND) fournit une base de comparaison en termes de pureté de cluster, obtenu en assignant aléatoirement une classe à chaque point de la base de données. Les résultats pour RAND, NMFR et INGRES sont tirés des articles [YHD+12] et [BHL+14].

Nous avons aussi évalué nos algorithmes en termes de temps de calcul. Dans l'article [BHL+14] les auteurs proposent un schéma multigrid afin d'accélérer leur algorithme. Ici nous n'avons pas effectué d'optimisation afin d'accélérer les temps de calcul, que ce soit en termes d'implantation ou d'algorithme. Ces temps sont donc fournis ici à simple titre indicatif et ils pourraient être grandement améliorés. Nous fournissons au tableau 6.8 les temps de calcul moyens (sur 10 exécutions) nécessaires à nos algorithmes pour atteindre 95% des scores maximums que nous avons obtenus sur les bases SEG, 20NEWS et MNIST.

Bases de données	GpLPL	FM
SEG	67.9s	10.1s
20NEWS	4768.3s	168.1s
MNIST	291.1s	53.9s

TABLE 6.8 – Temps d’exécution.

## 6.6 Conclusion

Dans ce chapitre, nous avons rappelé le formalisme des équations d’évolution de courbes basées sur les ensembles de niveaux, ainsi que leur transposition sur graphe. Afin d’introduire l’algorithme d’évolution de courbe sur graphe que nous proposons, nous avons aussi rappelé le lien entre les équations d’évolution de courbes et l’équation eikonale. Nous avons utilisé ce lien pour rappeler la transposition de l’équation eikonale ainsi que l’algorithme de résolution de cette équation sur graphe pondéré de topologie arbitraire. Nous proposons d’utiliser cet algorithme et de le combiner à celui proposé, dans le cadre de la classification semi-supervisée. Nous avons aussi proposé une implantation GPU de cet algorithme, accélérant les calculs d’un ordre de grandeur.

Nous avons ensuite évalué cet algorithme ainsi que ceux proposés aux chapitres précédents, dans le cadre de la classification semi-supervisée, sur plusieurs bases de données de la littérature, en les comparant à l’algorithme MTV proposé par [BTUvB13]. Nous avons aussi appliqué ces algorithmes dans le cadre de l’aide à la décision pour la microscopie virtuelle.

Enfin, nous avons proposé d’utiliser ces algorithmes de classification semi-supervisée dans le cadre du clustering de base de données, en s’inspirant des travaux de [BHL<sup>+</sup>14]. Nous avons évalué ces algorithmes sur vingt bases de données et montrés que les résultats en termes de pureté de cluster étaient tout à fait comparables aux méthodes de l’état de l’art, telle que la méthode proposée par [BHL<sup>+</sup>14] ou encore celle basée sur les non-negative matrix factorization de [YHD<sup>+</sup>12].



# Conclusion générale





Dans ce manuscrit, nous avons exploité le cadre des EdPs sur graphes initié par [BEM07, LEB07, ELB08, TEL11, DEL13] afin de transcrire des outils mathématiques continus et étudier leur application au traitement de données sur graphe.

Ce manuscrit est divisé en trois parties distinctes.

Dans la première partie, nous avons rappelé les notations et définitions essentielles à la compréhension de ce manuscrit.

Dans la deuxième partie, nous avons tout d'abord présenté l'adaptation du  $p$ -Laplacien normalisé sur graphe, puis nous avons proposé un opérateur Laplacien à coefficients variables permettant de retrouver les différents opérateurs introduits dans la première partie. Nous avons illustré le potentiel et l'intérêt de ces deux adaptations sur graphe à travers diverses applications, tel que le filtrage de données, la segmentation d'image et de base de données, ou encore l'inpainting sur des images et des nuages de points 3D. Nous avons ensuite proposé une adaptation de l'équation de Poisson sur graphe, en utilisant le Laplacien infini avec termes de gradients introduit précédemment. Nous avons montré que l'utilisation de cet opérateur mène à une équation hybride entre l'équation de Poisson infini et l'équation de Hamilton-Jacobi. Nous avons ensuite illustré son comportement dans le cadre du calcul de distance généralisée sur différents types de données discrètes (images, nuages de points 3D et bases de données). Nous avons ensuite proposé un algorithme de segmentation de données sur graphe basé sur cette formulation, que nous avons évalué quantitativement.

Dans la troisième et dernière partie, nous avons premièrement proposé une formulation multi-classes de l'équation d'évolution de courbe sur un graphe. Nous avons ensuite proposé d'évaluer quantitativement cette formulation, l'algorithme de propagation de front sur graphe reposant sur l'équation eikonale [DEL13], ainsi que les opérateurs proposés à la deuxième partie de ce manuscrit, dans le cadre de la classification semi-supervisée et du clustering de données.

Nous résumons dans cette conclusion les principales contributions de nos travaux :

## Une nouvelle famille de $p$ -Laplacien sur graphe

Cette partie est découpée en trois chapitres, où nous avons tout d'abord proposé une adaptation du  $p$ -Laplacien normalisé sur graphe, puis une formulation du  $p$ -Laplacien avec termes de gradients unifiant les différents opérateurs proposés par [BEM07, LEB07, ELB08, TEL11]. Enfin, nous avons utilisé cette dernière formulation dans le cadre du calcul de distance généralisée sur graphe, à travers l'équation de Poisson.

### **$p$ -Laplacien normalisé sur graphe**

Dans cette partie, nous avons tout d'abord proposé une adaptation du  $p$ -Laplacien normalisé sur graphe pondéré de topologie arbitraire en utilisant le cadre des EdPs. Cette adaptation nous a permis d'introduire une nouvelle classe de  $p$ -Laplacien sur graphe sous la forme d'une non-divergence. Il peut être considéré comme l'analogue discret du  $p$ -Laplacien des jeux défini en continu et s'exprime sous la forme d'une combinaison linéaire du 1-Laplacien, du Laplacien infini et du Laplacien sur graphe. Cet opérateur est obtenu à partir d'opérateurs statistiques transposés dans le cadre des graphes. Nous avons montré les relations entre l'opérateur proposé sur graphe et certains opérateurs différentiels locaux et non-locaux, ainsi qu'avec les jeux de type Tug-of-War. Comme beaucoup de problèmes en traitement d'images sont formulés comme des problèmes d'interpolations, nous avons étudié le problème de Dirichlet associé à cet opérateur et montré l'existence et l'unicité de la solution à cette équation. Nous avons aussi étudié l'équation de diffusion associée à cet opérateur et montré que le processus de résolution numérique de cette équation permet de retrouver divers filtres utilisés en traitement d'image et de les étendre au domaine des graphes.

### **$p$ -Laplacien et Laplacien infini avec termes de gradients**

Nous avons ensuite proposé un opérateur  $p$ -Laplacien et Laplacien infini défini comme une combinaison de termes de gradients à coefficients variables. Nous avons montré que cet opérateur est une généralisation des opérateurs défini sur graphe dans le cadre des EdPs. En effet les cas particuliers de cet opérateur permettent de retrouver les  $p$ -Laplacien anisotropes et le Laplacien infini, mais aussi les opérateurs de gradients morphologiques [ELB08, TEL11]. Nous avons aussi montré que cet opérateur permet de retrouver différents schémas de discrétisation existants, dans le cas d'opérateurs locaux et non-locaux. Cet opérateur unifie les formulations discrètes du  $p$ -Laplacien, du Laplacien infini, du  $p$ -Laplacien normalisé et du  $p$ -Laplacien non-local sur graphe, que ce soit sur des domaines réguliers ou irréguliers. Comme pour le  $p$ -Laplacien normalisé, nous avons étudié le problème de Dirichlet ainsi que l'équation de diffusion associée à cet opérateur. Nous avons montré l'existence et l'unicité de la solution au problème de Dirichlet et proposé et étudié un schéma numérique de résolution à l'équation de diffusion. Nous avons montré que des cas particuliers de ce processus permettent de retrouver des filtres de diffusion ainsi que des filtres morphologiques. Nous avons ensuite appliqué la résolution de ces deux processus au traitement d'image et de donnée, en illustrant leurs applications dans les tâches de filtrage, de segmentation et d'inpainting sur différents types

de données, tels que les images, les nuages de points 3D ou encore les bases de données.

### **Équation de Poisson infini et de Hamilton-Jacobi non-locale discrète**

Enfin, dans le dernier chapitre de cette partie, nous avons proposé une adaptation de l'équation de Poisson infini sur graphe, en utilisant la formulation du Laplacien infini avec termes de gradients sur graphe introduite précédemment. Nous avons fait le lien entre cette équation et la version biaisée du Tug-of-War. Nous avons montré que l'utilisation de cette formulation permet de retrouver une équation hybride de Poisson infini et de Hamilton-Jacobi. Nous avons aussi montré le lien entre cette version de l'équation de Poisson et l'adaptation de l'équation eikonale sur graphe pondéré [DEL13]. Notre but dans ce chapitre était d'utiliser cette extension de l'équation de Poisson afin de calculer des distances sur tout type de données discrètes pouvant être représentées par un graphe pondéré. Nous avons montré, à travers différentes expérimentations, que cette formulation peut être utilisée pour la résolution de différentes applications en traitement d'image, de nuages de point 3D et de données de grandes dimensions. En utilisant la distance générée par la résolution de cette équation, nous avons proposé un algorithme de segmentation de données sur graphe, que nous avons évalué quantitativement en utilisant la base de données d'images GrabCut pour le comparer à plusieurs algorithmes de l'état de l'art.

### **Équations d'ensembles de niveaux pour le clustering et la classification de données**

Dans cette dernière partie, nous avons tout d'abord proposé une formulation, basée sur la transposition des ensembles de niveaux sur graphes, permettant de faire évoluer plusieurs courbes à la fois sur un graphe. À partir de cette formulation, nous avons proposé d'utiliser le schéma de résolution proposé par [TEL11], ainsi qu'un algorithme combinant la propagation de front sur graphe [DEL13] et la formulation proposée, dans le cadre de la classification semi-supervisée, afin d'accélérer les temps de calcul et d'améliorer les résultats en termes de taux de classification. Nous avons aussi proposé une implantation GPU de cet algorithme, accélérant les calculs d'un ordre de grandeur. Dans ce cadre, nous avons proposé d'évaluer cet algorithme en utilisant différentes bases de données de l'état de l'art. Nous l'avons comparé aux formulations proposées dans la partie précédente, ainsi qu'aux formulations existantes sur graphes ( $p$ -Laplacien sur graphe, propagation de front sur graphe [ELB08, DEL13]) où la classification de données a été suggérée par les auteurs mais jamais évaluée.

Afin d'avoir une mesure étalon, nous avons comparé toutes ces formulations à l'algorithme MTV récemment proposé par [BTUvB13]. Nous avons aussi appliqué ces algorithmes à la classification de données dans le cadre de la microscopie virtuelle.

Nous avons ensuite proposé un algorithme de clustering, inspiré par les récents travaux de [BHL<sup>+</sup>14]. Celui-ci permet d'utiliser des algorithmes de classification transductive dans un cadre non-supervisé. Nous avons repris son fonctionnement afin d'utiliser les algorithmes de classification semi-supervisée que nous avons proposé, permettant ainsi d'effectuer du clustering avec ceux-ci. Ici aussi, nous avons évalué quantitativement ces algorithmes en les comparant avec l'algorithme INGRES proposé par [BHL<sup>+</sup>14] ainsi qu'à la méthode NMFR proposée par [YHD<sup>+</sup>12], en termes de pureté de cluster, sur vingt bases de données.

## Perspectives

Dans les chapitres sur le  $p$ -Laplacien normalisé sur graphe et sur le  $p$ -Laplacien et Laplacien infini avec termes de gradients, nous avons montré les liens entre les opérateurs que nous proposons et certains opérateurs continus locaux et non-locaux, en les discrétisant. Il serait intéressant ici d'utiliser des graphes particuliers en faisant tendre le nombre de sommets vers l'infini afin d'étudier la consistance continue des solutions, *i.e.* est-ce que les solutions aux équations de diffusions et aux problèmes de Dirichlet convergent vers les solutions continues de ces équations ?

L'un des aspects sur lequel nous pourrions aussi travailler est l'accélération de la résolution des équations que nous avons proposées. En effet, afin de résoudre ces équations, nous avons proposé des schémas de discrétisation temporelle simples. Nous pourrions ici utiliser des schémas de discrétisation d'ordre supérieurs, ou encore des schémas de résolution implicites, afin d'améliorer la convergence des algorithmes vers la solution. Une autre manière d'accélérer les schémas de résolution serait de rendre la représentation GPU du graphe plus efficace en termes de mémoire, afin de ne plus avoir de restriction quant à la régularité du graphe. L'accélération de ces algorithmes nous permettrait de passer à l'échelle et ainsi de les appliquer à des bases de données, par exemple en microscopie virtuelle, où les bases de données réelles peuvent contenir des millions d'objets.

Nous avons mis l'accent sur l'utilisation des opérateurs présentés dans le cadre de la classification semi-supervisée et du clustering de données sur graphe. Cependant, plusieurs problématiques restent ouvertes, telle que la construction optimale du graphe : on pourrait, par exemple, utiliser des

méthodes d'apprentissage de métriques et les utiliser lors de la construction d'un graphes des  $k$ -ppv. On peut aussi penser à des améliorations de l'algorithme de clustering que nous avons utilisé : au lieu d'utiliser une partition initiale aléatoire, on pourrait utiliser une méthode non-aléatoire, telle que les farthest points, pour initialiser la partition. On pourrait aussi améliorer les résultats, ou les rendre plus stable, en utilisant l'intersection de différents résultats sur une même base de données comme partition initiale (incomplète), et utiliser un algorithme de classification transductive afin de partitionner entièrement l'ensemble de données.



# Liste des publications

- [BTEL14] Pierre BUYSENS, Matthieu TOUTAIN, Abderrahim ELMOATAZ et Olivier LÉZORAY : Eikonal-based vertices growing and iterative seeding for efficient graph-based segmentation. *In IEEE International Conference on Image Processing (ICIP 2014)*, pages 5–pp, 2014.
- [ETTar] Abderrahim ELMOATAZ, Matthieu TOUTAIN et Daniel TENBRINCK : On the  $p$ -Laplacian and  $\infty$ -Laplacian on graphs with Applications in Image and Data Processing. *SIAM Journal on Imaging Sciences (SIIMS)*, pages 1–37, to appear.
- [ETew] Abderrahim ELMOATAZ et Matthieu TOUTAIN : Normalized  $p$ -Laplacian on Weighted Graphs with Applications in Image Processing and Machine Learning. *Applied and computational harmonic analysis*, pages 1–42, under review.
- [SET15] Ahcene SADI, Abderrahim ELMOATAZ et Matthieu TOUTAIN : Nonlocal pde morphology : a generalized shock operator on graph. *Signal, Image and Video Processing*, pages 1–8, 2015.
- [TDEL13] Matthieu TOUTAIN, Xavier DESQUESNES, Abderrahim ELMOATAZ et Olivier LEZORAY : A unified geometric model for virtual slide images processing. *In International Workshop on Adaptation and Learning in Control and Signal Processing (IFAC)*, pages 629–634, 2013.
- [TEDPar] Matthieu TOUTAIN, Abderrahim ELMOATAZ, Xavier DESQUESNES et Jean-Hugues PRUVOT : A unified geometric model for virtual slide image processing and classification. *Journal of Selected Topics in Signal Processing*, pages 1–11, to appear.
- [TEL14] Matthieu TOUTAIN, Abderrahim ELMOATAZ et Olivier LÉZORAY : Geometric pdes on weighted graphs for semi-supervised classifica-



tion. *In Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 231–236. IEEE, 2014.

- [TELM15] Matthieu TOUTAIN, Abderrahim ELMOATAZ, François LOZES et Alamin MANSOURI : Non-local discrete  $\infty$ -poisson and hamilton jacobi equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2015.

# References

- [AA98] E. Alpaydin and F. Alimoglu. UCI pen-based recognition of handwritten digits data set, 1998.
- [AC04] Pablo Andrés Arbeláez and Laurent D Cohen. Energy partitions and image segmentation. *Journal of Mathematical Imaging and Vision*, 20(1-2) :43–57, 2004.
- [AFCS11] Pablo Arias, Gabriele Facciolo, Vicent Caselles, and Guillermo Sapiro. A variational framework for exemplar-based image inpainting. *International journal of computer vision*, 93(3) :319–347, 2011.
- [AGLM93] Luis Alvarez, Frédéric Guichard, Pierre-Louis Lions, and Jean-Michel Morel. Axioms and fundamental equations of image processing. *Archive for rational mechanics and analysis*, 123(3) :199–257, 1993.
- [AK98] E. Alpaydin and C. Kaynak. UCI optical recognition of handwritten digits data set, 1998.
- [AMRT08] F Andreu, JM Mazón, JD Rossi, and J Toledo. A nonlocal p-laplacian evolution equation with neumann boundary conditions. *Journal de mathématiques pures et appliquées*, 90(2) :201–227, 2008.
- [Aro67] Gunnar Aronsson. Extension of functions satisfying lipschitz conditions. *Arkiv för Matematik*, 6(6) :551–561, 1967.
- [AVMRTM10] Fuensanta Andreu-Vaillo, José M Mazón, Julio D Rossi, and J Julián Toledo-Melero. *Nonlocal diffusion problems*, volume 165. American Mathematical Society, 2010.
- [BBW07] Michael Breuß, Bernhard Burgeth, and Joachim Weickert. Anisotropic continuous-scale morphology. In *Pattern Recognition and Image Analysis*, pages 515–522. Springer, 2007.

- [BCM05] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.
- [BCM08] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Nonlocal image and movie denoising. *International journal of computer vision*, 76(2) :123–139, 2008.
- [BCOS01] Marcelo Bertalmio, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. Variational problems and partial differential equations on implicit surfaces. *Journal of Computational Physics*, 174(2) :759–780, 2001.
- [BEM07] Sébastien Bogleux, Abderrahim Elmoataz, and Mahmoud Melkemi. Discrete regularization on weighted graphs for image and mesh filtering. In *Scale Space and Variational Methods in Computer Vision*, pages 128–139. Springer, 2007.
- [BEM09] Sébastien Bogleux, Abderrahim Elmoataz, and Mahmoud Melkemi. Local and nonlocal discrete regularization on weighted graphs for image and mesh processing. *International Journal of Computer Vision*, 84(2) :220–236, 2009.
- [Ber05] Gilles Bertrand. On topological watersheds. *Journal of Mathematical Imaging and Vision*, 22(2-3) :217–230, 2005.
- [BF12] Andrea L Bertozzi and Arjuna Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling & Simulation*, 10(3) :1090–1118, 2012.
- [BHL<sup>+</sup>14] Xavier Bresson, Huiyi Hu, Thomas Laurent, Arthur Szlam, and James von Brecht. An incremental reseeding strategy for clustering. *arXiv preprint arXiv :1406.3837*, 2014.
- [BJ01] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.
- [BL93] John W Barrett and WB Liu. Finite element approximation of the p-laplacian. *mathematics of computation*, 61(204) :523–537, 1993.

- [BM92] Roger W Brockett and Petras Maragos. Evolution equations for continuous-scale morphology. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 3, pages 125–128. IEEE, 1992.
- [BM05] Alain Bensoussan and José-Luis Menaldi. Difference equations on weighted graphs. *Journal of Convex Analysis*, 12(1) :13–44, 2005.
- [BNS06] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization : A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7 :2399–2434, 2006.
- [BS09] Xue Bai and Guillermo Sapiro. Geodesic matting : A framework for fast interactive image and video segmentation and matting. *International Journal of Computer Vision*, 82(2) :113–132, 2009.
- [BTEL14] Pierre Buysens, Matthieu Toutain, Abderrahim Elmoataz, and Olivier Lézoray. Eikonal-based vertices growing and iterative seeding for efficient graph-based segmentation. In *IEEE International Conference on Image Processing (ICIP 2014)*, pages 5–pp, 2014.
- [BTUvB13] Xavier Bresson, Laurent Thomas, David Uminsky, and James H. von Brecht. Multiclass total variation clustering. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *NIPS*, pages 1421–1429, 2013.
- [CBNC09] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts : Minimum spanning forests and the drop of water principle. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(8) :1362–1374, 2009.
- [CED14] Abdallah El Chakik, Abderrahim Elmoataz, and Xavier Desquesnes. Mean curvature flow on graphs for image and manifold restoration and enhancement. *Signal Processing*, 105 :449–463, 2014.
- [CES13] Abdallah El Chakik, Abderrahim Elmoataz, and Ahcene Sadi. On the mean curvature flow on graphs with applications in

- image and manifold processing. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 697–704, 2013.
- [CFL28] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *Math. Ann.*, 100 :32–74, 1928.
- [CGNT11] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watershed : A unifying graph-based optimization framework. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(7) :1384–1399, 2011.
- [Chu97] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.
- [CLM12] Antonin Chambolle, Erik Lindgren, and Régis Monneau. A hölder infinity laplacian. *ESAIM : Control, Optimisation and Calculus of Variations*, 18(03) :799–835, 2012.
- [CM06] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1) :53–94, 2006.
- [COS01] Tony F Chan, Stanley Osher, and Jianhong Shen. The digital tv filter and nonlinear denoising. *Image Processing, IEEE Transactions on*, 10(2) :231–241, 2001.
- [CT11] Fatih Calakli and Gabriel Taubin. Ssd : Smooth signed distance surface reconstruction. In *Computer Graphics Forum*, volume 30, pages 1993–2002. Wiley Online Library, 2011.
- [CV+01] Tony F Chan, Luminita Vese, et al. Active contours without edges. *Image processing, IEEE transactions on*, 10(2) :266–277, 2001.
- [DEL11] X. Desquesnes, A. Elmoataz, and O. Lézoray. PDEs level sets on weighted graphs. In *International Conference on Image Processing (IEEE)*, pages 3377 – 3380, 2011.
- [DEL13] Xavier Desquesnes, Abderrahim Elmoataz, and Olivier Lézoray. Eikonal equation adaptation on weighted graphs : fast geometric diffusion process for local and non-local image and data processing. *Journal of Mathematical Imaging and Vision*, 46(2) :238–257, 2013.

- [Drá07] Pavel Drábek. The p-laplacian–mascot of nonlinear analysis. *Acta Math. Univ. Comenianae*, 76(1) :85–98, 2007.
- [EDL12] Abderrahim Elmoataz, Xavier Desquesnes, and Olivier Lézoray. Non-local morphological pdes and-laplacian equation on graphs with applications in image processing and machine learning. *Selected Topics in Signal Processing, IEEE Journal of*, 6(7) :764–779, 2012.
- [EDLL11] Abderrahim Elmoataz, Xavier Desquesnes, Zakaria Lakhdari, and Olivier Lezoray. On the infinity laplacian equation on graph with applications to image and manifolds processing. In *International Conference on Approximation Methods and Numerical Modelling in Environment and Natural Resources*, pages 7–pages, 2011.
- [EDLL14] Abderrahim Elmoataz, Xavier Desquesnes, Zakaria Lakhdari, and Olivier Lézoray. Nonlocal infinity laplacian equation on graphs with applications in image processing and machine learning. *Mathematics and Computers in Simulation*, 102 :153–163, 2014.
- [EL<sup>+</sup>99] Alexei Efros, Thomas K Leung, et al. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.
- [ELB08] Abderrahim Elmoataz, Olivier Lezoray, and Sébastien Boughoux. Nonlocal discrete regularization on weighted graphs : a framework for image and manifold processing. *Image Processing, IEEE Transactions on*, 17(7) :1047–1060, 2008.
- [FSdAL04] Alexandre X Falcão, Jorge Stolfi, and Roberto de Alencar Lotufo. The image foresting transform : Theory, algorithms, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(1) :19–29, 2004.
- [FVGS13] M Falcone, S Finzi Vita, T Giorgi, and RG Smits. A semi-lagrangian scheme for the game p-laplacian via p-averaging. *Applied Numerical Mathematics*, 73 :63–80, 2013.
- [GEL11] Mahmoud Ghoniem, Abderrahim Elmoataz, and Olivier Lezoray. Discrete infinity harmonic functions : towards a

- unified interpolation framework on graphs. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 1361–1364. IEEE, 2011.
- [GGS<sup>+</sup>06] Lena Gorelick, Meirav Galun, Eitan Sharon, Ronen Basri, and Achi Brandt. Shape representation and classification using the poisson equation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12) :1991–2005, 2006.
- [GM01] Frederic Guichard and Jean-Michel Morel. A note on two classical shock filters and their asymptotics. In *Scale-Space and Morphology in Computer Vision*, pages 75–84. Springer, 2001.
- [GO07] Guy Gilboa and Stanley Osher. Nonlocal linear image regularization and supervised segmentation. *Multiscale Modeling & Simulation*, 6(2) :595–630, 2007.
- [GO08] Guy Gilboa and Stanley Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3) :1005–1028, 2008.
- [GP10] Leo J Grady and Jonathan Polimeni. *Discrete calculus : Applied analysis on graphs for computational science*. Springer Science & Business Media, 2010.
- [Gra06] Leo Grady. Random walks for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11) :1768–1783, 2006.
- [GSZ02] Guy Gilboa, Nir A Sochen, and Yehoshua Y Zeevi. Regularized shock filters and complex diffusion. In *Computer Vision—ECCV 2002*, pages 399–413. Springer, 2002.
- [HB10] Matthias Hein and Thomas Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse pca. In *Advances in Neural Information Processing Systems*, pages 847–855, 2010.
- [Hei94] Henk JAM Heijmans. Morphological image operators. *Advances in Electronics and Electron Physics Suppl., Boston : Academic Press, / c1994*, 1, 1994.
- [Hir03] Anil N Hirani. *Discrete exterior calculus*. PhD thesis, California Institute of Technology, 2003.

- [HM06] Matthias Hein and Markus Maier. Manifold denoising. In *Advances in neural information processing systems*, pages 561–568, 2006.
- [HNTV92] HJAM Heumans, P Nacken, A Toet, and Luc Vincent. Graph morphology. *Journal of visual communication and image representation*, 3(1) :24–38, 1992.
- [Hul94] Jonathan J. Hull. A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(5) :550–554, 1994.
- [HVG11] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2) :129–150, 2011.
- [JO05] Maarten H Jansen and Patrick J Oonincx. *Second generation wavelets and applications*. Springer Science & Business Media, 2005.
- [Kaw90] Bernhard Kawohl. On a family of torsional creep problems. *J. reine angew. Math*, 410(1) :1–22, 1990.
- [Kaw11] Bernd Kawohl. Variations on the p-laplacian. *Nonlinear Elliptic Partial Differential Equations, Eds. D. Bonheure, P. Takac et al., Contemporary Mathematics*, 540 :35–46, 2011.
- [KB75] Henry P Kramer and Judith B Bruckner. Iterations of a non-linear transformation for enhancement of digital images. *Pattern recognition*, 7(1) :53–58, 1975.
- [KBCL09] Miklós Kurucz, András A Benczúr, Károly Csalogány, and László Lukács. Spectral clustering in social networks. In *Advances in Web Mining and Web Usage Analysis*, pages 1–20. Springer, 2009.
- [KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.
- [KDTN00] Daniel Keysers, Jörg Dahmen, Thomas Theiner, and Hermann Ney. Experiments with an extended tangent distance. In *ICPR*, pages 2038–2042, 2000.



- [Ker04] Charles Kervrann. An adaptive window approach for image smoothing and structures preserving. In Tomás Pajdla and Jiri Matas, editors, *ECCV (3)*, volume 3023 of *Lecture Notes in Computer Science*, pages 132–144. Springer, 2004.
- [LC10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>, 2010.
- [LC11] Rongjie Lai and Tony F Chan. A framework for intrinsic image processing on surfaces. *Computer vision and image understanding*, 115(12) :1647–1661, 2011.
- [LC12] Young-Su Lee and Soon-Yeong Chung. Extinction and positivity of solutions of the p-laplacian evolution equation on networks. *Journal of Mathematical Analysis and Applications*, 386(2) :581–592, 2012.
- [LEB07] Olivier Lezoray, Abderrahim Elmoataz, and Sébastien Bougleux. Graph regularization for color image processing. *Computer Vision and Image Understanding*, 107(1) :38–55, 2007.
- [LEC03] O. Lezoray, A. Elmoataz, and H. Cardot. A color object recognition scheme : application to cellular sorting. *Machine Vision and Applications*, 14 :166–171, 2003.
- [LEL13] François Lozes, Abderrahim Elmoataz, and Olivier Lézoray. Morphological pdes on graphs for filtering and inpainting of point clouds. In *Image and Signal Processing and Analysis (ISPA), 2013 8th International Symposium on*, pages 542–547. IEEE, 2013.
- [LEL14a] F. Lozes, A. Elmoataz, and O. Lezoray. Partial difference operators on weighted graphs for image processing on surfaces and point clouds. *Image Processing, IEEE Transactions on*, 23(9) :3896–3909, Sept 2014.
- [LEL14b] François Lozes, Abderrahim Elmoataz, and Olivier Lézoray. Partial difference operators on weighted graphs for image processing on surfaces and point clouds. 2014.
- [Lin06] Peter Lindqvist. *Notes on the p-Laplace equation*. Univ., 2006.

- [Mey94] Fernand Meyer. Topographic distance and watershed lines. *Signal processing*, 38(1) :113–125, 1994.
- [MPR12] Juan J Manfredi, Mikko Parviainen, and Julio D Rossi. Dynamic programming principle for tug-of-war games with noise. *ESAIM : Control, Optimisation and Calculus of Variations*, 18(01) :81–90, 2012.
- [MRT] JOSÉ M MAZÓN, JULIO D ROSSI, and JULIÁN TOLEDO. Fractional p-laplacian evolution equations. *RN*, 1 :2p.
- [MS89] F Meyer and J Serra. Contrasts and activity lattice. *Signal Processing*, 16(4) :303–317, 1989.
- [Neu06] John M Neuberger. Nonlinear elliptic partial difference equations on graphs. *Experimental Mathematics*, 15(1) :91–107, 2006.
- [NVI11] NVIDIA Corporation. *NVIDIA CUDA C Programming Guide*, June 2011.
- [Obe13] Adam M Oberman. Finite difference methods for the infinity laplace and p-laplace equations. *Journal of Computational and Applied Mathematics*, 254 :65–80, 2013.
- [OF03] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2003.
- [OR90] Stanley Osher and Leonid I Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis*, 27(4) :919–940, 1990.
- [OS88] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed : algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1) :12–49, 1988.
- [OS00] Stanley Osher and Jianhong Shen. Digitized pde method for data restoration. *Analytic-Computational Methods in Applied Mathematics*, 698, 2000.
- [PBC08] Gabriel Peyré, Sébastien Bogleux, and Laurent Cohen. Non-local regularization of inverse problems. In *Computer Vision–ECCV 2008*, pages 57–68. Springer, 2008.

- [PC11] Jea-Hyun Park and Soon-Yeong Chung. Positive solutions for discrete boundary value problems involving the  $p$ -laplacian with potential terms. *Computers & Mathematics with Applications*, 61(1) :17–29, 2011.
- [PLD05] Hanchuan Peng, Fulmi Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8) :1226–1238, 2005.
- [PPS10] Yuval Peres, Gábor Pete, and Stephanie Somersille. Biased tug-of-war, the biased infinity laplacian, and comparison with exponential cones. *Calculus of Variations and Partial Differential Equations*, 38(3-4) :541–564, 2010.
- [PS<sup>+</sup>08] Yuval Peres, Scott Sheffield, et al. Tug-of-war with noise : A game-theoretic view of the  $p$ -laplacian. *Duke Mathematical Journal*, 145(1) :91–120, 2008.
- [PSSW09] Yuval Peres, Oded Schramm, Scott Sheffield, and David Wilson. Tug-of-war and the infinity laplacian. *Journal of the American Mathematical Society*, 22(1) :167–210, 2009.
- [RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut : Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3) :309–314, 2004.
- [RM08] Steven J Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics*, 227(3) :1943–1961, 2008.
- [Rud14] Matthew Rudd. Statistical exponential formulas for homogeneous diffusion. *arXiv preprint arXiv :1403.1853*, 2014.
- [SB11] Carola-bibiane Schönlieb and Andrea Bertozzi. Unconditionally stable schemes for higher order inpainting. *Communications in Mathematical Sciences*, 9(2) :413–457, 2011.
- [SET15] Ahcene Sadi, Abderrahim Elmoataz, and Matthieu Toutain. Nonlocal pde morphology : a generalized shock operator on graph. *Signal, Image and Video Processing*, pages 1–8, 2015.

- [SG07] Ali Kemal Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [SK07] Alon Spira and Ron Kimmel. Geometric curve flows on parametric manifolds. *Journal of Computational Physics*, 223(1) :235–249, 2007.
- [SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8) :888–905, 2000.
- [Sta03] Jos Stam. Flows on surfaces of arbitrary topology. *ACM Transactions On Graphics (TOG)*, 22(3) :724–731, 2003.
- [Tad12] Eitan Tadmor. A review of numerical methods for nonlinear partial differential equations. *Bulletin of the American Mathematical Society*, 49(4) :507–554, 2012.
- [TDEL13] Matthieu Toutain, Xavier Desquesnes, Abderrahim Elmoataz, and Olivier Lezoray. A unified geometric model for virtual slide images processing. In *International Workshop on Adaptation and Learning in Control and Signal Processing (IFAC)*, pages 629–634, Caen, France, 2013.
- [TEL08] Vinh-Thong Ta, Abderrahim Elmoataz, and Olivier L  zoray. Partial difference equations over graphs : Morphological processing of arbitrary discrete data. In *Computer Vision–ECCV 2008*, pages 668–680. Springer, 2008.
- [TEL11] Vinh-Thong Ta, Abderrahim Elmoataz, and Olivier L  zoray. Nonlocal pdes-based morphology on weighted graphs for image and data processing. *Image Processing, IEEE transactions on*, 20(6) :1504–1516, 2011.
- [VDBS94] Rein Van Den Boomgaard and Arnold Smeulders. The morphological structure of images : The differential equations of morphological scale-space. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(11) :1101–1113, 1994.
- [vGGOB14] Yves van Gennip, Nestor Guillen, Braxton Osting, and Andrea L Bertozzi. Mean curvature, threshold dynamics,

- and phase field theory on finite graphs. *Milan Journal of Mathematics*, 82(1) :3–65, 2014.
- [VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416, 2007.
- [VS91] Luc Vincent and Pierre Soille. Watersheds in digital spaces : an efficient algorithm based on immersion simulations. *IEEE transactions on pattern analysis and machine intelligence*, 13(6) :583–598, 1991.
- [Wei03] Joachim Weickert. Coherence-enhancing shock filters. In *Pattern Recognition*, pages 1–8. Springer, 2003.
- [YHD<sup>+</sup>12] Zhirong Yang, Tele Hao, Onur Dikmen, Xi Chen, and Erkki Oja. Clustering by nonnegative matrix factorization using graph random walk. In *Advances in Neural Information Processing Systems*, pages 1079–1087, 2012.
- [ZMP04] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NIPS*, 2004.
- [ZS04] Dengyong Zhou and Bernhard Schölkopf. A regularization framework for learning from graph data. In *ICML workshop on statistical relational learning and Its connections to other fields*, volume 15, pages 67–68, 2004.
- [ZS06] Dengyong Zhou and Bernhard Schölkopf. Discrete regularization. In *Semi-Supervised Learning*, pages 221–232. MIT Press, 2006.

# Table des figures

1.1	Graphe pondéré . . . . .	14
1.2	Frontières sur graphe . . . . .	16
1.3	Graphe des $k$ -ppv . . . . .	19
1.4	Illustration graphe des $k$ -ppv bdd . . . . .	20
1.5	Graphe $\epsilon$ . . . . .	21
1.6	$\epsilon$ -voisinage sur une grille 2D, avec différentes métriques . . . . .	22
1.7	Illustration de différents patches d'une image . . . . .	23
1.8	Construction d'un graphe d'adjacence de région . . . . .	25
1.9	Construction d'un graphe à partir d'un maillage 3D . . . . .	25
2.1	Opérations morphologiques sur graphe . . . . .	37
3.1	Lissage local d'image avec $\Delta_{w,p}^N$ . . . . .	73
3.2	Filtrage de base de données utilisant l'opérateur $\Delta_{w,p}^N$ . . . . .	74
3.3	Segmentation d'image avec $\Delta_{w,p}^N$ . . . . .	77
3.4	Classification semi-supervisée avec $\Delta_{w,p}^N$ . . . . .	78
3.5	Inpainting de texture avec $\Delta_{w,p}^N$ . . . . .	78
3.6	Inpainting local et non-local avec $\Delta_{w,p}^N$ . . . . .	79
3.7	Inpainting d'image naturelles avec $\Delta_{w,p}^N$ . . . . .	79
4.1	Processus morphologiques avec $\mathcal{L}_{w,p}$ . . . . .	104
4.2	Filtrage d'image avec $\mathcal{L}_{w,p}$ . . . . .	105
4.3	Lissage des couleurs d'un nuage de points avec $\mathcal{L}_{w,p}$ . . . . .	108
4.4	Filtre de choc sur une image de QR code en utilisant $\mathcal{L}_{w,p}$ . . . . .	109
4.5	Filtre de choc sur une image naturelle en utilisant $\mathcal{L}_{w,p}$ . . . . .	110
4.6	Filtre de choc sur un nuage de point 3D en utilisant $\mathcal{L}_{w,p}$ . . . . .	111
4.7	Filtre de choc sur un graphe d'adjacence de région en utilisant $\mathcal{L}_{w,p}$ . . . . .	112

4.8	Contours actifs avec $\mathcal{L}_{w,p}$ . . . . .	113
4.9	Segmentation interactive d'images avec $\mathcal{L}_{w,p}$ . . . . .	115
4.10	Segmentation interactive de carte de régions avec $\mathcal{L}_{w,p}$ . . . . .	116
4.11	Classification semi-supervisée de données avec $\mathcal{L}_{w,p}$ . . . . .	117
4.12	Inpainting d'images avec $\mathcal{L}_{w,p}$ . . . . .	118
4.13	Inpainting de nuage de point 3D avec $\mathcal{L}_{w,p}$ . . . . .	119
5.1	Calcul de distance sur une image synthétique . . . . .	130
5.2	Génération de distance sur une image de forme . . . . .	131
5.3	Calcul de distance sur une image naturelle . . . . .	133
5.4	Calcul de distance sur un nuage de points 3D . . . . .	134
5.5	Calcul de distance sur plusieurs nuages de points 3D . . . . .	135
5.6	Segmentation d'image à partir du calcul de distance . . . . .	136
5.7	Segmentation d'image texturée à partir du calcul de distance . . . . .	137
5.8	Classification de données à partir du calcul de distance . . . . .	139
5.9	Taux de convergence de l'algorithme de calcul de distance . . . . .	140
6.1	Représentation et accès aux éléments du graphe algorithmique GPU	153
6.2	Comparaison des trois fonctions de potentiels utilisées avec l'équation eikonale pour la classification semi-supervisée . . . . .	159
6.3	Comparaison des estimations locales et globales du paramètre $\sigma$ en termes de taux de classification . . . . .	160
6.4	Comparaison des estimations locales et globales du paramètre $\sigma$ en termes d'écart type du taux de classification . . . . .	161
6.5	Taux de classification obtenu en utilisant $\Delta_{w,p}^N$ avec la base de données PENDIGITS, pour $1 < p \leq 2$ . . . . .	163
6.6	Détails de la figure 6.5 pour $1 < p \leq 1.1$ . . . . .	164
6.7	Exemples de lames cytologiques. . . . .	165
6.8	Exemples d'images cytologiques. . . . .	166
6.9	Galerie de noyaux segmentés et classifiés. . . . .	167
6.10	Taux de classification de la base de cellules séreuses. . . . .	168

# Liste des Algorithmes

1	Propagation de front sur graphe . . . . .	148
2	Évolution de courbe multi-classes . . . . .	155
3	ParallelLS . . . . .	155
4	Algorithme INCRES . . . . .	171



## EdP géométriques pour le traitement et la classification de données sur graphes

**Résumé :** Les équations aux dérivées partielles (EDPs) jouent un rôle clé dans la modélisation mathématique des phénomènes en sciences appliquées. En particulier, en traitement et analyse d'image et en vision par ordinateur, les EDPs géométriques ont été utilisées avec succès pour résoudre différents problèmes, tels que la restauration, la segmentation, l'inpainting, etc. De nos jours, de plus en plus de données sont collectées sous la forme de graphes ou réseaux, ou de fonctions définies sur ces réseaux. Il y a donc un intérêt à étendre les EDPs pour traiter des données irrégulières ou des graphes de topologies arbitraires. Les travaux de cette thèse s'inscrivent dans ce contexte. Ils traitent précisément des EDPs géométriques pour le traitement et la classification de données sur graphes. Dans une première partie, nous proposons une adaptation du  $p$ -Laplacien normalisé sur graphes pondérés de topologie arbitraire en utilisant le cadre des équations aux différences partielles (EdPs). Cette adaptation nous permet d'introduire une nouvelle classe de  $p$ -Laplacien sur graphe sous la forme d'une non-divergence. Nous introduisons aussi dans cette partie une formulation du  $p$ -Laplacien sur graphe définie comme une combinaison convexe de gradient. Nous montrons que cette formulation unifie et généralise différents opérateurs de différences sur graphe existants. Nous utilisons ensuite cet opérateur à travers l'équation de Poisson afin de calculer des distances généralisées sur graphe. Dans une deuxième partie, nous proposons d'appliquer les opérateurs sur graphes que nous avons proposés pour les tâches de classification semi-supervisée et de clustering, et de les comparer aux opérateurs sur graphes existants ainsi qu'à certaines méthodes de la littérature, telles que le Multiclass Total Variation clustering (MTV), le clustering par nonnegative matrix factorization (NMFR), ou encore la méthode INGRES.

**Mots-clés :** Équations aux différences, Laplacien, Traitement d'images – techniques numériques, Ensemble de niveaux, Méthode d', Informatique

### Geometric PdEs for data processing and classification on graphs

**Abstract :** Partial differential equations (PDEs) play a key role in the mathematical modelization of phenomena in applied sciences. In particular, in image processing and computer vision, geometric PDEs have been successfully used to solve many problems, such as image restoration, segmentation, inpainting, etc. Nowadays, more and more data are collected as graphs or networks, or functions defined on these networks. Knowing this, there is an interest to extend PDEs to process irregular data or graphs of arbitrary topology. The presented work follows this idea. More precisely, this work is about geometric partial difference equations (PdEs) for data processing and classification on graphs. In the first part, we propose a transcription of the normalized  $p$ -Laplacian on weighted graphs of arbitrary topology by using the framework of PdEs. This adaptation allows us to introduce a new class of  $p$ -Laplacian on graphs as a non-divergence form. In this part, we also introduce a formulation of the  $p$ -Laplacian on graphs defined as a convex combination of gradient terms. We show that this formulation unifies and generalizes many existing difference operators defined on graphs. Then, we use this operator with the Poisson equation to compute generalized distances on graphs. In the second part, we propose to apply the operators on graphs we defined, for the tasks of semi-supervised classification and clustering. We compare them to existing graphs operators and to some of the state of the art methods, such as Multiclass Total Variation clustering (MTV), clustering by non-negative matrix factorization (NMFR) or the INGRES method.

**Keywords:** Difference equations, Laplacian operator, Image processing – digital techniques, Level set methods, Data processing