Techniques variationnelles et calcul parallèle en imagerie : Estimation du flot optique avec luminosité variable en petits et larges déplacements

Diane Gilliocq-Hirtz

Laboratoire de Mathématiques, Informatique et Applications
Université de Haute Alsace
6 rue des frères Lumière
68093 Mulhouse

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE HAUTE ALSACE

École Doctorale de Mathématiques, Sciences de l'Information et de l'Ingénieur

Spécialité

Mathématiques Appliquées

Présentée par

Diane Gilliocq-Hirtz

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE HAUTE ALSACE

Techniques variationnelles et calcul parallèle en imagerie. Estimation du flot optique avec luminosité variable en petits et larges déplacements.

Soutenue le 7 juillet 2016 *devant le jury composé de :*

Rapporteur :	Frédéric Hecht	*Professeur, Université Pierre et Marie Curie*
Rapporteur :	Faker Ben Belgacem	*Professeur, Université de Compiègne*
Directeur de thèse :	Zakaria Belhachmi	*Professeur, Université de Haute Alsace*
Président du jury :	Pierre Charbonnier	*Directeur de recherche HDR, Cerema (Strasbourg)*
Invité :	Bernard Brighi	*Professeur, Université de Haute Alsace*
Invité :	Cornel Murea	*Maître de conférences HDR, Université de Haute Alsace*

Laboratoire de Mathématiques, Informatique et Applications
Université de Haute Alsace
6 rue des frères Lumière
68093 Mulhouse

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE HAUTE ALSACE

École Doctorale de Mathématiques, Sciences de l'Information et de l'Ingénieur

Spécialité

Mathématiques Appliquées

Présentée par

Diane Gilliocq-Hirtz

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE HAUTE ALSACE

Techniques variationnelles et calcul parallèle en imagerie. Estimation du flot optique avec luminosité variable en petits et larges déplacements.

Soutenue le 7 juillet 2016 *devant le jury composé de* :

Rapporteur :	Frédéric Hecht	*Professeur, Université Pierre et Marie Curie*
Rapporteur :	Faker Ben Belgacem	*Professeur, Université de Compiègne*
Directeur de thèse :	Zakaria Belhachmi	*Professeur, Université de Haute Alsace*
Président du jury :	Pierre Charbonnier	*Directeur de recherche HDR, Cerema (Strasbourg)*
Invité :	Bernard Brighi	*Professeur, Université de Haute Alsace*
Invité :	Cornel Murea	*Maître de conférences HDR, Université de Haute Alsace*

Résumé

Le travail présenté dans cette thèse porte sur l'estimation du flot optique par méthodes variationnelles en petits et en grands déplacements. Nous proposons un modèle basé sur la combinaison locale-globale à laquelle nous ajoutons la prise en compte des variations de la luminosité. La particularité de ce manuscrit réside dans l'utilisation de la méthode des éléments finis pour la résolution des équations. En effet, cette méthode se fait pour le moment très rare dans le domaine du flot optique. Grâce à ce choix de résolution, nous proposons d'implémenter un contrôle local de la régularisation ainsi qu'une adaptation de maillage permettant d'affiner la solution au niveau des arêtes de l'image. Afin de réduire les temps de calcul, nous parallélisons les programmes. La première méthode implémentée est la méthode parallèle en temps appelée pararéel. En couplant un solveur grossier et un solveur fin, cet algorithme permet d'accélérer les calculs. Pour pouvoir obtenir un gain de temps encore plus important et également traiter les séquences en haute définition, nous utilisons ensuite une méthode de décomposition de domaine. Combinée au solveur massivement parallèle MUMPS, cette méthode permet un gain de temps de calcul significatif. Enfin, nous proposons de coupler la méthode de décomposition de domaine et le pararéel afin de profiter des avantages de chacune. Dans une seconde partie, nous appliquons tous ces modèles dans le cas de l'estimation du flot optique en grands déplacements. Nous proposons de nous servir du pararéel afin de traiter la non-linéarité de ce problème. Nous terminons par un exemple concret d'application du flot optique en restauration de films.

Mots clés : Flot optique, méthode variationnelle, méthode des éléments finis, variation de luminosité, contrôle adaptatif, calcul parallèle, pararéel, décomposition de domaine.

Abstract

The work presented in this thesis focuses on the estimation of the optical flow through variational methods in small and large displacements. We propose a model based on the combined local-global strategy to which we add the consideration of brightness intensity variations. The particularity of this manuscript is the use of the finite element method to solve the equations. Indeed, for now, this method is really rare in the field of the optical flow. Thanks to this choice of resolution, we implement an adaptive control of the regularization and a mesh adaptation to refine the solution on the edges of the image. To reduce computation times, we parallelize the programs. The first method implemented is a parallel in time method called parareal. By combining a coarse and a fine solver, this algorithm speeds up the computations. To save even more time and to also be able to handle high resolution sequences, we then use a domain decomposition method. Combined with the massively parallel solver MUMPS, this method allows a significant reduction of computation times. Finally, we propose to couple the domain decomposition method and the parareal to have the benefits of both methods. In the second part, we apply all these models to the case of the optical flow estimation in large displacements. We use the parareal method to cope with the non-linearity of the problem. We end by a concrete example of application of the optical flow in film restoration.

Keywords : Optical flow, variational method, finite element method, varying illumination, adaptive control, parallel computation, parareal, domain decomposition.

Conférences et Publications

Publications

[1] D. Gilliocq-Hirtz and Z. Belhachmi. Coupling parareal and adaptive control in optical flow estimation with application in movie's restoration. In *International Conference on Computer Vision and Image Analysis Applications (ICCVIA), 2015*, pages 1–6. IEEE, Jan 2015.

[2] D. Gilliocq-Hirtz and Z. Belhachmi. A massively parallel multi-level approach to a domain decomposition method for the optical flow estimation with varying illumination. *SMAI Journal of Computational Mathematics, soumis*, 2016.

[3] D. Gilliocq-Hirtz and Z. Belhachmi. Parallel methods for the optical flow in large displacements. *A soumettre*, 2016.

Communication dans une conférence internationale

Adaptive control and parareal algorithm in optical flow estimation. *ICCVIA International Conference on Computer Vision and Image Processing*, Sousse, Tunisia, 18 Janvier 2015. Avec proceedings.

Communications dans des conférences nationales

Contrôle adaptatif et algorithme pararéel pour l'estimation du flot optique. *Workshop Gradient Systems and Applications*, Strasbourg, France, 26 Septembre 2014.

Techniques variationnelles et méthodes parallèles pour l'estimation du flot optique. *Journée Doctorale des Sciences*, Mulhouse, France, 2 Juillet 2015. Communications orale et murale. Prix de la meilleure présentation de la journée doctorale des sciences.

Communication murale. *Masterclass Partial Differential Equations and Applications*, Strasbourg, France, 23 mai 2016.

Remerciements

En premier lieu, je remercie mon directeur de thèse, Zakaria Belhachmi, de m'avoir permis de mener à bien ce travail, pour tous ses conseils et sa patience. Merci à Frédéric Hecht et Faker Ben Belgacem d'avoir accepté d'être rapporteurs de ce manuscrit, ainsi qu'à Pierre Charbonnier, Cornel Murea et Bernard Brighi d'avoir bien voulu faire partie du jury.

Je tiens à remercier plus particulièrement Pierre Charbonnier et Philippe Foucher du Centre d'Etude et d'Expertise sur les Risques, l'Environnement, la Mobilité et l'Aménagement (Cerema) de Strasbourg pour les séquences d'images de trafic routier m'ayant permis de réaliser des tests avec des images réelles en haute définition.

Un grand merci à Christophe Prud'homme et à toute l'équipe de CEMOSIS de l'université de Strasbourg de m'avoir donné la possibilité d'utiliser la machine parallèle Atlas, et plus particulièrement à Alexandre Ancel pour le temps passé à l'installation de toutes les librairies nécessaires et l'assistance téléphonique toujours d'une très grande gentillesse.

Merci au professeur Sung Ha Kang de la Georgia Institute of Technology d'Atlanta pour les cours particuliers dont j'ai pu bénéficier lors de sa venue à Mulhouse.

Merci également aux secrétaires du LMIA, Liliane Fricker puis Christine-Lucienne Gerber, de m'avoir accompagnée dans chacune des démarches administratives ainsi que dans l'organisation des différents déplacements. Votre gentillesse et votre bienveillance maternelle ont été d'un grand soutient. Merci à Mathieu Sieffert pour l'installation et la permanence informatique.

Merci à l'antenne de l'école doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur (MSII, ED 269) de l'université de Haute Alsace, en particulier à Aurélia Robert, Sandra Fernandez et à la directrice madame Marie-Hélène Tuilier pour l'encadrement de qualité et leur disponibilité.

Merci aux professeurs du LMIA avec qui j'ai été en contact au cours de la thèse, pour l'ambiance chaleureuse et pour tout ce que vous m'avez apporté par vos enseignements, vos conseils ou votre soutient : Nicolas Chevallier, Abdenacer Makhlouf, Cornel Murea,

Sylvia Anicic, Michel Goze, Guido Ahumada, Elisabeth Remm, Robert Lutz, Martin Bordemann, Bernard Brighi, Augustin Fruchard, Amine Hadjar et Daniel Panazzolo.

J'en profite également pour adresser ma sincère gratitude à tous les professeurs qui m'ont éduquée tout au long de ces années d'études, de la primaire à aujourd'hui. Chacun m'a donné la chance d'apprendre un peu plus et tous ces moments m'ont été très précieux. Si je suis arrivée aussi loin, c'est en grande partie grâce à vous. J'ajoute ici une pensée particulière à mon professeur de maths de terminale, monsieur Dreyer, qui m'a montré une autre dimension des mathématiques et qui a renforcé mon goût pour cette matière.

Une pensée à toutes les personnes avec qui j'ai partagé le bureau, les permanents et les invités : Said, Abdennour, Mourad, Philippe, Othman, Benedikt, Anis, Diana, Hanene, Hadjer, Leila, Ahmed Zahari, Anja, Karima, Mohamed et tous ceux que je n'ai pas cité. Merci pour toutes les discussions enrichissantes qui ont rendues le parcours plus agréable. Bon courage à tous pour la suite. Merci également à Olivier pour les coups de pouce lors de mes débuts dans l'enseignement ainsi que pour toutes les conversations mathématiques plus éloignées du travail (dont les longs mails que j'ai à chaque fois pris grand plaisir à lire). Je remercie aussi toutes les personnes que j'ai pu rencontrer lors de conférences, écoles d'été, workshop, qui m'ont offert de bons souvenirs.

Merci à Anaïs, mon amie de longue (très longue) date d'avoir toujours été présente, pour tous les bons moments et à Sacha pour les sorties des week-ends avant ton envol vers le Canada.

Merci à ma famille de m'avoir soutenue, encouragée, d'avoir veillé sur moi et de m'avoir accompagnée depuis le début. Merci à ma mère d'avoir toujours cru en moi avec une certitude souvent déconcertante de mon point de vue. Merci à mes grands-parents d'être toujours là et de m'entourer de toute leur tendresse. Les repas familiaux ont été, sont et resteront des moments parmi les plus précieux de tous. Et bien sûr, merci à mon père d'avoir toujours gardé un oeil sur moi, même de loin. Merci également à Jean-François, Marie-Christine, Giancarlo, Valentin, Maxime, Madeleine et Jocelyne. Une pensée particulière à ma petite soeur, Amy, même si l'on ne se voit pas souvent, je ne t'oublie pas ! Comme je ne peux malheureusement pas citer tout le monde, je remercie les familles Hirtz, Gilliocq et Massaro.

Enfin, comme chacun le sait, les personnes les plus importantes viennent souvent en dernier (peut-être parce que ce sont celles que l'on ne risque jamais d'oublier) : merci à Michel d'avoir été là pour moi du début à la fin, de m'avoir aidée, soutenue, écoutée, réconfortée, (supportée !) tout du long. Ces onze années à tes côtés ont été les plus belles.

Merci à tous ceux qui, malgré la longue liste, ont été oubliés.

A ma famille.

Table des matières

Table des figures

Introduction

L'estimation du mouvement est un thème de recherche très étudié depuis son introduction dans le milieu du XX^{e} siècle par le psychologue spécialisé en perception visuelle James J. Gibson [51]. De nombreuses applications existent, comme par exemple en robotique pour la détection d'obstacles, en informatique pour la compression d'images, en médecine pour l'analyse de l'évolution ou de l'apparition de certaines pathologies, dans le domaine des jeux vidéo pour l'amélioration du taux de rafraichissement des images, en ingénierie civile pour l'analyse de mouvements de foules et du trafic routier ou encore dans le domaine de la vidéo pour la restauration de films endommagés. Il existe actuellement de nombreuses méthodes pour estimer le flot optique parmis lesquelles celles utilisant les équations aux dérivées partielles et en particulier les méthodes variationnelles.

Dans le premier chapitre, nous faisons un état de l'art des méthodes variationnelles pour la résolution du flot optique. Ces méthodes serviront de base aux modèles présentés dans ce manuscrit. Nous introduirons également les problématiques liées aux variations de l'intensité lumineuse [49], [29] ainsi que l'estimation du flot optique en grands déplacements. Enfin, nous étudierons certaines méthodes permettant l'amélioration de la solution telles que la méthode du contrôle local de la régularisation [15], la méthode nommée *Coarse to fine* ou encore une stratégie de rapprochement itératif.

Le chapitre 2 introduira les différentes méthodes parallèles choisies afin de réduire les temps de calcul. En particulier, nous présenterons une méthode de décomposition de domaine ainsi qu'un algorithme de parallélisation en temps introduit par Lions, Maday et Turinici [77].

Le coeur de notre travail débutera dans le chapitre 3 où nous étudierons l'estimation du flot optique en petits déplacements en utilisant la méthode des éléments finis. Nous commencerons par étudier le problème en supposant la luminosité constante entre les images de la séquence. Dans un second temps, nous utiliserons un modèle se basant sur les travaux de Gennert et Negahdaripour [49] prenant en compte la variation de la luminosité. Dans le but d'obtenir une amélioration de la solution sur les arêtes de l'image, nous appliquerons une méthode de contrôle local de la régularisation présentée par Belhachmi et Hecht [15] utilisant un indicateur d'erreur a posteriori. Le chapitre sera

conclu par une étude des résultats obtenus à l'aide de nos modèles.

L'utilisation de la méthode des éléments finis étant plus coûteuse que la méthode des différences finies utilisée par la majorité des auteurs, nous étudierons dans le quatrième chapitre des implémentations parallèles de la résolution du problème du flot optique. Nous commencerons par reprendre la méthode de parallélisation en temps introduite dans le chapitre 2 afin de l'adapter au problème du flot optique. Dans un deuxième temps, nous proposerons une parallélisation intuitive pour une séquence complète d'images consistant à déterminer simultanément de nombreuses estimations du flot optique. Ensuite, nous implémenterons une méthode de décomposition de domaine utilisant une méthode de Schwarz. Pour finir ce chapitre, nous couplerons cette méthode de décomposition de domaine avec le solveur massivement parallèle MUMPS afin d'obtenir une parallélisation multi-niveaux et ainsi réduire les temps de calcul. Une combinaison avec la méthode pararéelle sera également développée visant à bénéficier des avantages des deux modèles.

Les chapitres 5 et 6 reprendront les plans des chapitres 3 et 4 et viseront à étudier le problème du flot optique dans le cas des grands déplacements. Le problème étant non-linéaire dans ce cas, nous utiliserons une méthode itérative de point fixe ainsi qu'une décomposition des variables afin de le linéariser. La prise en compte de la variation de la luminosité ainsi que la stratégie du contrôle local de la régularisation seront également implémentés.

Concernant les méthodes parallèles utilisées, nous implémenterons la méthode de décomposition de domaine, le pararéel ainsi que le couplage de ces deux méthodes afin de réduire les temps de calcul. Dans le cas de la méthode pararéelle, nous nous servirons du solveur fin afin de considérer différentes linéarisations du problème initial. Nous effectuerons les tests sur des séquences en grands déplacements ainsi que sur des photographies réelles.

Dans le dernier chapitre, nous étudierons une application de l'étude du flot optique pour la restauration de films. Après avoir déterminé le masque de dégradation de l'image concernée, nous utiliserons les équations de Ginzburg-Landau permettant la reconstruction de l'image. Nous proposerons pour ce modèle, une méthode de parallélisation de séquence similaire à celle introduite dans le chapitre 4.

Chapitre 1

Etat de l'art

Sommaire

1.1 Définition

Afin de définir le flot optique, nous reprenons l'analyse d'image proposée par Aubert et Deriche [8]. Considérons l'exemple concret d'une séquence prise dans une rue (voir figure 1.1).

Une première approche permettant de comprendre la différence entre ce que nous observons et le mouvement réel en trois dimensions est de voir la caméra comme un modèle

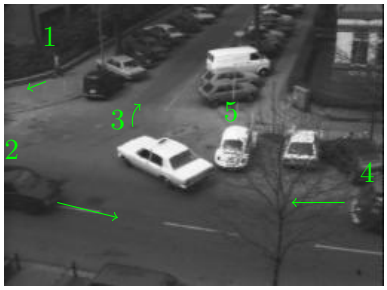


FIGURE 1.1 – Exemple de séquence. Quatre objets sont en mouvement. 1 : un piéton, 2 et 3 : deux voitures, 4 : un van. 5 est une voiture immobile mais implique une variation de luminosité dûe au bruit.

projectif et ainsi considérer le mouvement 2D du champ de vitesses sur l'image comme la projection du champ de vitesses 3D des objets. C'est notamment le cas pour la voiture 3. Cependant, nous observons également derrière le véhicule que l'ombre de celui-ci se déplace avec lui. Le déplacement perçu ne correspond dans ce cas à aucun mouvement réel. C'est également le cas en présence d'objets réfléchissants lorsque la luminosité varie rapidement avec la position, par exemple pour la vitre du van 4. Enfin, notons le problème du bruit sur l'image. Par exemple, bien que la voiture 5 soit stationnée donc immobile, des changements aléatoires de luminosité dus aux bruits présents sur la séquence seront observés. Ainsi, suivant la définition communément acceptée donnée dans [8], nous définissons le flot optique comme le champ de vitesses 2D décrivant les changements d'intensité des pixels entre deux images successives d'une séquence. Nous allons voir dans la suite comment traduire ce problème mathématicalement.

1.2 Méthodes variationnelles

Depuis son introduction par Gibson en 1966 [51], des modèles analytiques pour l'estimation du mouvement ont été proposé [57, 74, 79]. Il a fallu cependant attendre le début des années 80 pour voir apparaître les premiers algorithmes [6, 12, 21, 33, 63, 69, 80, 90, 92, 98]. Il existe actuellement une bibliographie très riche dans ce domaine où de nombreuses méthodes sont employées comme par exemple des méthodes par filtrage [18, 19, 47, 109], des méthodes statistiques [108, 121, 122] ou encore des méthodes variationnelles traitant le problème sous la forme d'un système d'équations différentielles [8, 28, 31, 49, 69, 80, 99, 103, 119, 120]. Nous renvoyons le lecteur intéressé vers [12] qui propose une comparaison entre différentes techniques d'estimation du flot optique à l'aide de tests numériques, ainsi que vers [97] qui est une mise à jour de ce dernier. Notre contribution s'inscrit dans le sillage des méthodes variationnelles pour l'estimation du flot optique.

1.2.1 Luminosité constante

Le point commun des méthodes variationnelles est qu'elles reposent pour la plupart sur l'hypothèse de la constance de l'intensité des pixels au cours du mouvement. Plus précisément, l'intensité d'un point reste constante le long de sa trajectoire. Cette hypothèse est appelée la *contrainte fondamentale du flot optique*. Elle apparaît raisonnable dans le cas des petits déplacements, les changements de luminosité d'une image à l'autre étant faibles.

Nous considérons une séquence d'images successives et définissons $\Omega \subset \mathbb{R}^2$ le domaine de l'image. L'intensité d'un pixel (x, y) au moment t est définie par la fonction

$$\begin{array}{rccc} \mathrm{f}: & \Omega \times [0,T] & \rightarrow & \mathbb{R} \\ & (x,y),t & \mapsto & f(x,y,t). \end{array}$$

Nous définissons la trajectoire

$$t \mapsto (x(t), y(t), t)$$

telle que pour le point (x_0, y_0) à l'intant t_0 nous ayons

$$(x(t_0), y(t_0), t_0) = (x_0, y_0, t_0)$$

et

$$f(x(t), y(t), t) = f(x_0, y_0, t_0) \qquad \forall t. \tag{1.1}$$

En dérivant (1.1) par rapport à t, nous obtenons à $t = t_0$

$$\frac{\partial f}{\partial t}(x_0, y_0, t_0) + \frac{\partial f}{\partial x}(x_0, y_0, t_0)\frac{dx}{dt}(t_0) + \frac{\partial f}{\partial y}(x_0, y_0, t_0)\frac{dy}{dt}(t_0) = 0. \tag{1.2}$$

Les dérivées en temps de x et de y représentent respectivement les déplacements suivant x et y. Nous définissons donc le flot optique tel que

$$\mathbf{u}(x_0, y_0) = (u_1(x_0, y_0), u_2(x_0, y_0))$$

où

$$u_1(x_0, y_0) = \frac{dx}{dt}(t_0) \text{ et } u_2(x_0, y_0) = \frac{dy}{dt}(t_0).$$

En utilisant la notation $f_* = \frac{\partial f}{\partial *}$, nous réécrivons (1.2) et obtenons *la contrainte fondamentale du flot optique*

$$f_x u_1 + f_y u_2 + f_t = 0. \tag{1.3}$$

Remarquons que cette seule équation ne suffit pas à déterminer les deux composantes du flot optique u_1 et u_2. Le problème est mal posé. Il est communément appellé *problème de l'ouverture*. D'autre hypothèses doivent être considérées.

Remarque 1.2.1. *Même si la contrainte fondamentale est largement utilisée afin de calculer le flot optique, plusieurs raisons peuvent nous forcer à considérer des hypothèses différentes. Parmi elles, nous pouvons citer le cas où l'on souhaite prendre en compte les possibles changements de luminosité d'une image à une autre. Comme nous le verrons dans la section correspondante, en suivant le travail de Gennert et Negahdaripour [49], le modèle proposé permettra une variation affine de l'intensité au cours du temps. Une autre raison de ne pas considérer la contrainte fondamentale est le cas des grands déplacements. Comme nous le verrons, nous éviterons cette hypothèse puisque lors d'un large déplacement, les variations d'intensité sont souvent plus nombreuses et plus prononcées.*

Afin de résoudre le problème de l'ouverture, plusieurs méthodes ont été proposées. Nous en détaillons deux d'entre elles : la méthode locale et la méthode globale.

1.2.1.1 Méthode Locale

Afin de résoudre le problème de l'ouverture, en 1981, Lucas et Kanade [80] proposent une hypothèse locale. Ils supposent que les points appartenant à un même voisinage ont un déplacement proche. Soit $\mathcal{V}(x, y)$ le voisinage d'un pixel (x, y), le problème du flot optique est alors défini pour chaque pixel comme la minimisation au sens des moindres carrés de la fonctionnelle

$$\int_{\mathcal{V}(x,y)} (f_{x'} u_1 + f_{y'} u_2 + f_t)^2 dx' dy'. \tag{1.4}$$

où $(x', y') \in \mathcal{V}(x, y)$.

L'intégration sur un voisinage peut être considérée comme une convolution par la fonction indicatrice $\mathbb{1}_{\mathcal{V}(x,y)}$ valant 1 dans le voisinage de (x, y) et 0 ailleurs. La minimisation de la fonctionnelle (1.4) peut alors être remplacée par la minimisation de

$$\mathbb{1}_{\mathcal{V}(x,y)} \star (f_x u_1 + f_y u_2 + f_t)^2.$$

Nous pouvons cependant supposer que le caractère local du mouvement autour du pixel (x, y) diminue à mesure que nous nous en éloignons. Ainsi, dans le but de donner un rôle plus important aux pixels les plus proches de (x, y), nous remplaçons la fonction indicatrice par une fonction gaussienne de déviation standard ρ notée K_ρ. Finalement, pour chaque pixel (x, y), la méthode de Lucas et Kanade vise à minimiser l'expression

$$Loc(\mathbf{u}) = K_\rho \star (f_x u_1 + f_y u_2 + f_t)^2. \tag{1.5}$$

Après avoir écrit les équations d'Euler-Lagrange, le problème à résoudre est donné par

$$\begin{bmatrix} K_\rho \star (f_x)^2 & K_\rho \star (f_x f_y) \\ K_\rho \star (f_y f_x) & K_\rho \star (f_y)^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -K_\rho \star (f_x f_t) \\ -K_\rho \star (f_y f_t) \end{bmatrix} \tag{1.6}$$

La simplicité de cette méthode est un avantage en termes de performances de calcul puisque pour chaque pixel, il n'y a qu'un système linéaire 2×2 à résoudre. En revanche, le caractère local de l'hypothèse peut impliquer un manque de précision de la solution au niveau des petits objets ainsi que sur les discontinuités (bords des objets).

1.2.1.2 Méthode Globale

En 1981 également, Horn et Schunck [69] proposent une approche globale pour résoudre le problème de l'ouverture. Contrairement à l'hypothèse locale, ils supposent non pas que le champ de vecteurs soit constant par morceaux mais qu'il soit globalement régulier et considèrent la minimisation de la fonctionnelle

$$Glob(\mathbf{u}) = \int_\Omega \underbrace{(f_x u_1 + f_y u_2 + f_t)^2}_{\mathcal{D}} + \underbrace{\alpha(|\nabla u_1|^2 + |\nabla u_2|^2)}_{\mathcal{R}} dxdy \tag{1.7}$$

où α est un paramètre de régularisation constant jouant le rôle de pénaliseur pour les gradients de la solution. Ce paramètre a donc pour effet de lisser le champ de vecteurs. En d'autres termes, le but de ce problème est donc de déterminer le champ de vecteurs

$\mathbf{u} = (u_1, u_2)$ vérifiant au mieux la contrainte fondamentale du flot optique (terme $\mathcal{D}$) tel que la valeur des dérivées soit petite (terme $\mathcal{R}$). Nous proposons un exemple de résultat donné par cette méthode figure 1.5.

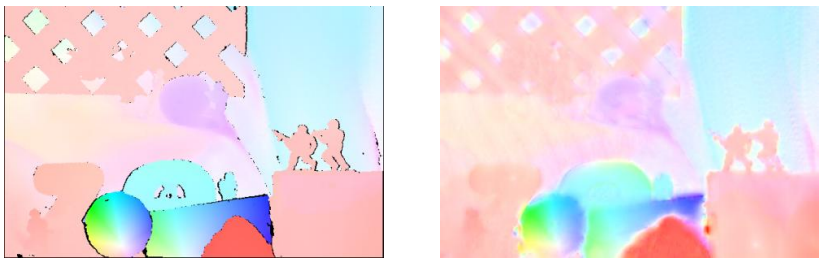


FIGURE 1.2 – Gauche : Solution exacte pour le cas test *Army* du site *Middleburry* www.vision.middleburry.edu/flow/. Droite : Estimation à l'aide de la méthode globale de Horn et Schunck [1].

Ce type de régularisation a été introduit par Tikhonov et Arsenin [113] et est connu pour lisser le champ de vecteurs de façon uniforme sans prendre en compte les discontinuités. Dès lors, de nombreux travaux ont été développés afin de calculer des champs de vecteurs discontinus. La plupart reposent sur une modification du terme de régularisation $\mathcal{R}$. Nous décrivons ici certaines de ces méthodes, également décrites dans [9].

Le premier à avoir modifié la fonctionnelle de Horn et Schunk en introduisant des normes robustes fut Black [20]. Plusieurs auteurs ont ensuite poursuivi les recherches dans ce sens. L'idée est de remplacer le terme de régularisation par

$$\int_\Omega \psi(|\nabla u_1|) + \psi(|\nabla u_2|) dxdy.$$

où les ψ sont des fonctions permettant d'ôter le bruit et de conserver les bords des objets de l'image. Par exemple, Cohen [36] ainsi que Kumar, Tannenbaum et Balas [76] ont utilisé la norme L^1, c'est à dire, la variation totale ($\psi(t) = t$). Deriche, Kornprobst et Aubert [40] ont utilisé des fonctions plus générales pour préserver les discontinuités. Cette idée est également appelée méthode des "estimateurs robustes" dans la littérature stochastique.

Une deuxième approche, utilisée par Gupta et Prince [62] ainsi que par Guichard et Rudin [61], consiste à ajouter des termes de pénalisations basés sur la divergence ou le rotationnel du flot optique en considérant

$$\int_\Omega \varphi(\mathrm{div}(\mathbf{u}), \mathrm{rot}(\mathbf{u})) dxdy$$

1. Source : http ://vision.middlebury.edu/flow/eval/results/

où plusieurs propositions pour φ peuvent être faites. Par exemple, dans [61], le terme de régularisation est seulement

$$\int_\Omega |\operatorname{div}(\mathbf{u})|\, dxdy.$$

Dans ce cas, l'idée est de noter que les objets 2D rigides en mouvements 2D ont un déplacement de divergence libre. La divergence est non-nulle uniquement sur les frontières des occlusions où cela s'apparente à une mesure concentrée.

Une troisième proposition a été faite par Nagel et Enkelmann [92] ainsi que par Nagel [90]. Ils considèrent une contrainte de régularisation orientée dans laquelle le lissage n'est pas imposé au niveau des zones à fortes variations des gradients (arêtes) afin de mieux traiter les occlusions. Le terme de pénalisation est alors de la forme

$$\int_\Omega \frac{1}{|\nabla f|^2 + 2\delta}[(\partial_x u_1 \partial_y f - \partial_y u_1 \partial_x f)^2 + (\partial_x u_2 \partial_y f - \partial_y u_2 \partial_x f)^2 + \delta(\partial_x u_1^2 + \partial_y u_1^2 + \partial_x u_2^2 + \partial_y u_2^2)]dxdy$$

où δ est une constante. Minimiser cette fonctionnelle par rapport à $\mathbf{u}$ permet d'atténuer les variations du flot dans la direction perpendiculaire au gradient.

La dernière approche que nous présenterons est la méthode proposée par Nési [95]. Il adapte la formulation de Horn et Schunck en introduisant la longueur de l'ensemble des discontinuités de $\mathbf{u}$ noté $|S_{\mathbf{u}}|$. Cette idée a été introduite par Mumford et Shah dans le domaine de la segmentation d'image [89]. Le terme de régularisation est de la forme :

$$\int_\Omega \alpha(|\nabla u_1|^2 + |\nabla u_2|^2)dxdy + \alpha^d\, |S_{\mathbf{u}}|\,.$$

Numéricalement, la principale difficulté est l'approximation du dernier terme. Une solution possible est d'utiliser le concept de Γ-convergence (voir [37, 39] pour plus de détails). On introduit une séquence de fonctionnelles telle que la séquence des minimiseurs converge vers un minimum de la fonctionnelle initiale. Typiquement, la façon d'estimer le terme de régularisation est (voir [5] pour plus de détails)

$$\int_\Omega \alpha z^2(|\nabla u_1|^2 + |\nabla u_2|^2)dxdy + \int_\Omega \alpha^d \left(\frac{|\nabla z|^2}{k} + \frac{k(1-z)^2}{4}\right) dxdy \qquad (1.8)$$

où z est une fonction additionnelle et k est un paramètre qui tendra vers l'infini. La fonction z peut être considérée comme une variable de contrôle qui est égale à zéro près des discontinuités et qui est proche de 1 dans les régions homogènes.

La liste des méthodes citées n'est pas exhaustive. Nous avons parlé de méthodes axées sur la modification du terme de régularisation. Afin d'améliorer la précision de l'estimation, il existe également des modifications concernant le terme d'adéquation aux données $\mathcal{D}$. Nous référons le lecteur vers l'étude complète proposée par Bruhn [29]. D'autres modèles sont également proposés dans [100], [125], [102], [10], [24], [22], [6], [21].

1.2.1.3 Combinaison Locale–Globale

En 2005, Bruhn *et al.* [31] proposent de combiner les méthodes locale et globale. Le terme de régularisation de la méthode globale est conservé mais le terme relatif aux données est remplacé par celui de la méthode locale. La fonctionnelle à minimiser est alors

$$CLG(\mathbf{u}) = \int_{\Omega} K_\rho \star (f_x u_1 + f_y u_2 + f_t)^2 + \alpha(|\nabla u_1|^2 + |\nabla u_2|^2) dxdy. \tag{1.9}$$

Comme dans le cas de la méthode globale, il est possible de choisir différentes configurations de cette équation en modifiant les pénalisations de la partie concernant les données du problème et la partie concernant la régularisation. Les travaux de Bruhn [29] proposent une explications complète de ces différentes configurations.

1.2.2 Méthodes adaptatives

De manière générale, le choix du paramètre de régularisation α de la méthode globale de Horn et Schunck est déterminé de manière empirique pour chaque problème afin d'obtenir le résultat le plus précis possible. Le mieux étant d'éviter de lisser trop fortement le champ de vecteurs afin de ne pas perdre d'informations concernant les arêtes sans pour autant obtenir un champ de vecteurs affecté par les textures. Afin de gérer ce choix de manière automatique et locale, des méthodes permettant un contrôle de ce paramètre ont été introduites. Parmi elles, une méthode de contrôle adaptatif local a posteriori du paramètre de régularisation α qui Γ-converge vers Mumford-Shah a été proposée par Belhachmi et Hecht [15, 16]. Dans cette méthode, le paramètre α n'est plus une constante mais une fonction constante par morceaux dépendante des directions x et y. La fonction est alors définie de la manière suivante. Nous commençons par considérer une subdivision du domaine Ω

$$\Omega = \bigcup_i \Omega_i.$$

Pour tout ℓ, on définit la fonction

$$\alpha(x, y) = \alpha_\ell \text{ sur } \Omega_\ell,\ \alpha_\ell > 0.$$

La fonctionnelle à minimiser (1.9) devient

$$\int_\Omega \underbrace{K_\rho \star (f_x u_1 + f_y u_2 + f_t)^2}_{\text{données}} + \underbrace{\alpha(x, y)(|\nabla u_1|^2 + |\nabla u_2|^2)}_{\text{régularisation}} dxdy. \tag{1.10}$$

La valeur du paramètre est ensuite modifiée en fonction de la géométrie à l'aide d'un indicateur d'erreur a posteriori (voir la section 3.1.5 pour plus de détails). Près des arêtes de l'image, c'est à dire au niveau des bords des différents objets, cette valeur est diminuée afin de minimiser la diffusion et donc d'obtenir des angles mieux définis. Sur les zones homogènes, la fonction $\alpha(x, y)$ conserve sa valeur initiale. Cette méthode vise également à appliquer une adaptation du maillage permettant d'affiner les zones contenant des discontinuités et d'obtenir un maillage plus grossier sur les zones homogènes.

Notre travail s'appuie sur ce contrôle local du paramètre de régularisation que nous expliquerons en détails dans le chapitre 2.3. Contrairement aux méthodes variationnelles précédemment citées qui utilisent la méthode des différences finies, nous utiliserons la méthode des éléments finis. Bien que plus coûteuse, cette méthode a l'avantage de nous permettre d'effectuer les calculs sur une géométrie raffinée grâce à l'adaptation de maillage et d'utiliser la méthode adaptative du paramètre de régularisation.

1.3 Généralisation du flot optique

1.3.1 Flot optique avec luminosité variable

L'hypothèse de la constance de l'intensité (1.1) suppose que l'intensité lumineuse est la même entre deux images successives. Cependant, comme nous l'avons vu sur la figure 1.1, dans le cas d'images réelles nous pouvons nous trouver face à des occlusions, des ombres ou des reflets. L'utilisation d'une telle hypothèse peut donc mener à des estimations peu précises, voire totalement erronées. Pour résoudre ce problème, deux modèles n'utilisant pas (1.1) se sont imposés dans la littérature. Le premier considère la variation de lumière

entre deux images comme une transformation linéaire, le deuxième consiste à ajouter une hypothèse impliquant la constance du gradient de la luminosité.

1.3.1.1 Transformation linéaire de la luminosité

Afin de prendre en compte les variations de luminosité d'une image à l'autre, Gennert et Negahdaripour [49] ont proposé de considérer la variation d'intensité lumineuse comme une transformation linéaire. Cette nouvelle hypothèse, moins restrictive que celle sur la constance de la luminosité, peut s'écrire

$$f(x+\delta x, y+\delta y, t+\delta t) = M(x,y,t)f(x,y,t) + C(x,y,t) \tag{1.11}$$

où M est le coefficient multiplicateur et C le coefficient de translation.

Pour des petites variations de δt, nous pouvons supposer que M est proche de 1 et que C est proche de 0. On note alors $M = 1 + \delta m$ et $C = \delta c$. Avec ces notations, l'équation (1.11) devient

$$f(x+\delta x, y+\delta y, t+\delta t) = (1+\delta m(x,y,t))f(x,y,t) + \delta c(x,y,t). \tag{1.12}$$

En appliquant un développement de Taylor sur le terme de gauche, nous obtenons

$$f(x+\delta x, y+\delta y, t+\delta t) = f(x,y,t) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y + \frac{\partial f}{\partial t}\delta t + o(t).$$

En insérant cette expression dans l'équation (1.12), nous obtenons

$$f_x \delta x + f_y \delta y + f_t \delta t - f\delta m - \delta c + o(t) = 0. \tag{1.13}$$

En divisant (1.13) par δt et en faisant tendre δt vers 0, nous obtenons finalement la nouvelle contrainte du flot optique permettant de considérer des séquences à luminosité variable

$$f_x u_1 + f_y u_2 + f_t - f m_t - c_t = 0 \tag{1.14}$$

où

$$u_1 = \lim_{\delta t \to 0} \frac{\delta x}{\delta t},\ u_2 = \lim_{\delta t \to 0} \frac{\delta y}{\delta t},\ m_t = \lim_{\delta t \to 0} \frac{\delta m}{\delta t} \text{ et } c_t = \lim_{\delta t \to 0} \frac{\delta c}{\delta t}.$$

Remarque 1.3.1. *Dans le cas où la luminosité reste constante, on a $M = 1$ et $C = 0$, donc $m_t = 0$ et $c_t = 0$. Ainsi, nous avons l'équivalence (1.3) $\Leftrightarrow$ (1.14)*

1.3.1.2 Constance du gradient de la luminosité

Une autre méthode utilisée par Bruhn [29] et Papenberg *et al.* [99] pour le traitement de la luminosité variable consiste à considérer la constance du gradient de la luminosité en plus de (1.1). Nous avons alors

$$\begin{aligned} f_x(x, y, t) &= f_x(x + u_1, y + u_2, t + 1) \\ f_y(x, y, t) &= f_y(x + u_1, y + u_2, t + 1). \end{aligned}$$

De la même manière que dans la section précédente, pour un petit déplacement, on peut linéariser ces équations à l'aide d'un développement de Taylor pour obtenir

$$\begin{aligned} f_{xx}u_1 + f_{xy}u_2 + f_{xt} &= 0 \\ f_{yx}u_1 + f_{yy}u_2 + f_{yt} &= 0 \end{aligned}$$

où $f_{xy} = \frac{\partial^2 f}{\partial x \partial y}$. La méthode consiste ensuite à minimiser la fonctionnelle

$$\begin{aligned} CLGB(\mathbf{u}) = \int_\Omega \Psi(&K_\rho \star (f_x u_1 + f_y u_2 + f_t)^2 \\ &+ K_\rho \star (f_{xx}u_1 + f_{xy}u_2 + f_{xt})^2 \\ &+ K_\rho \star (f_{yx}u_1 + f_{yy}u_2 + f_{yt})^2) \\ &+ \alpha\Psi(|\nabla u_1|^2 + |\nabla u_2|^2) dxdy. \end{aligned}$$

La fonction Ψ est une fonction de régularisation, comme auparavant, permettant de s'assurer d'avoir un problème bien posé et la convergence globale du problème. Nous avons vu précédemment qu'elle peut être choisie de diverses manières. Cette fonction a été introduite dans [120] et permet également de mieux traiter les discontinuités de l'image créées par les bords des objets ainsi que les zones d'occlusions. Dans le cas des

trois derniers articles cités, la fonction Ψ est définie pour une valeur ε de l'ordre de 10^{-6} par

$$\Psi(s^2) = \sqrt{s^2 + \varepsilon^2}.$$

Dans la suite, nous utilisons une variation de la méthode de Gennert et Negahdaripour [49] dans laquelle nous supposons que le terme de translation est négligeable. Nous choisissons cette méthode car elle nous permettra d'obtenir un nouveau paramètre de régularisation associé à la nouvelle variable m_t, que nous pourrons également adapter de manière locale.

1.3.2 Flot optique en grands déplacements

Nous introduisons dans cette section le calcul du flot optique dans le cas de grands déplacements dont une étude plus approfondie sera proposée dans le chapitre 5. C'est un problème plus complexe qui ne peut pas être traité de la même manière que pour les petits déplacements. Un grand déplacement dans une image entraîne une large zone d'occlusions dans laquelle il n'y a aucune information. Pour les méthodes variationnelles basées sur les méthodes locale et globale, cela implique de ne plus pouvoir effectuer une différenciation de l'équation donnée par la contrainte sur la luminosité. Comme dans le cas des petits déplacements, nous commençons par expliciter le problème dans le cas de la luminosité constante avant de l'étendre au cas plus général de la luminosité variable.

1.3.2.1 Luminosité constante

Dans le cas des grands déplacements, nous ne pouvons plus écrire la contrainte fondamentale du flot optique sous la forme (1.3) puisque les déplacements de l'image sont trop importants pour pouvoir appliquer une différenciation. Elle reste ainsi sous la forme

$$f(x + u_1, y + u_2, t + 1) - f(x, y, t) = 0.$$

L'intégrale à minimiser (1.9) s'écrit donc à présent sous la forme

$$\int_\Omega \underbrace{K_\rho \star (f(x+u_1, y+u_2, t+1) - f(x,y,t))^2 dxdy}_{\text{non-linéaire}} + \int_\Omega \underbrace{\alpha(x,y)(|\nabla u_1|^2 + |\nabla u_2|^2)}_{\text{linéaire}}. \quad (1.15)$$

Afin de linéariser ce problème, une des stratégies possibles [29,111] consiste à utiliser une méthode de point fixe afin d'obtenir une première semi-linéarisation du problème puis à faire la décomposition de variable suivante

$$\mathbf{u}^{n+1} = (u_1^{n+1}, u_2^{n+1}) = (u_1^n + \delta u_1^n, u_2^n + \delta u_2^n).$$

Dans le chapitre 5, nous reviendrons en détails sur le fonctionnement de cette stratégie.

Bien que cette méthode soit une de celles les plus employées, il existe également des méthodes utilisant la segmentation [33], l'identification d'objets [28] ou encore des méthodes basées sur le *coarse to fine* [6,21,99].

1.3.2.2 Luminosité variable

Pour des séquences en grands déplacements, les variations de luminosité ainsi que les occlusions dûent aux mouvements des éléments de la scène sont plus importantes. Pour ces raisons, nous étendons le modèle du flot optique des grands déplacements au modèle prenant en compte la luminosité variable. Dans ce cas, le problème revient à la minimisation de la fonctionnelle

$$\int_\Omega \underbrace{K_\rho \star (f(x+u_1, y+u_2, t+1) - (1+m_t) f(x,y,t) - c_t)^2 dxdy}_{\text{non-linéaire}} + \int_\Omega \underbrace{\alpha(|\nabla u_1|^2 + |\nabla u_2|^2) + \lambda |\nabla m_t|^2 dxdy}_{\text{linéaire}}. \quad (1.16)$$

Nous verrons dans le chapitre 5 que nous linéariserons ce problème avec la méthode itérative de point fixe.

1.4 Optimisation des résultats : *Coarse to fine*

Nous venons de voir qu'il existe de nombreuses méthodes permettant de résoudre le problème du flot optique. Le choix de l'utilisation d'une méthode par rapport à une autre se fait en fonction de sa complexité et de son efficacité en termes de temps et de précision de la solution. En effet, certaines des méthodes présentées ci-dessus peuvent souffrir d'un manque de précision. Il existe deux stratégies permettant d'améliorer la précision des résultats indépendamment de la méthode choisie.

La première, que nous appelons *rapprochement itératif* [82], consiste à déterminer une suite de solutions qui converge vers la solution exacte du problème. Pour cela, nous déterminons le flot optique $\mathbf{u}$ entre l'image f_1 et l'image f_2 puis nous utilisons ce résultat afin de déterminer, à l'aide d'une interpolation bilinéaire, l'image $f_3 = f(\mathbf{x} + \mathbf{u})$. Nous calculons ensuite le flot optique entre l'image f_3 et l'image f_2. La méthode est schématisée sur la figure 1.3. La solution finale est donnée par la somme des flots optiques calculés au cours des itérations.

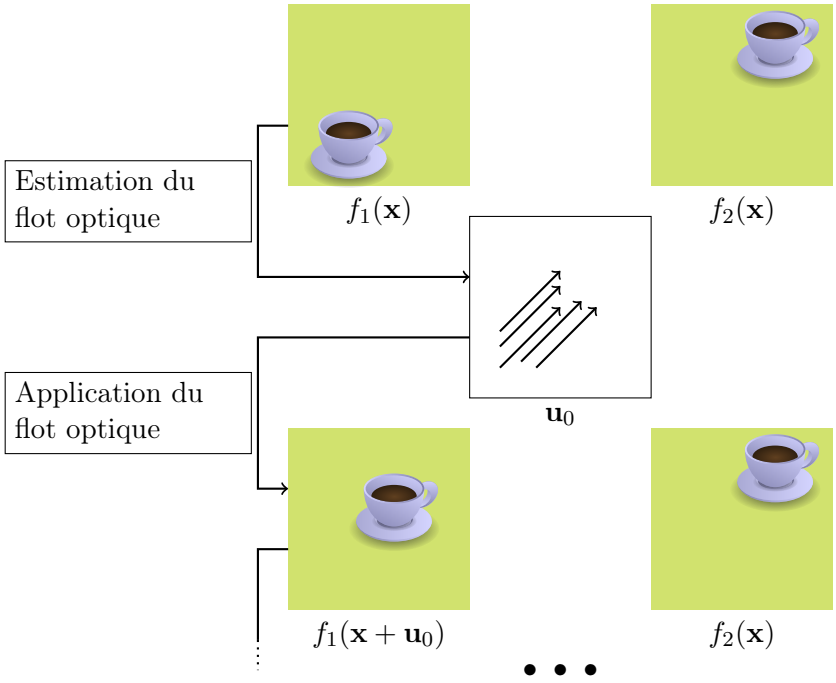


FIGURE 1.3 – Technique d'optimisation pour l'estimation du flot optique.

La méthode appelée *coarse to fine* est une combinaison de ce rapprochement itératif avec l'utilisation d'une pyramide gaussienne. Nous commmençons par déterminer le flot optique entre deux images sur lesquelles est appliqué un flou gaussien important. Puis, à chaque itération nous considérons des images de moins en moins lissées (voir figure 1.4).

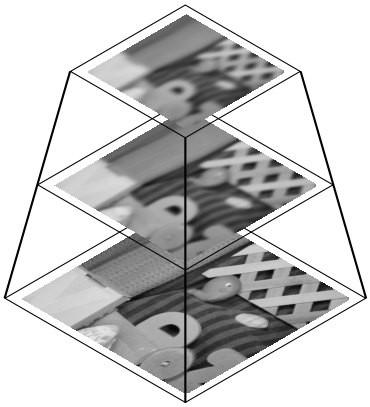


FIGURE 1.4 – Pyramide gaussienne.

Sur la figure 1.5, nous présentons une solution obtenue à l'aide de la méthode locale de Lucas et Kanade à laquelle est ajoutée l'optimisation par *coarse to fine*.

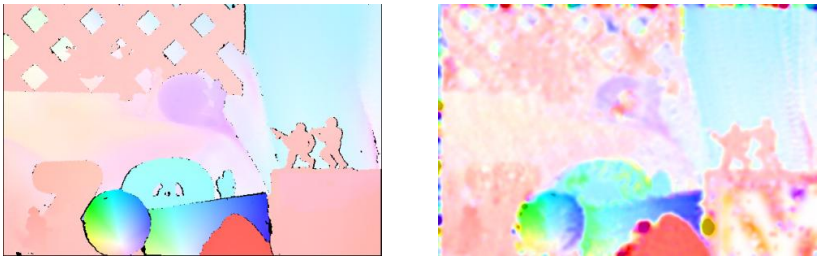


FIGURE 1.5 – Gauche : Solution exacte pour le cas test *Army* du site *Middleburry* www.vision.middleburry.edu/flow/. Droite : Estimation obtenue avec la méthode locale de Lucas et Kanade optimisée à l'aide de la pyramide gaussienne [2].

Dans la suite, nous reviendrons sur ces méthodes que nous appliquerons, en particulier pour l'estimation du flot optique en grands déplacements dans le chapitre 5.

1.5 Conclusion

Ce premier chapitre a été consacré à l'étude générale des méthodes variationnelles existantes pour l'estimation du mouvement. Nous avons pu constater que ces méthodes sont nombreuses dans la littérature et que le travail autour de ce problème est constant depuis plus de 30 ans. Nous avons présenté de manière succincte les méthodes locale, globale et la combinaison locale/globale. Le calcul du flot optique avec la prise en compte de la luminosité a été introduit, il permettra de condidérer des séquences d'images en situa-

2. Source : http ://vision.middlebury.edu/flow/eval/results/

tion réelle sur lesquelles la lumière peut varier. Nous avons également étudié le cas du problème de l'estimation du flot optique en grands déplacements ainsi qu'une méthode variationnelle employée traditionnellement. Enfin, une méthode d'optimisation de la précision des résultats combinant un rapprochement itératif ainsi que la méthode du *coarse to fine* a été présentée.

Chapitre 2

Outils et méthodes parallèles pour les méthodes variationnelles

Sommaire

La demande croissante d'algorithmes efficaces en termes de temps de calcul et l'émergence du calcul sur carte graphique ont imposé au scientifique actuel une nouvelle vision. En plus de proposer un modèle pour la résolution d'un problème, il faut réfléchir en termes de parallélisation. La parallélisation consiste à partager l'exécution d'un calcul entre plusieurs processeurs. Toutes les implémentations ne sont pas nécessairement parallélisables, c'est pourquoi il est nécessaire d'analyser les dépendances des opérations du modèle de résolution. Le calcul peut être partagé entre les coeurs de calcul des processeurs, entre les différents processeurs d'un super ordinateur ou sur les coeurs d'une carte graphique. Le développement d'une telle méthode peut être complexe mais les gains en termes de temps de calcul peuvent aller du facteur 2 au facteur 100 pour les algorithmes les plus efficaces.

L'estimation du flot optique, de par la taille des images à traiter, est un problème coûteux en calculs. En effet, avec les méthodes variationnelles que nous allons utiliser par la suite, les tailles des systèmes linéaires dépendent du nombre de pixels des images traitées.

Ainsi, l'utilisation du calcul parallèle représente un atout majeur pour la résolution de ces systèmes. Nous introduisons dans cette section la méthode de parallélisation par décomposition de domaine [11, 42, 78, 104]. Une telle méthode a déja été présentée pour le problème du flot optique par Kohlberger *et al.* [75] dans le cas d'un maillage structuré pour la combinaison Locale-Globale. La méthode par décomposition de domaine s'est largement imposée dans le domaine de la parallélisation pour son efficacité en termes de scalabilité. Dans nos travaux, nous utiliserons également la méthode *pararéelle* [13, 77] représentant une parallélisation en temps. Nous présentons ce modèle à travers un exemple dans la deuxième partie de ce chapitre.

2.1 Décomposition de domaine : Méthode de Schwarz

Les méthodes de décomposition de domaine consistent à séparer l'image à traiter en plusieurs parties puis à résoudre le problème sur chaque sous-domaine de manière indépendante. Elles peuvent être classées en deux groupes, les méthodes sans recouvrement entre les sous-parties et les méthodes avec recouvrement.

Afin d'expliquer ces méthodes, nous prenons l'exemple du problème elliptique linéaire général

$$\begin{cases} Lu = f & \text{dans } \Omega \\ u = g & \text{sur } \partial\Omega \end{cases} \tag{2.1}$$

dans lequel Ω représente le domaine sur lequel nous recherchons une solution et $\partial\Omega$ représente sa frontière.

2.1.1 Sans recouvrement

Une version de la méthode de Schwarz sans recouvrement (figure 2.1) a été proposée par P.-L. Lions dans [78].

Afin de résoudre le problème (2.1), l'algorithme itératif consiste, à l'étape n, à résoudre le problème suivant sur chaque sous-domaine Ω_i

$$\begin{cases} Lu^n_{|\Omega_i} = f & \text{dans } \Omega_i \\ \frac{\partial u^n_{|\Omega_i}}{\partial n_{ij}} + \lambda_{ij} u^n_{|\Omega_i} = -\frac{\partial u^{n-1}_{|\Omega_j}}{\partial n_{ji}} + \lambda_{ij} u^n_{|\Omega_j} & \text{sur } \Sigma_{ij} = \partial\Omega_i \cap \partial\Omega_j \end{cases}$$

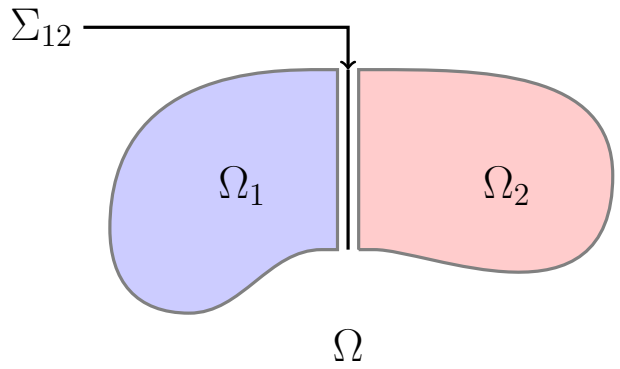


FIGURE 2.1 – Décomposition du domaine Ω sans recouvrement.

où n_{ij} désigne le vecteur normal orienté du sous-domaine Ω_i vers le sous-domaine Ω_j (nous pouvons en déduire l'égalité $n_{ij} = -n_{ji}$). Le paramètre λ_{ij} désigne un réel strictement positif. A l'instar de la méthode additive de Schwarz, cette méthode peut être effectuée en parallèle sur les différents sous-domaines. Le théorème de convergence de cette méthode est démontré dans [78].

2.1.2 Avec recouvrement

Pour la méthode de décomposition de domaine avec recouvrement, nous subdivisons le domaine Ω en deux sous-domaines Ω_1 et Ω_2 tels que $\Omega = \Omega_1 \cup \Omega_2$ et $\Omega_1 \cap \Omega_2 \neq \emptyset$ (voir figure 2.2).

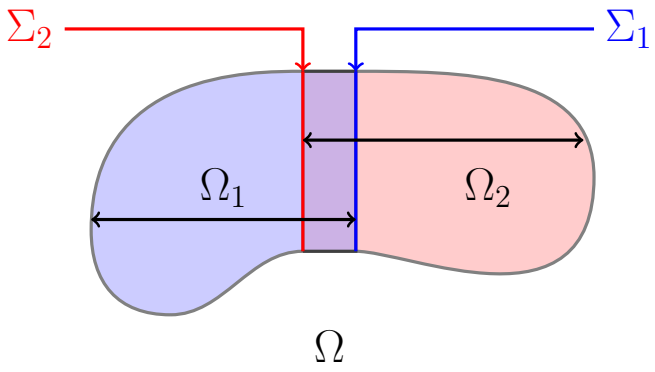


FIGURE 2.2 – Décomposition du domaine Ω avec recouvrement.

Parmi les méthodes de Schwarz avec recouvrement, on distingue les méthodes multiplicatives et les méthodes additives. La première, dite *multiplicative*, est une méthode itérative dont le but est de résoudre alternativement, à l'étape n, les deux problèmes

$$\begin{cases} Lu^n_{|\Omega_1} = f & \text{dans } \Omega_1 \\ u^n_{|\Omega_1} = g & \text{sur } \partial\Omega_1 \backslash \Sigma_1 \\ u^n_{|\Omega_1} = u^{n-1}_{|\Omega_2} & \text{sur } \Sigma_1 \end{cases}$$

et

$$\begin{cases} Lu^n_{|\Omega_2} = f & \text{dans } \Omega_2 \\ u^n_{|\Omega_2} = g & \text{sur } \partial\Omega_2 \backslash \Sigma_2 \\ u^n_{|\Omega_2} = u^n_{|\Omega_1} & \text{sur } \Sigma_2. \end{cases}$$

Cette méthode permet la résolution du problème de manière indépendante sur les deux sous-domaines. Cependant, la dépendance du terme $u^n_{|\Omega_1}$ dans le deuxième problème empêche une résolution en parallèle. L'intérêt de cette méthode réside dans le fait de pouvoir réduire la taille des systèmes à résoudre en découpant le domaine.

La méthode de Schwarz *additive* est très semblable à la méthode *multiplicative*. Elle consiste à résoudre les deux problèmes

$$\begin{cases} Lu^n_{|\Omega_1} = f & \text{dans } \Omega_1 \\ u^n_{|\Omega_1} = g & \text{sur } \partial\Omega_1 \backslash \Sigma_1 \\ u^n_{|\Omega_1} = u^{n-1}_{|\Omega_2} & \text{sur } \Sigma_1 \end{cases}$$

et

$$\begin{cases} Lu^n_{|\Omega_2} = f & \text{dans } \Omega_2 \\ u^n_{|\Omega_2} = g & \text{sur } \partial\Omega_2 \backslash \Sigma_2 \\ u^n_{|\Omega_2} = u^{n-1}_{|\Omega_1} & \text{sur } \Sigma_2. \end{cases}$$

Dans cette variante, le terme $u^n_{|\Omega_1}$ qui impliquait une dépendance entre les systèmes est remplacé par $u^{n-1}_{|\Omega_1}$, ce qui autorise alors le calcul en parallèle sur les sous-domaines. A la fin d'une itération, les solutions $u_{|\Omega_i}$ sur Σ sont échangées afin de pouvoir passer à l'itération $n+1$. L'étude de la convergence de ces schémas peut être trouvée dans [43]. Dans le chapitre 4, nous verrons que la vitesse de cette convergence est proportionnelle à la taille du recouvrement. En effet, plus celui-ci est grand, plus la convergence est rapide.

Bien que la méthode sans recouvrement permette de résoudre des problèmes plus petits et évite la redondance de calculs, elle reste plus complexe. C'est pourquoi nous lui

préférerons la méthode additive de Schwarz avec recouvrement. De plus, nous verrons que dans notre cas, un recouvrement de l'ordre d'une dizaine de pixels est suffisant pour obtenir la convergence rapide de l'algorithme.

2.2 Méthode pararéelle

2.2.1 Présentation de l'algorithme

Le terme "pararéel" a été introduit par J.-L. Lions, Y. Maday and G. Turinici [77]. Le but de cette méthode est de coupler une résolution grossière et une résolution fine, où la résolution fine est effectuée en parallèle. Nous utilisons ensuite un processus de prédiction-correction [11].

Afin de pouvoir illustrer les étapes de la méthode pararéelle, nous allons dans un premier temps expliciter la résolution d'un problème plus simple que celui du flot optique. Nous considérons donc la résolution du problème

$$\begin{cases} \dfrac{\partial u}{\partial t} = \cos(t) \\ u(0) = 0. \end{cases} \tag{2.2}$$

dont la solution exacte recherchée est la fonction sinus.

Soit $0 = T_0 < T_1 < \cdots < T_N = T$ une partition de $[0, T]$. La première étape du pararéel consiste à résoudre le problème à l'aide d'un solveur grossier sur $[0, T]$ avec un large pas de temps ΔT. Ensuite, sur chaque intervalle $[T_n, T_{n+1}]$, nous résolvons le problème avec un solveur plus fin (au moins aussi précis que le solveur grossier). Le pas de temps de ce solveur est noté Δt et vérifie $\Delta t < \Delta T$. Nous noterons dans la suite $\mathcal{G}$ le solveur grossier et $\mathcal{F}$ le solveur fin du problème. On considère $\Delta T = \frac{T}{N}$ et $\Delta t = \frac{\Delta T}{M}$ les pas de temps grossiers et fins correspondants où N et M représentent les nombres totaux de pas de temps respectifs.

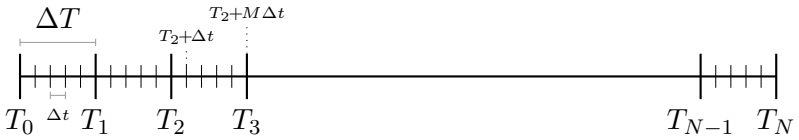


FIGURE 2.3 – Représentation de la partition de $[0, T]$.

Dans notre exemple, nous choisissons $T = 10$, $N = 10$ et $M = 50$. Il en découle que $\Delta T = 1$ et $\Delta t = 1/50$. Pour les solveurs $\mathcal{G}$ et $\mathcal{F}$, nous prenons dans les deux cas un

solveur d'Euler explicite. La précision du solveur fin est supérieure à celle du solveur grossier grâce à son pas de temps cinquante fois plus petit. On a donc les deux solveurs donnés par

$$u_{n+1} = \mathcal{G}(u_n) = u_n + \cos(n * \Delta T) * \Delta T$$
$$u_{m+1} = \mathcal{F}(u_m) = u_m + \cos(m * \Delta t) * \Delta t.$$

Dans la suite, pour des raisons de clarté, nous noterons u^g et u^f les solutions respectives des solveurs grossier et fin.

Nous allons maintenant voir plus en détails les étapes de l'algorithme du pararéel pour la résolution du problème (2.2). On commence la méthode pararéelle par une première approximation grossière de la solution u sur $[0, T]$ en résolvant le problème

$$\begin{cases} u^g_{n+1} = \mathcal{G}(u^g_n), & n = 0, \dots, N \\ u^g_0 = u^g(T_0 = 0) = 0. \end{cases} \tag{2.3}$$

Les solutions obtenues par le solveur grossier ne sont pas précises (figure 2.4) mais le solveur présente l'avantage d'être rapide. En effet, cette partie étant effectuée en séquentiel, il est important qu'elle prenne le moins de temps possible. Si la précision n'a pas d'importance pour la résolution, le solveur doit en revanche être un solveur stable.

Après cette première résolution, nous commençons la partie itérative globale de la méthode. Sur chaque intervalle $[T_n, T_{n+1}]$, nous résolvons le problème fin en prenant comme condition initiale $u^f_{n,0} = u^g(T_n)$ (voir figure 2.5). Nous avons ainsi

$$\begin{cases} u^f_{n,m+1} = \mathcal{F}(u^f_{n,m}), & m = 0, \dots, M \\ u^f_{n,0} = u^g_n \end{cases} \tag{2.4}$$

Remarque 2.2.1. *Lors de l'itération k de la méthode pararéelle, les estimations sur les sous-intervalles $[T_n, T_{n+1}]$ avec $n \leq k$ sont optimales. En effet, pour le premier sous-intervalle, la condition initiale est la condition initiale réelle du problème. En appliquant le solveur fin, la solution trouvée sur ce sous-intervalle est la même que celle qu'aurait trouvé une méthode classique séquentielle avec un solveur fin. Après la première itération, il suffit de mettre à jour la solution u^g_1 en la remplaçant par $u^f_{0,M}$. La suite du raisonnement se fait par récurrence.*

D'après la remarque 2.2.1, à chaque itération globale, nous pourrions simplement mettre à jour les valeurs u^g_n pour tout n avec les dernières solutions fines $u^f_{n-1,M}$ des sous-intervalles

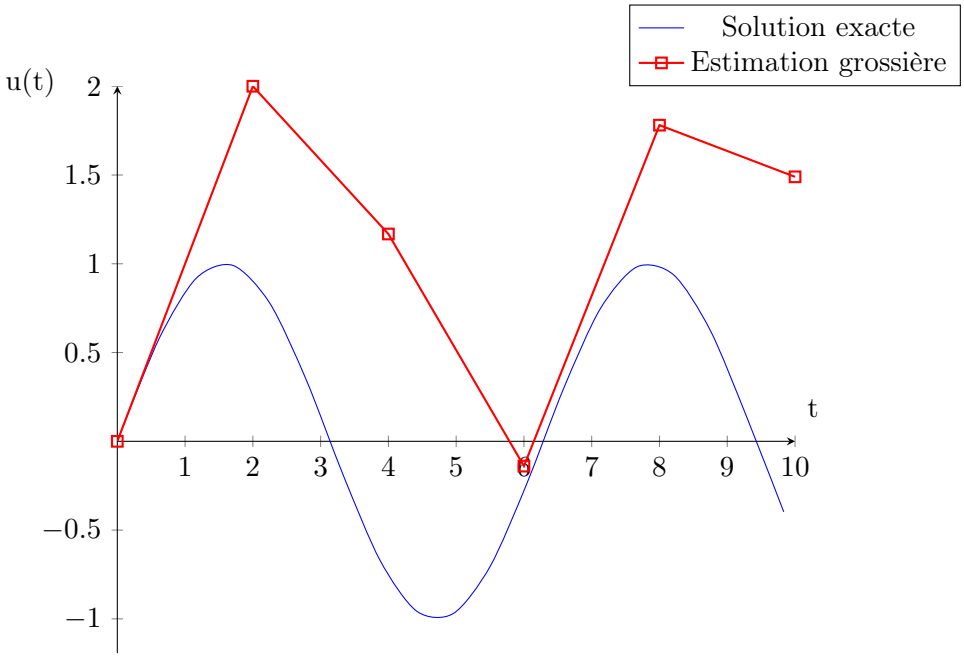


FIGURE 2.4 – Représentation de la résolution grossière initiale de la méthode pararéelle (rouge) et de la solution exacte recherchée (bleu).

$[T^{n-1}, T^n]$ correspondants. Cependant, afin d'améliorer la rapidité de convergence de l'algorithme pararéel vers la solution donnée par la méthode séquentielle itérative classique, nous utilisons une méthode de prédiction-correction [11].

Notons qu'à l'itération globale $k < N$ fixée, seuls les u_n^g pour n allant de k à N sont mis à jour. Les u_n^g pour n allant de 0 à $k-1$ sont déjà optimaux d'après la remarque précédente.

Pour simplifier la compréhension de cette partie, nous ajoutons un indice concernant l'itération globale concernée. Ainsi, nous noterons $u_n^{g,k}$, la $n^{\text{ième}}$ solution grossière à l'itération globale k. Lors de l'itération globale $k < N$, l'élément $u_k^{g,k}$ peut être mis à jour de manière optimale avec l'affectation

$$u_k^{g,k} = u_{k-1,M}^{f,k}$$

et servira de point de départ pour l'étape de prédiction. Lors de cette étape, chaque terme $u_n^{g,k}$ pour $n > k$ est remplacé par une prédiction calculée en effectuant une nouvelle résolution grossière. Ainsi, pour tout $n > k$

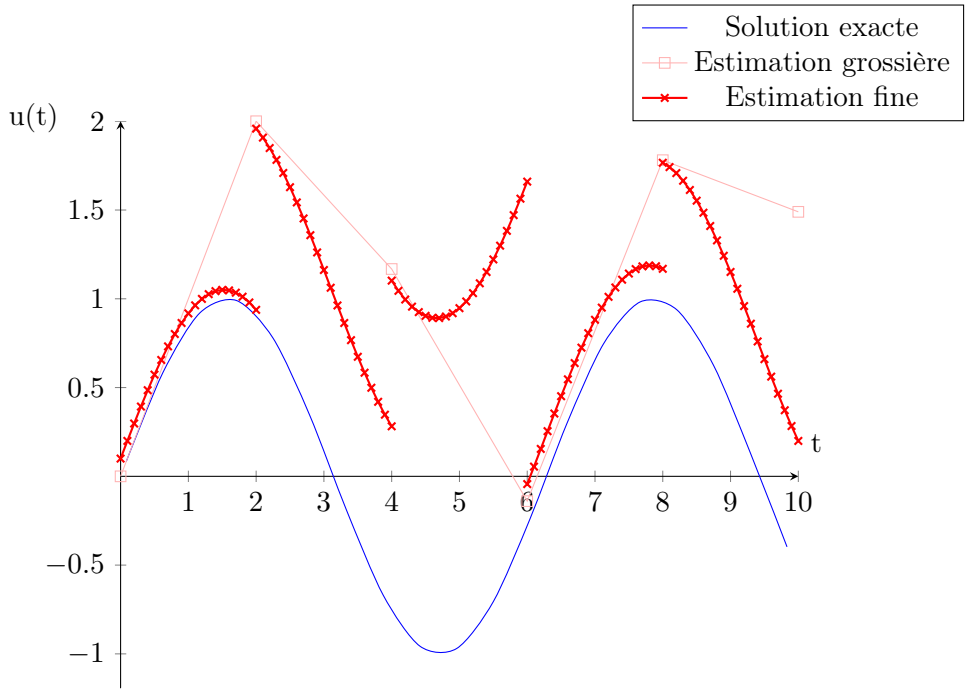


FIGURE 2.5 – Représentation de la première résolution fine de la méthode pararéelle.

$$u_n^{g,k} = \mathcal{G}(u_{n-1}^{g,k}).$$

Cette valeur est ensuite corrigée en ajoutant la différence entre la solution déjà déterminée $u_n^{g,k-1}$ et la solution finale obtenue lors de la résolution fine $u_{n-1,M}^{f,k}$. Finallement, à l'itération globale k, pour tout $n > k$, $u_n^{g,k}$ est mis à jour à l'aide de la formule de prédiction-correction

$$u_n^{g,k} = \underbrace{\mathcal{G}(u_{n-1}^{g,k})}_{\text{prédiction}} + \underbrace{u_{n-1,M}^{f,k} - u_n^{g,k-1}}_{\text{correction}}.$$

Sur la figure 2.6, nous présentons la solution obtenue après la deuxième itération globale. Nous remarquons que les solutions des deux premiers sous-intervalles sont quasi-identiques aux solutions exactes correspondantes. Le reste des estimations s'est globalement rapproché de la courbe du sinus également.

Le nombre maximum d'itérations de la méthode est N. Cependant, faire N itérations correspondrait, d'après la remarque 2.2.1, au même résultat que l'algorithme séquentiel. Le programme serait même plus long à exécuter à cause des temps de communications

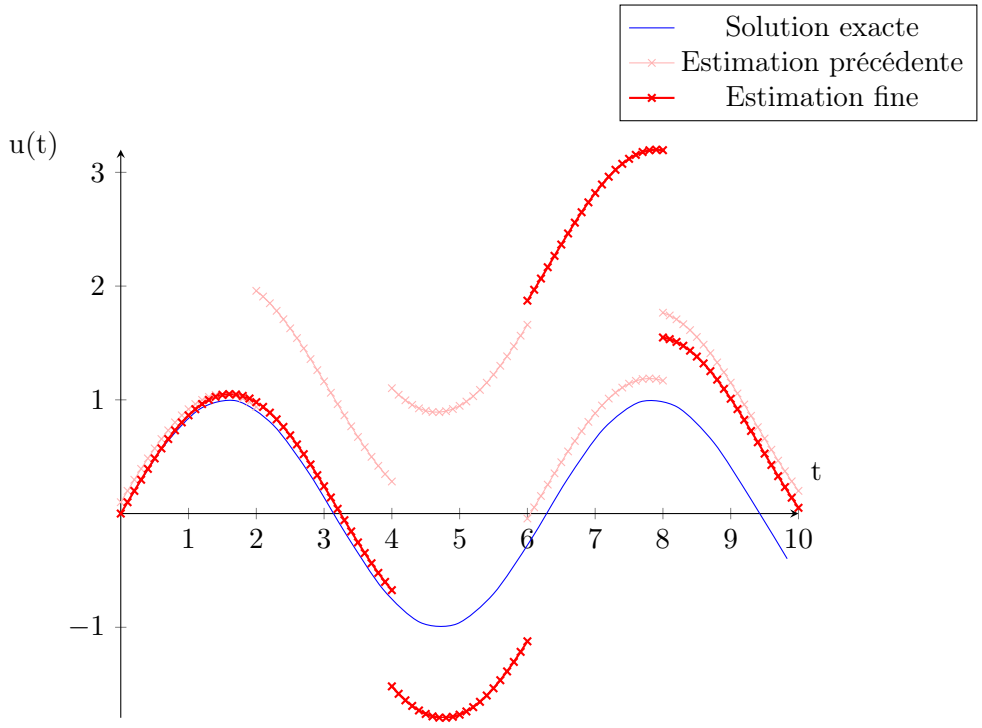


FIGURE 2.6 – Représentation de la deuxième itération globale de la méthode pararéelle.

entre les processeurs et de l'étape de prédiction-correction. Cependant, comme on peut le voir sur la figure 2.7, l'étape de mise à jour permet une bonne convergence des résultats. Dans notre exemple, dès la troisième itération, nous avons une bonne approximation de la solution. Afin de déterminer le moment où nous cessons les itérations, nous comparons le résultat avec celui de l'itération précédente à la fin de chaque itération de la méthode. Lorsque l'erreur entre deux itérations est en dessous du seuil choisi, nous arrêtons le programme. Comme on le voit sur l'exemple, la convergence est rapide.

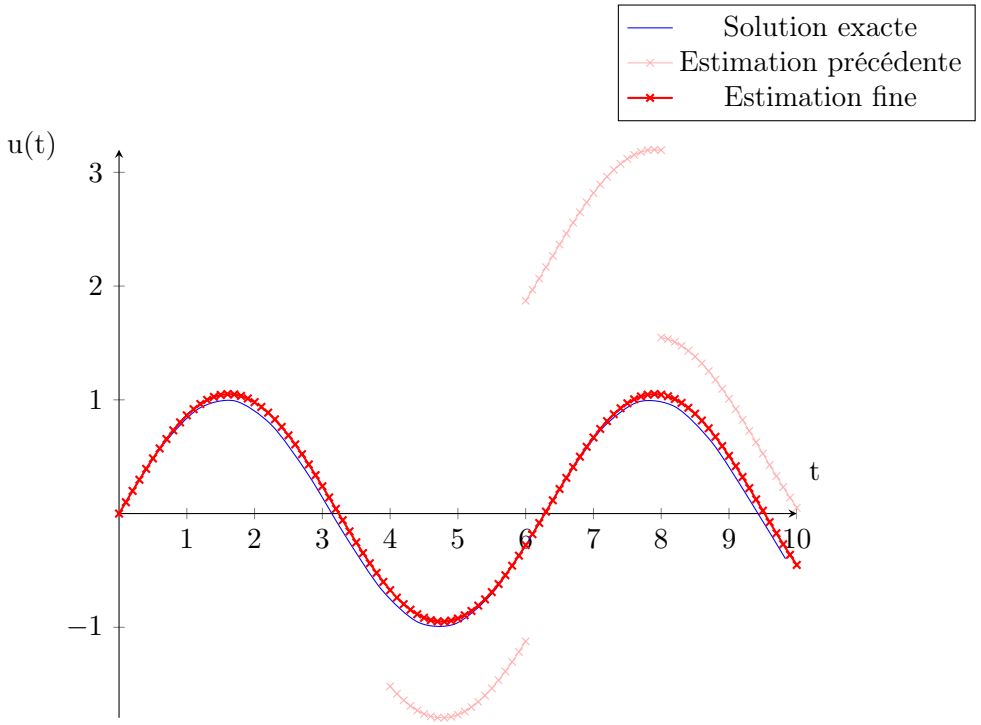


FIGURE 2.7 – Représentation de la troisième itération globale de la méthode pararéelle.

2.2.2 Pseudo-code de la méthode pararéelle

Afin de compléter l'explication, nous donnons un pseudo-code de l'algorithme expliquant dans ses grandes lignes le déroulement de l'algorithme. Nous utilisons les notations : "MPI" pour *Message Passing Interface* et "CPU" pour *Central Processing Unit*.

1. Initialisations :
$u^g_{old} = 0$
Résolution grossière :
Pour $n = 0$ à $N - 1$
$u^g = \mathcal{G}(u^g_{old})$
$u^g_{old} = u^g$
$u^g_{end}[n] = u^g$
$u^f_{start}[n+1] = u^g$
Fin pour
2. Boucle itérative globale pour la résolution pararéelle :
$k = 0$
Tant que erreur $>$ tolérance et $k < N$
a) Résolution fine parallèle (identique pour chaque CPU i) :
$u^f_{old} = u^f_{start}[i]$
Pour $m = 0$ à $M - 1$
$u^f = \mathcal{F}(u^f_{old})$
$u^f_{old} = u^f$
Fin pour
b) MPI Communications : partage des données
Chaque CPU envoie son u^f_{old} dans $u^f_{temp}[i]$
c) Calcul de l'erreur
erreur $= \|u^f_{temp}[mpisize] - u^f_{arret}\|_{L^2}$
$u^f_{arret} = u^f_{temp}[mpisize]$
d) Prédiction-correction :
$u^f_{start}[k+1] = u^f_{temp}[k]$
Pour $I = k + 1$ à $N - 1$
$u^g_{old} = u^f_{start}[I]$
$u^g = \mathcal{G}(u^g_{old})$
$u^g_{old} = u^g$
$\widetilde{u^g} = u^g_{end}[I]$
$u^g_{end}[I] = u^g_{old}$
$u^f_{start}[I+1] = u^g + u^f_{temp}[I] - \widetilde{u^g}$
Fin pour
$k++$
Fin tant que

2.2.3 Convergence de la méthode pararéelle

Afin de prouver la convergence de la méthode, nous comparons la solution obtenue par la méthode pararéelle avec celle obtenue par une méthode séquentielle dans laquelle nous effectuons $N \times M$ itérations avec le solveur fin. D'après la remarque 2.2.1, pendant la

première itération de la méthode, le premier processeur a le même point de départ que l'algorithme séquentiel et exécute M pas de temps fins. Ainsi, les M premiers pas sont les mêmes dans le programme séquentiel et dans le programme pararéel au cours de la première itération. Avant de commencer la deuxième itération globale, la condition initiale du deuxième processeur est mise à jour avec le résultat obtenu par le premier processeur. De cette manière, pour la deuxième itération, les résultats des deux premiers processeurs sont les mêmes que les résultats du programme séquentiel et ainsi de suite. Nous souhaitons donc obtenir une erreur nulle après N itérations de la méthode. De plus, pour que l'algorithme soit efficace, nous souhaitons obtenir une erreur proche de zéro en un minimum d'itérations. Sur la Fig. 2.8, on montre l'évolution de l'erreur en norme L^2 entre le programme séquentiel et le pararéel pour chaque itération globale. La partie prédiction-correction améliore la précision de la solution obtenue pour tous les processeurs suivants.

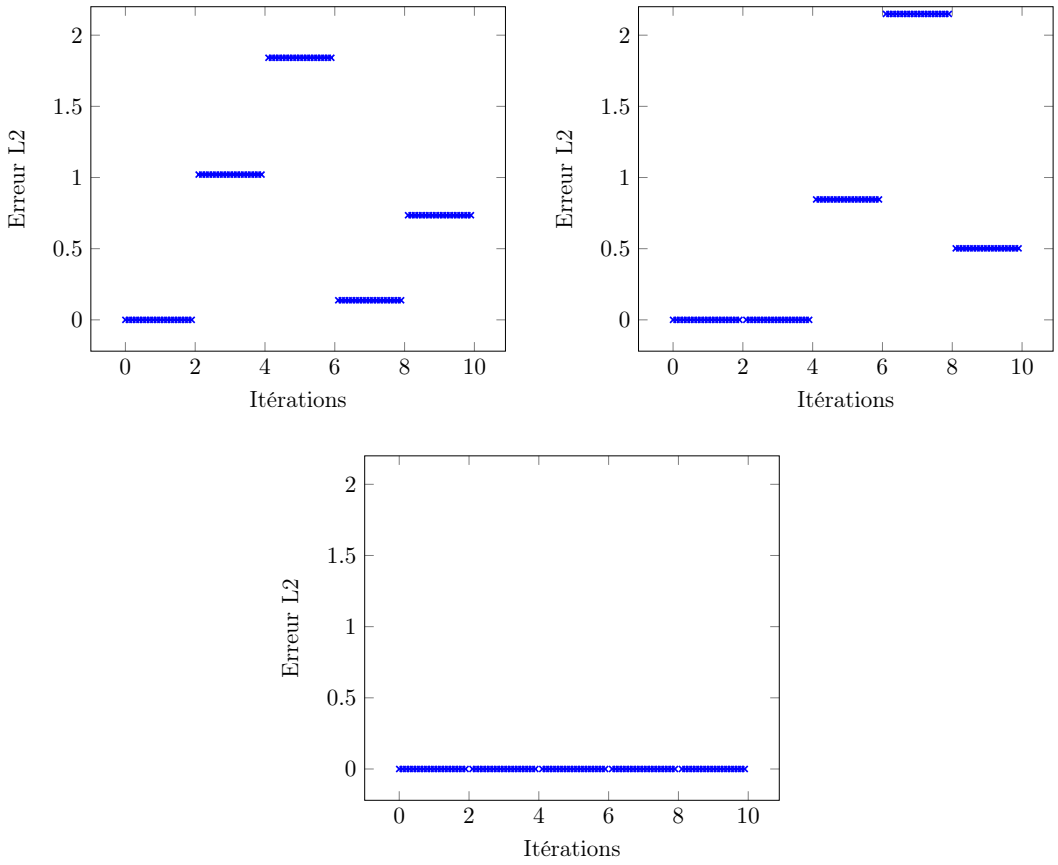


FIGURE 2.8 – Erreur en norme L^2 entre la solution obtenue avec la méthode pararéelle et l'algorithme séquentiel classique.

Comme nous l'avons dit précédemment, dans notre exemple nous pouvons arrêter le code après trois itérations. Comme chaque sous-intervalle comporte vingt itérations fines

réalisées en parallèle, nous avons donc effectué soixante itérations fines contre cent pour l'algorithme séquentiel. A cela on ajoute les temps de transfert de données et les itérations grossières rapides effectuées à l'initialisation et lors de l'étape de prédiction-correction. On peut donc espérer dans cet exemple un gain de temps de 20%. Nous nous rendons donc compte que la méthode pararéelle ne présentent un réel avantage que si le nombre de processeurs utilisés est important et si il existe un solveur grossier suffisamment précis pour permettre une convergence rapide de la méthode tout en restant rapide par rapport au solveur fin.

2.3 Conclusion

Les méthodes de calculs parallèles font désormais partie intégrante du domaine de la simulation numérique. Dans ce chapitre, nous avons présenté certaines des méthodes les plus utilisées pour la parallélisation d'algorithmes que nous allons appliquer dans le cas du problème du flot optique. En particulier, nous avons présenté les méthodes de décomposition de domaine avec et sans recouvrement. Dans la suite de ce manuscrit nous choisirons la méthode additive de Schwarz avec recouvrement, pour sa simplicité. Nous avons également développé dans ce chapitre la méthode parallèle en temps appelée pararéelle. Nous avons vu que cette méthode, relativement complexe, pouvait permettre une réduction du temps de calcul en couplant les solveurs grossier et fin à une étape de prédiction-correction.

Chapitre 3

Estimation du flot optique en petits déplacements

Sommaire

A l'instar de nombreux auteurs [15, 28, 30, 31, 99], nous utilisons une méthode variationnelle afin de résoudre le problème du flot optique. Cependant, au lieu d'utiliser une méthode de différences finies comme ce qui se fait en général, nous choisissons une méthode d'éléments finis. Bien que plus coûteuse, elle a l'avantage de nous permettre d'utiliser une méthode de contrôle adaptatif pour le paramètre de régularisation. Ce contrôle permet

d'améliorer l'estimation du résultat au niveau des arêtes de l'image. De plus, nous pourrons également utiliser une adaptation de maillage afin de réduire la taille du problème de manière significative. Dans ce chapitre, nous commençons par la mise en place de la méthode des éléments finis pour les petits déplacements. Pour cela, nous utilisons la statégie locale-globale de Bruhn et Weickert [31] en appliquant la méthode de contrôle local de la régularisation du paramètre α [15]. Nous décrivons également de manière détaillée les équations du flot optique dans le cas de la prise en compte de la variation de la luminosité ainsi que les méthodes utilisées pour le contrôle local du paramètre de régularisation.

3.1 Problème du flot optique

3.1.1 Luminosité constante

Une des méthodes efficaces pour l'étude du mouvement en petits déplacements est la stratégie locale-globale proposée par Bruhn et Weickert [31]. Elle combine la méthode locale de Lucas et Kanade [80] supposant que le déplacement est local par blocs et la méthode globale de Horn et Schunck [69] qui considère régulier le champ de vecteurs à déterminer. En plus de cette méthode, nous utilisons la méthode de régularisation adaptative [15]. Nous considérons donc désormais $\alpha(x, y)$, une fonction constante par morceaux, à la place du paramètre constant α utilisé dans la littérature. La fonctionnelle à minimiser est ainsi composée d'un terme d'adéquation aux données de l'image et d'un terme de régularisation du flot optique

$$\int_\Omega \underbrace{(f_x u_1 + f_y u_2 + f_t)^2}_{\text{données}} + \underbrace{\alpha(x,y)(|\nabla u_1|^2 + |\nabla u_2|^2)}_{\text{régularisation}} dxdy. \tag{3.1}$$

Remarque 3.1.1. *Pour une meilleure lisibilité, nous considérons implicitement que les fonctions f_x, f_y, et f_t sont convoluées par K_ρ, une gaussienne de déviation ρ afin de respecter l'hypothèse locale de Lucas et Kanade (voir chapitre 1).*

Le système d'Euler-Lagrange correspondant au problème de minimisation (3.1) est donné par

$$\begin{cases} (f_x)^2 u_1 + (f_x f_y) u_2 - \operatorname{div}(\alpha(x,y)\nabla u_1) = -f_x f_t \\ (f_y f_x) u_1 + (f_y)^2 u_2 - \operatorname{div}(\alpha(x,y)\nabla u_2) = -f_y f_t \end{cases}$$

et s'écrit sous forme vectorielle

$$\begin{cases} -\mathrm{div}(\alpha(x,y)\nabla \mathbf{u}) + J_\rho \mathbf{u} = \mathbf{f} \text{ sur } \Omega \\ \dfrac{\partial \mathbf{u}}{\partial n} = 0 \text{ sur } \partial\Omega \end{cases} \tag{3.2}$$

avec

$$J_\rho = \begin{bmatrix} f_x^2 & f_x f_y \\ f_y f_x & f_y^2 \end{bmatrix} \text{ et } \mathbf{f} = \begin{bmatrix} -f_x f_t \\ -f_y f_t \end{bmatrix}.$$

Nous notons cette méthode, la méthode $\mathrm{REGU}_{\mathrm{loc}}$. Nous étudierons l'existence et l'unicité de ce problème dans le cas, plus général, où nous utilisons le modèle prenant en compte la luminosité variable.

3.1.2 Prise en compte de la luminosité

En suivant le travail de Gennert and Negahdaripour dans [49] (voir chapitre 1), nous considérons que l'intensité lumineuse entre deux images de la séquence peut être représentée par une transformation affine. Contrairement à leurs travaux, nous supposons dans notre cas que le coefficient de translation C est négligeable. La nouvelle contrainte du problème du flot optique prenant en compte la variation de la luminosité s'écrit

$$f_x u_1 + f_y u_2 + f_t - m_t f = 0. \tag{3.3}$$

Comme dans les cas précédents, nous utilisons la combinaison des hypothèses locale et globale afin de lever le problème de l'ouverture et, comme nous l'avons fait pour les composantes u_1 et u_2, nous imposons une régularisation sur le gradient de m_t. La contrainte de régularisation sur ∇m_t est contrôlée par une fonction constante par morceaux $\lambda(x,y)$ définie, comme pour la fonction $\alpha(x,y)$, sur une subdivision de l'espace Ω par

$$\lambda = \lambda_\ell \text{ sur } \Omega_\ell,\ \lambda_\ell > 0,\ \forall \ell.$$

Bien que cette subdivision ne soit en pratique pas la même que celle associée à la fonction $\alpha(x,y)$, pour plus de simplicité dans l'écriture, nous utilisons les mêmes notations. De plus, nous considèrerons dans un premier temps que la fonction $\lambda(x,y)$ est constante et vaut λ.

Finalement, le problème du flot optique avec contrôle local de la régularisation et prenant en compte la variation de la luminosité revient à minimiser la fonctionnelle suivante

$$\int_\Omega (f_x u_1 + f_y u_2 + f_t - m_t f)^2 + \alpha(x,y)(|\nabla u_1|^2 + |\nabla u_2|^2) + \lambda(x,y)|\nabla m_t|^2 dxdy. \quad (3.4)$$

Le système étendu de Euler-Lagrange de ce nouveau problème est donné par

$$\begin{cases} (f_x)^2 u_1 + (f_x f_y) u_2 - (f_x f) m_t - \mathrm{div}(\alpha(x,y)\nabla u_1) = -f_x f_t \\ (f_y f_x) u_1 + (f_y)^2 u_2 - (f_y f) m_t - \mathrm{div}(\alpha(x,y)\nabla u_2) = -f_y f_t \\ -(f f_x) u_1 - (f f y) u_2 + (f^2) m_t - \mathrm{div}(\lambda(x,y)\nabla m_t) = f f_t \end{cases}$$

que nous réécrivons sous forme vectorielle

$$\begin{cases} -\mathrm{div}(\mathbf{\Lambda}(x,y)\nabla \mathbf{U}) + \mathbf{A}\mathbf{U} = \mathbf{F} \text{ dans } \Omega & (3.5) \\ \dfrac{\partial \mathbf{U}}{\partial n} = 0 \text{ sur } \partial\Omega & (3.6) \end{cases}$$

avec

$$\mathbf{\Lambda}(x,y) = \begin{bmatrix} \alpha(x,y) & 0 & 0 \\ 0 & \alpha(x,y) & 0 \\ 0 & 0 & \lambda(x,y) \end{bmatrix}, \mathbf{U} = \begin{bmatrix} u_1 \\ u_2 \\ m_t \end{bmatrix}, \mathbf{A} = \begin{bmatrix} f_x^2 & f_x f_y & -f_x f \\ f_y f_x & f_y^2 & -f_y f \\ -f_x f & -f_y f & f^2 \end{bmatrix}$$

et

$$\mathbf{F} = \begin{bmatrix} -f_x f_t \\ -f_y f_t \\ f f_t \end{bmatrix}.$$

Cette méthode prenant en compte les changements de luminosité, basée sur la méthode $\mathrm{REGU}_{\mathrm{loc}}$, sera nommée LUM-$\mathrm{REGU}_{\mathrm{loc}}$.

3.1.2.1 Existence et unicité d'une solution

Afin de prouver l'existence et l'unicité de la solution du problème (3.5), nous utilisons le théorème de Lax-Milgram. Nous commençons par déterminer la formulation variationnelle du problème. Nous nous plaçons dans l'espace de Hilbert $H^1(\Omega)$ et nous considèrons

une fonction test $\mathbf{V} \in H^1(\Omega)$. Pour trouver la formulation variationnelle, nous multiplions l'équation (3.5) par la fonction test $\mathbf{V}$ et nous intégrons sur le domaine Ω. Nous obtenons alors

$$-\int_\Omega \operatorname{div} \mathbf{\Lambda}(x,y)\nabla\mathbf{U}\mathbf{V}dxdy + \int_\Omega \mathbf{V}^t\mathbf{A}\mathbf{U}dxdy = \int_\Omega \mathbf{F}\mathbf{V}dxdy.$$

En utilisant la formule de Green, nous avons

$$\int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}\nabla\mathbf{V}dxdy + \int_\Omega \mathbf{V}^t\mathbf{A}\mathbf{U}dxdy - \int_{\partial\Omega} \mathbf{V}\nabla\mathbf{U}\cdot\mathbf{n}d\gamma(x,y) = \int_\Omega \mathbf{F}\mathbf{V}dxdy.$$

Or, comme

$$\int_{\partial\Omega} \mathbf{V}\nabla\mathbf{U}\cdot\mathbf{n}d\gamma(x,y) = 0,$$

la formulation variationnelle du problème du flot optique avec la méthode LUM-REGU$_{\text{loc}}$ s'écrit ainsi

$$\begin{cases} \text{Trouver } \mathbf{U}_{\mathbf{\Lambda}} \in H^1(\Omega) \text{ tel que } \forall \mathbf{V} \in H^1(\Omega) \\ a_{\mathbf{\Lambda}}(\mathbf{U}_{\mathbf{\Lambda}}, \mathbf{V}) = L(\mathbf{V}) \end{cases} \tag{3.7}$$

avec

$$a_{\mathbf{\Lambda}}(\mathbf{U}_{\mathbf{\Lambda}}, \mathbf{V}) = \int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}_{\mathbf{\Lambda}}\nabla\mathbf{V}dxdy + \int_\Omega \mathbf{V}^t\mathbf{A}\mathbf{U}_{\mathbf{\Lambda}}dxdy$$

et

$$L(\mathbf{V}) = \int_\Omega \mathbf{F}\mathbf{V}dxdy.$$

Si $\mathbf{A}$ est définie positive, on peut aisément vérifier les hypothèses de Lax-Milgram et en déduire l'existence et l'unicité de la solution. Cependant, dans le cas contraire, on cherche $\mathbf{U}$ dans l'espace $\mathcal{L}^{1,2}(\Omega)$ définit par

$$\mathcal{L}^{1,2}(\Omega) = \{\mathbf{U} \in L^2_{loc}(\Omega); \nabla\mathbf{U} \in L^2(\Omega), \int_\Omega \mid \mathbf{A}^{\frac{1}{2}}\mathbf{U} \mid^2 dxdy < \infty\}$$

(cf. [15]). On définit la relation d'équivalence

$$\mathbf{U}\mathcal{R}\mathbf{V} \leftrightarrow \int_\Omega \mid \nabla(\mathbf{U}-\mathbf{V}) \mid^2 dxdy + \int_\Omega (\mathbf{U}-\mathbf{V})^t\mathbf{A}(\mathbf{U}-\mathbf{V})dxdy = 0.$$

L'espace quotient $L^{1,2}(\Omega) = (\mathcal{L}^{1,2}(\Omega))/\mathcal{R}$ est un espace de Hilbert pour le produit scalaire

$$< \mathbf{U}, \mathbf{V} >_{L^{1,2}} = \int_\Omega \nabla\mathbf{U}\nabla\mathbf{V}dxdy + \int_\Omega \mathbf{V}^t\mathbf{A}\mathbf{U}dxdy$$

munit de la norme

$$\|\mathbf{U}\|_{L^{1,2}} = (< \mathbf{U}, \mathbf{U} >_{L^{1,2}})^{\frac{1}{2}}.$$

Dans cet espace, l'existence et l'unicité de la solution faible s'obtient également par le théorème de Lax-Milgram. Comme $\alpha > 0$ et $\lambda > 0$, la coercivité et la continuité de $a_{\mathbf{\Lambda}}$ est immédiate. Le seul point à vérifier est la continuité de L($\mathbf{V}$). En suivant [15], nous réécrivons $\mathbf{F}$ sous la forme $\mathbf{F} = \mathbf{AG} + \mathbf{H}$, où $\mathbf{H}$ est tel que $\mathbf{A}^{\frac{1}{2}}\mathbf{H} \in L^2(\Omega)$, $\sup \mathbf{G} \subset \Omega$ et $\int_\omega \mathbf{G}dxdy = 0$ pour toute composante connexe ω de Ω. En utilisant l'inégalité de Cauchy-Schwarz puis le théorème de la trace, on a

$$\begin{aligned}
\mid L(\mathbf{V}) \mid &\leqslant \int_\Omega \mid \mathbf{FV} \mid dxdy \\
&\leqslant \int_\Omega \mid \mathbf{U}^t\mathbf{AH} \mid dxdy + \int_\Omega \mid \mathbf{GU} \mid dxdy \\
&\leqslant \left(\int_\Omega \mathbf{AH}^2 dxdy\right)^{\frac{1}{2}} \left(\int_\Omega \mathbf{AU}^2 dxdy\right)^{\frac{1}{2}} + C\left(\int_\omega \mid \nabla\mathbf{U} \mid^2 dxdy\right)^{\frac{1}{2}} \left(\int_\omega \mid \mathbf{G} \mid^2 dxdy\right)^{\frac{1}{2}} \\
&\leqslant C \|\mathbf{U}\|_{L^{1,2}(\Omega)}.
\end{aligned}$$

où C est la constante de l'inégalité de Poincaré-Wirtinger appliquée dans $H^1(\omega, \mathbb{R}^2)\backslash\mathbb{R}^2$ et l'ensemble lisse ω est tel que $\sup \mathbf{G} \subset \omega \subset \Omega$ [45]. L est donc linéaire.

Ainsi, d'après le théorème de Lax-Milgram, le problème faible (3.7) admet une unique solution. Les problèmes (3.5) et (3.7) étant équivalents, nous en concluons qu'il existe

une unique solution $\mathbf{U_\Lambda} \in H^1(\Omega)$ ou dans $L^{1,2}$ pour le problème (3.5). Dans la suite de ce manuscrit, afin de simplifier les notations, nous utiliserons l'écriture $\mathbf{\Lambda} = \mathbf{\Lambda}(x, y)$. On en déduit :

Théorème 3.1.1. *Le problème (3.5) admet une solution unique* $\mathbf{U_\Lambda}$ *dans* H^1 *si* $\mathbf{A}$ *est définie positive et dans* $L^{1,2}(\Omega, \mathbb{R}^2)$ *sinon.*

Nous définissons la norme

$$\|\mathbf{U}\|^2_{\rho,\mathbf{\Lambda}} = \|\mathbf{\Lambda}^{\frac{1}{2}}\nabla\mathbf{U}\|^2_{L^2(\Omega)} + \|\mathbf{A}^{\frac{1}{2}}\mathbf{U}\|^2_{L^2(\Omega)}$$

où $\|\mathbf{A}^{\frac{1}{2}}\mathbf{U}\|^2_{L^2(\Omega)} =< A\mathbf{U}, \mathbf{U} >$. En notant, $\beta_M = \max(\alpha_{max}, \lambda_{max})$ et $\beta_m = \min(\alpha_{min}, \lambda_{min})$, nous pouvons étendre le résultat ([15], proposition 2.7) pour déduire les estimations suivantes.

Proposition 3.1.1. *Soit* $\boldsymbol{U}$ *une solution de (3.3) et* $\boldsymbol{U_\Lambda}$ *la solution du problème (3.7). Il existe une constante* $C > 0$ *indépendante de* $\mathbf{\Lambda}$ *telle que l'on ait les deux inégalités*

$$\|\boldsymbol{U_\Lambda}\|_{\rho,\mathbf{\Lambda}} \leqslant C\|\boldsymbol{A}^{\frac{1}{2}}\boldsymbol{U}\|_{L^2(\Omega)} \tag{3.8}$$

et, si $\boldsymbol{U} \in H^1(\Omega)$,

$$\|\boldsymbol{U} - \boldsymbol{U_\Lambda}\|_{\rho,\mathbf{\Lambda}} \leqslant C\left(\frac{\beta_M}{\beta_m}\right)^{\frac{1}{2}} \|\mathbf{\Lambda}^{\frac{1}{2}}\nabla\boldsymbol{U_\Lambda}\|_{L^2(\Omega)}. \tag{3.9}$$

Remarque 3.1.2. *L'estimation (3.9) n'est pas une estimation a priori, mais elle interviendra dans l'obtention d'estimations a posteriori.*

Démonstration. Comme $\mathbf{U}$ est solution du problème $\int_\Omega \mathbf{V}^t\mathbf{A}\mathbf{U}dxdy = \int_\Omega \mathbf{F}\mathbf{V}dxdy$ et que $\mathbf{U_\Lambda}$ est solution du problème (3.7), nous en déduisons que pour toute fonction test $\mathbf{V}$ dans $H^1(\Omega)$,

$$\int_\Omega \mathbf{\Lambda}(x, y)\nabla\mathbf{U_\Lambda}\nabla\mathbf{V}dxdy + \int_\Omega \mathbf{V}^t\mathbf{A}\mathbf{U_\Lambda}dxdy = \int_\Omega \mathbf{V}^t\mathbf{A}\mathbf{U}dxdy.$$

En particulier, pour $\mathbf{V} = \mathbf{U_\Lambda}$ nous avons

$$\int_\Omega \mathbf{\Lambda}(x, y)\nabla\mathbf{U}^2_\mathbf{\Lambda}dxdy + \int_\Omega \mathbf{U}^t_\mathbf{\Lambda}\mathbf{A}\mathbf{U_\Lambda}dxdy = \int_\Omega \mathbf{U}^t_\mathbf{\Lambda}\mathbf{A}\mathbf{U}dxdy.$$

A l'aide de l'inégalité de Cauchy-Schwarz, on en déduit

$$\|\mathbf{U}_{\mathbf{\Lambda}}\|_{\rho,\mathbf{\Lambda}} \leq \|\mathbf{A}^{\frac{1}{2}}\mathbf{U}\|_{L^2(\Omega)}\|\mathbf{A}^{\frac{1}{2}}\mathbf{U}_{\mathbf{\Lambda}}\|_{L^2(\Omega)}.$$

Cela nous conduit à l'inégalité (3.8).

De manière équivalente, en prenant $\mathbf{V} = \mathbf{U} - \mathbf{U}_{\mathbf{\Lambda}}$, nous avons

$$\int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}_{\mathbf{\Lambda}}\nabla(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy + \int_\Omega (\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})^t\mathbf{A}\mathbf{U}_{\mathbf{\Lambda}}dxdy = \int_\Omega (\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})^t\mathbf{A}\mathbf{U}dxdy$$

ou encore

$$\int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}_{\mathbf{\Lambda}}\nabla(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy = \int_\Omega (\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})^t\mathbf{A}(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy.$$

Sans perte de généralité, nous pouvons alors écrire l'équation

$$\begin{aligned}\int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}_{\mathbf{\Lambda}}\nabla(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy + \int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}\nabla(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy \\ - \int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}\nabla(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy = \int_\Omega (\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})^t\mathbf{A}(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy. \quad (3.10)\end{aligned}$$

Après regroupement des termes, nous obtenons

$$\begin{aligned}\int_\Omega \mathbf{\Lambda}(x,y)\nabla(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})^2dxdy + \int_\Omega (\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})^t\mathbf{A}(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy \\ = \int_\Omega \mathbf{\Lambda}(x,y)\nabla\mathbf{U}\nabla(\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}})dxdy.\end{aligned}$$

Finalement, en utilisant une nouvelle fois l'inégalité de Cauchy-Schwarz, nous obtenons l'inégalité

$$\|\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda}}\|_{\rho,\mathbf{\Lambda}} \leqslant \beta_M\|\nabla\mathbf{U}\|^2_{L^2(\Omega)} + \beta_M^{1+2\varepsilon}\|\nabla\mathbf{U}\|^2_{L^2(\Omega)} + \frac{\beta_M}{\beta_m}\beta_M^{-2\varepsilon}\|\mathbf{\Lambda}^{\frac{1}{2}}\nabla\mathbf{U}_{\mathbf{\Lambda}}\|^2_{L^2(\Omega)}$$

qui nous donne le résultat (3.9) recherché. □

Nous utiliserons l'équation (3.9) après la discrétisation lors de l'obtention de l'estimation a posteriori. Néanmoins, en analyse d'image, $\mathbf{U}$ n'est que rarement dans $H^1(\Omega)$ d'où la nécessité de trouver d'autres outils d'analyse et, dans notre cas, de se placer dans la topologie de la Γ-convergence.

3.1.3 Γ-convergence et fonctionnelle de Mumford-Shah

Le but de cette section est de montrer que par un choix judicieux du paramètre de régularisation, l'algorithme adaptatif converge au sens de la Γ-convergence vers la solution du problème du flot optique avec variation de luminosité. Les notations et définitions utiles à la compréhension de cette partie sont données dans l'annexe B.

Le modèle de Mumford-Shah est un des modèles les plus utilisés dans le domaine de la segmentation d'images. Dans le cas du problème du flot optique [38], la forme faible s'écrit : trouver la solution $\mathbf{U}$ qui minimise la fonctionnelle

$$\frac{1}{2}\int_\Omega \mid \nabla \mathbf{U} \mid^2 + \beta \mathcal{H}^1(S_{\mathbf{U}}) + (\frac{1}{2}\int_\Omega \mathbf{U}^t \mathbf{A}\mathbf{U} - \int_\Omega \mathbf{F}\mathbf{U}) \tag{3.11}$$

où β est une constante positive, $\mathcal{H}^1$ la mesure de Hausdorff de l'ensemble $S_{\mathbf{U}}$, ensemble des singularités essentielles de $\mathbf{U}$ [4]. Les minimiseurs de (3.11) sont des fonctions à variations bornées spéciales $SBV(\Omega, \mathbb{R}^2)$ (voir annexe B) et sont les solutions du problème de segmentation. Afin de gérer le terme $\mathcal{H}^1(S_{\mathbf{U}})$, Ambrosio et Tortorelli [5] proposent d'utiliser une approximation de la mesure à l'aide d'une famille de densités. Cette solution conduit à la gestion de nombreux paramètres (voir (1.8)), c'est pourquoi nous utilisons une approche adaptative consistant à contrôler la diffusion du paramètre de régularisation α à l'aide d'un indicateur d'erreur a posteriori [25, 32]. Le principe de cette méthode est de construire une famille discrète d'énergies dont les minimiseurs convergent vers ceux de (3.11).

Soient $\nu_0 \leqslant \frac{\pi}{3}$ et $c > 6$. On note $\forall \varepsilon > 0$, $\mathcal{T}_\varepsilon(\Omega)$ l'ensemble des maillages réguliers de Ω vérifiant que la longueur des arêtes des triangles $K \in \mathcal{T}_\varepsilon$ est comprise entre ε et $c\varepsilon$ et tels que les angles des triangles K soient supérieurs à ν_0.

Soit $V_\varepsilon(\Omega, \mathbb{R}^2)$ l'ensemble des fonctions continues $\mathbf{U}$ définies sur Ω à valeurs dans $\mathbb{R}^2$ tel que $\mathbf{U}$ soit un vecteur de trois fonctions affines sur chaque triangle K. L'ensemble des maillages tels que $\mathbf{U} \in V_\varepsilon(\Omega, \mathbb{R}^2)$ est noté $\mathcal{T}_\varepsilon(u)$. Nous définissons g une fonction continue non-décroissante de $\mathbb{R}^+$ dans $\mathbb{R}^+$ telle que

$$\lim_{t\to 0} \frac{g(t)}{t} = 1, \ \lim_{t\to\infty} g(t) = g_\infty.$$

Choisir $g(t) = \min(t, g_\infty)$ revient à utiliser la méthode de Black et Zissermann [23]. Dans l'analyse mathématique qui va suivre, nous considérerons $\forall t \geqslant 0$,

$$g(t) \leqslant \min(t, g_\infty).$$

Soient $\mathbf{U} \in (L^2(\Omega))^3$ et $\mathcal{T} \in \mathcal{T}_\varepsilon(\Omega)$ un maillage, nous posons

$$G_\varepsilon(\mathbf{U}, \mathcal{T}) = \begin{cases} \sum\limits_{K \in \mathcal{T}} \mid K \cap \Omega \mid \frac{1}{h_K} g(h_k \mid \nabla \mathbf{U} \mid^2) & si\ \mathbf{U} \in V_\varepsilon(\Omega), \mathcal{T} \in \mathcal{T}_\varepsilon(\Omega) \\ +\infty & sinon. \end{cases} \tag{3.12}$$

Dans le cas scalaire, d'après [32], si on pose

$$G_\varepsilon(u) = \min_{\mathcal{T} \in \mathcal{T}_\varepsilon(\Omega)} G_\varepsilon(u, \mathcal{T}) \tag{3.13}$$

alors pour ε tendant vers 0 et ν_0 suffisamment petit, $G_\varepsilon(u)$ Γ-converge vers la fonctionnelle de Mumford-Shah

$$G(u) = \begin{cases} \int_\Omega \mid \nabla u \mid^2 + g_\infty \mathcal{H}^1(S_u) & si\ u \in L^2(\Omega) \cap GSBV(\Omega) \\ +\infty & sinon. \end{cases}$$

Dans le cas vectoriel, la preuve est similaire. Elle est basée sur le théorème suivant ([25], Théorème 2).

Théorème 3.1.2. *Pour toute suite $(\boldsymbol{U}_\varepsilon)_\varepsilon$ telle que $\boldsymbol{U}_\varepsilon \in V_\varepsilon(\Omega)$ et*

$$\sup_{\varepsilon > 0} \left(G_\varepsilon(\boldsymbol{U}_\varepsilon) + \| \boldsymbol{U}_\varepsilon \|_{L^2(\Omega)} \right) < +\infty,$$

il existe $\boldsymbol{U} \in GSBV(\Omega)$ et une sous-suite $(\boldsymbol{U}_{\varepsilon_j})_{\varepsilon_j}$ convergente vers $\boldsymbol{U}$ telle que

$$G(\boldsymbol{U}) \leqslant \liminf_j G(\boldsymbol{U}_{\varepsilon_j}).$$

De plus, si pour tout ε, $\boldsymbol{U}_\varepsilon$ est une solution de

$$\min_{\boldsymbol{V}} \left(G_\varepsilon(\boldsymbol{V}) + \frac{1}{2}(\int_\Omega \boldsymbol{V}^t J_\rho \boldsymbol{V} dxdy - \int_\Omega \boldsymbol{F}\boldsymbol{V} dxdy) \right)$$

alors, $\boldsymbol{U}$ *est solution de*

$$\min_{\boldsymbol{V}} \left(G_\varepsilon(\boldsymbol{V}) + \frac{1}{2}(\int_\Omega \boldsymbol{V}^t \boldsymbol{A}\, \boldsymbol{V} dxdy - \int_\Omega \boldsymbol{F}\boldsymbol{V} dxdy) \right)$$

et la suite $(\boldsymbol{U}_{\varepsilon_j})_{\varepsilon_j}$ *converge fortement vers* $\boldsymbol{U}$*.*

Dans [25], il est montré que pour un maillage $\mathcal{T}_\varepsilon$, trouver le minimum de G_ε est un problème équivalent à trouver un minimum pour $\mathbf{U} \in V_\varepsilon(\Omega)$ et $\mathbf{V} = (\mathbf{V}_K)_{K\in\mathcal{T}_\varepsilon}$ de la fonction

$$G'_\varepsilon(\mathbf{U}, \mathbf{V}, \mathcal{T}_\varepsilon) = \sum_{K\in\mathcal{T}_\varepsilon} |K \cap \Omega| \left(\mathbf{V}_K \, |\nabla \mathbf{U}_K|^2 + \frac{\psi(\mathbf{V}_K)}{h_K} \right) \tag{3.14}$$

où ψ représente la transformé de Legendre-Fenchel de g. Finalement en prenant

$$\mathbf{V}_K = g'(h_K \, |\nabla \mathbf{U}_K|^2)$$

cela nous conduit à l'algorithme adaptatif que nous décrivons ci-dessous, après avoir discrétisé le problème.

3.1.4 Discrétisation

Soit un maillage triangulaire noté $\mathcal{T}_h$ où chaque pixel de l'image est représenté par deux triangles (figure 3.1).

Nous définissons l'espace d'approximation de la solution

$$\mathcal{V}_h = \{\mathbf{V}_h \in C(\overline{\Omega}), \quad \mathbf{V}_h|_K \in P_1(K)\}$$

où, h représente le diamètre maximal des cellules K, $C(\overline{\Omega})$ représente l'ensemble des fonctions continues sur $\overline{\Omega}$ et $P_1(K)$ est l'ensemble des fonctions polynômiales de degré inférieur ou égal à 1 sur K.

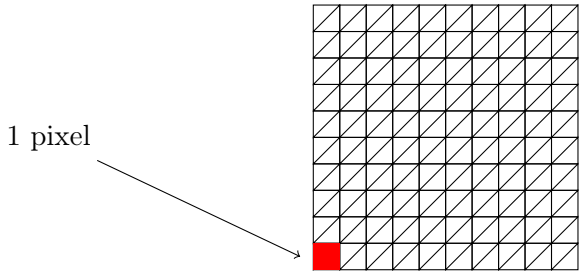


FIGURE 3.1 – Représentation du maillage initial $\mathcal{T}_h$.

Dans cet espace, nous notons $\mathbf{A}_h$ et $a_{\mathbf{\Lambda},h}$, les approximations respectives de $\mathbf{A}$ et $a_{\mathbf{\Lambda}}$, et $L_h(\boldsymbol{\varphi})$ l'approximation de $L(\boldsymbol{\varphi})$. La formulation variationnelle discrétisée consiste à trouver $\mathbf{U}_{\mathbf{\Lambda},h} \in \mathcal{V}_h$ tel que

$$a_{\mathbf{\Lambda},h}(\mathbf{U}_{\mathbf{\Lambda},h}, \mathbf{V}_h) = L_h(\mathbf{V}_h) \quad \forall \mathbf{V}_h \in \mathcal{V}_h. \tag{3.15}$$

Théorème 3.1.3. *Le problème variationnel discret (3.15) admet une solution unique.*

La preuve sera donnée dans la section suivante, après avoir énoncé le théorème 3.1.4.

3.1.5 Variation locale et adaptative du paramètre α

Afin de présenter la stratégie adaptative du paramètre de régularisation α, première et deuxième composante de $\mathbf{\Lambda}$, nous donnons dans un premier temps les différentes étapes de l'algorithme pour la méthode LUM-REGU$_{\text{loc}}$.

1. — Calcul d'une première approximation $\mathbf{U}^0$ du flot optique. Cette estimation est faite sur un maillage cartésien T_h^0 où nous avons deux mailles triangulaires par pixel.
   — Définition de $i = 0$.
2. $i = i + 1$. Construction d'un maillage adapté T_h^i.
3. Choix local de $\alpha^i(x, y)$ sur T_h^i.
4. Calcul de l'approximation $\mathbf{U}^i$ du flot optique sur le maillage T_h^i.
5. Retour à l'étape 2.

En ce qui concerne le troisième point de cet algorithme, le choix n'est pas unique. Un premier choix possible consiste à prendre la fonction

$$g(t) = \frac{2\mu}{\pi} \arctan(\frac{\pi t}{2\mu})$$

où μ est un paramètre réel constant. En considérant ensuite

$$\mathbf{V}_K = g'(h_K \mid \nabla \mathbf{U}_{\mathbf{\Lambda},h} \mid^2),$$

le paramètre de régularisation est dans ce cas réévalué à l'aide de l'expression

$$\alpha_K^{n+1} = \frac{\alpha_K^n}{1 + \kappa \left(\frac{\pi h_K |\nabla \mathbf{U}_{\mathbf{\Lambda},h}|^2}{2\mu} \right)^2}. \tag{3.16}$$

Ce choix implique une réduction lente et régulière de la régularisation α dans les zones où le terme $h_K \mid \nabla \mathbf{U}_{\mathbf{\Lambda},h} \mid^2$ est grand. Afin de donner plus de poids à ce terme dans l'équation, un réél constant κ est ajouté.

Dans notre cas, nous choisissons de réévaluer α en suivant la stratégie adaptative proposée dans [15]. L'indicateur d'erreur que nous utilisons est défini pour chaque élément $K \in \mathcal{T}_h$ par

$$\eta_K = \mathbf{\Lambda}_K^{-\frac{1}{2}} h_K \left\| \mathbf{F}_h + \operatorname{div}(\mathbf{\Lambda}_K \nabla \mathbf{U}_{\mathbf{\Lambda},h}) + \mathbf{A}_h \mathbf{U}_{\mathbf{\Lambda},h} \right\|_{L^2(K)} + \frac{1}{2} \sum_{e \in \varepsilon_K} \mathbf{\Lambda}_e^{-\frac{1}{2}} h_e^{\frac{1}{2}} \left\| [\mathbf{\Lambda} \nabla \mathbf{U}_{\mathbf{\Lambda},h} \cdot \mathbf{n}_e]_e \right\|_{L^2(e)} \tag{3.17}$$

où ε_K représente l'ensemble de tous les bords e de K. Le diamètre de K est noté h_K et le diamètre d'un bord e est h_e. $\mathbf{n}_e$ représente le vecteur normal sortant de e, $\mathbf{\Lambda}_e$ est le maximum (pour chaque composante) entre la valeur de $\mathbf{\Lambda}$ de deux voisins d'un bord, et $[.]_e$ représente le saut d'un bord e c'est-à-dire, la différence entre les valeurs intérieures et les valeurs extérieures à la cellule K.

L'indicateur d'erreur est équivalent à la norme H^1 de l'erreur des éléments finis [15, 115] et est principalement utilisé pour effectuer des adaptations de maillages. Cette valeur est élevée lorsque l'erreur résiduelle du modèle est importante ou lorsque la valeur moyenne du gradient est grande. C'est par exemple le cas sur les discontinuités puisque dans ce cas, la valeur de $\nabla U_{\mathbf{\Lambda},h}$ est élevée. Ainsi, pour améliorer la solution, nous réduisons la valeur du paramètre α des deux premières composantes de $\mathbf{\Lambda}$ sur ces zones. La troisième

composante λ est gardée constante. Le choix local du paramètre de régularisation α est ainsi donné par

$$\alpha_K^{n+1} = \max\left(\frac{\alpha_K^n}{1+\kappa\max\left(\frac{\eta_K}{\|\eta\|_\infty}-\frac{\zeta}{100},0\right)},\alpha_s\right) \tag{3.18}$$

où κ est un paramètre habituellement choisi entre 5 et 10, α_s est un paramètre seuil empêchant la régularisation d'être trop petite et $\zeta > 0$ correspond à la tolérance souhaitée. De cette façon, si l'erreur relative est plus grande que $\zeta\%$ nous réduisons la valeur de α. Sinon, si la valeur est plus petite que $\zeta\%$, le dénominateur étant égal à 1, α garde la même valeur qu'à l'itération précédente.

Dans l'approche adaptative, les estimations suivantes garantissent l'optimalité des estimateurs d'erreur a posteriori que nous utilisons [15].

Théorème 3.1.4. *Il existe une constante C indépendante de h et $\mathbf{\Lambda}$ telle que*

$$\|\boldsymbol{U}_{\mathbf{\Lambda}} - \boldsymbol{U}_{\mathbf{\Lambda},h}\|_{\rho,\mathbf{\Lambda}} \leq C\left(1+\frac{\beta_M}{\beta_m}\right)^{\frac{1}{2}}\left(\sum_{K\in\mathcal{T}_h}\eta_K^2 + h_K^2\mathbf{\Lambda}_K^{-1}\|\boldsymbol{F}-\boldsymbol{F}_h\|_{L^2(K)}^2\right)^{\frac{1}{2}}.$$

Nous pouvons ainsi combiner ce résultat avec l'inégalité (3.9), pour obtenir le résultat de convergence souhaité dans le théorème 3.1.3.

$$\|\mathbf{U}-\mathbf{U}_{\mathbf{\Lambda},h}\|_{\rho,\mathbf{\Lambda}} \leq C\left(1+\frac{\beta_M}{\beta_m}\right)^{\frac{1}{2}}\left(\sum_{K\in\mathcal{T}_h}\eta_K^2 + h_K^2\mathbf{\Lambda}_K^{-1}\|\mathbf{F}-\mathbf{F}_h\|_{L^2(K)}^2 + \mathbf{\Lambda}_K\|\nabla\mathbf{U}_{\mathbf{\Lambda}}\|_{L^2(K)}^2\right)^{\frac{1}{2}}.$$

3.2 Résultats numériques

Dans cette partie, après avoir présenté la méthode de représentation des champs de vecteurs résultats, nous présentons les résultats obtenus pour différents cas tests donnés par le site de *Middleburry*. La méthode que nous utilisons est celle présentée pour l'estimation du flot optique prenant en compte les variations de luminosité. Nous étudions la légitimité de cette méthode à travers un exemple dans lequel l'intensité lumineuse est variable. Nous présentons l'évolution de l'erreur angulaire et de l'erreur en norme L^2 par rapport à la solution exacte. Enfin, nous explicitons les avantages de l'utilisation de l'adaptation du maillage au cours des itérations.

3.2.1 Représentation des résultats

La manière la plus intuitive de réprésenter les résultats est d'afficher le champ de vecteur tel que présenté sur la figure 3.2.

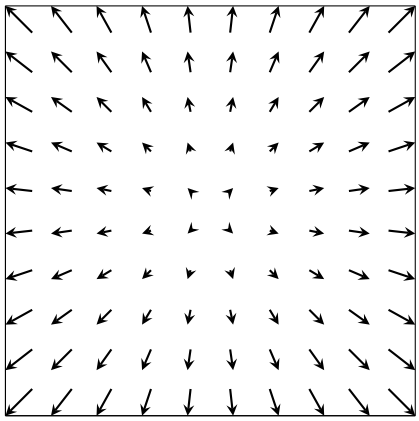


FIGURE 3.2 – Champ de vecteurs représentant les mouvements sur une image.

Dans un cas comme celui-ci, on remarque que le déplacement est équivalent à un effet de zoom. L'inconvénient de cette représentation, dans le cas où l'on associe un vecteur à chaque pixel, est que la lecture du résultat est rapidement difficile à cause du nombre total de vecteurs, en particulier si les mouvements sont complexes. Ainsi, la représentation de champs denses se fait à l'aide d'une carte de coloration où chaque vecteur est représenté par une couleur en fonction de son orientation et de sa norme. Elle permet une lecture plus claire du champ de vecteurs obtenu. On présente sur la figure 3.3 deux exemples de cartes pouvant être utilisées.

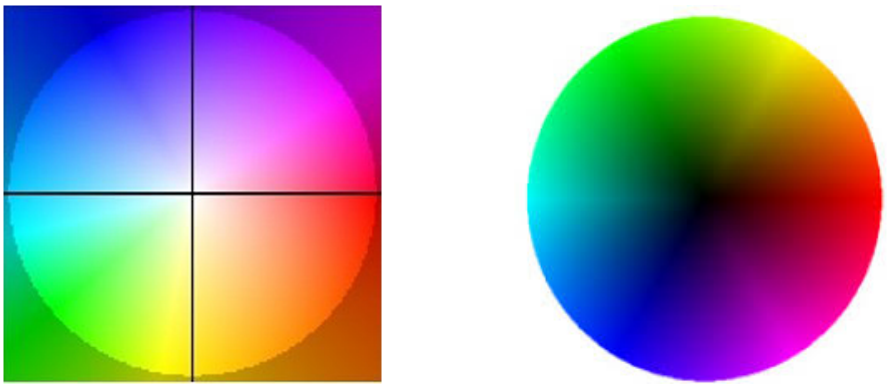


FIGURE 3.3 – Exemples de cartes de coloration utilisées pour la représentation d'un champ de vecteurs dense.

L'utilisation d'une carte ou d'une autre dans la littérature peut donc donner des résultats visuellement différents pour un même test. Dans notre cas, nous utiliserons la carte de

gauche. Ce choix est motivé par sa popularité à travers la communauté. C'est aussi celle proposée par le site *Middleburry* www.vision.middleburry.edu/flow/ qui nous servira de base de données pour les différents cas tests.

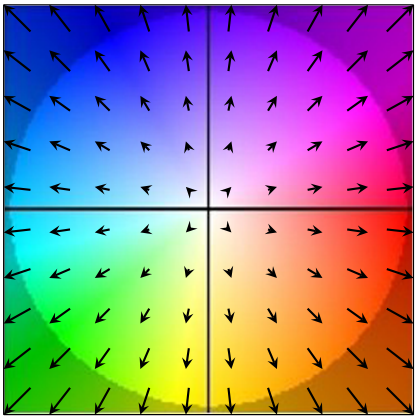


FIGURE 3.4 – Carte de coloration pour la représentation des champs de vecteurs donnée par le site de *Middleburry* avec le champ de vecteurs correspondant.

Dans le cas de la séquence de RubberWhale donnée par *Middleburry* (figure 3.5),

FIGURE 3.5 – Images 10 et 11 de la séquence de RubberWhale de *Middleburry*.

le flot optique exact représenté à l'aide de cette carte de coloration est donné sur la figure 3.6.

3.2.2 Résultats

3.2.2.1 Motivations de la prise en compte des variations de luminosité

Afin de justifier l'ajout de la prise en compte de la luminosité, nous proposons un test avec la séquence de RubberWhale 3.5 dans laquelle l'intensité lumineuse de la première

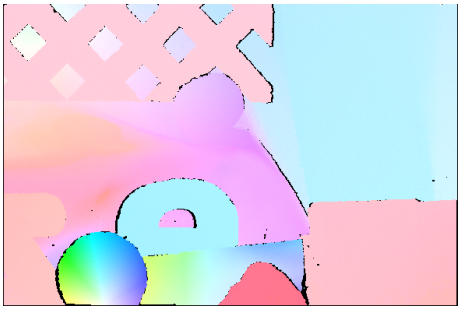


FIGURE 3.6 – Solution exacte du flot optique pour la séquence RubberWhale.

image a été modifiée (figure 3.7). Ce genre de situation peut arriver dans des séquences d'images réelles lorsque l'on passe d'une zone éclairée à une zone d'ombre.

FIGURE 3.7 – Séquence de RubberWhale avec augmentation de la luminosité sur la première image.

Comme le mouvement reste inchangé, le flot optique exact également. Nous montrons sur la figure 3.8 le résultat obtenu avec la stratégie de programmation considérant l'hypothèse de la constance de luminosité (3.2) puis avec celle de la luminosité variable. Pour cet exemple, nous n'effectuons aucune itération d'adaptation de la régularisation, le but étant de montrer l'effet de la prise en compte de la variation de luminosité.

Nous remarquons que dans le cas de séquences présentant une forte variation de luminosité, les résultats de la méthode classique (considérant la luminosité constante) peuvent être en grande partie erronés. Cette méthode est, en effet, incapable de faire la correspondance entre les pixels des deux images. La solution trouvée en prenant en compte la variation de la luminosité se révèle bien plus précise. Dans le but de pouvoir traiter des images en conditions réelles avec un code moins sensible aux variations de luminosité, nous garderons cette dernière pour la suite de nos tests.

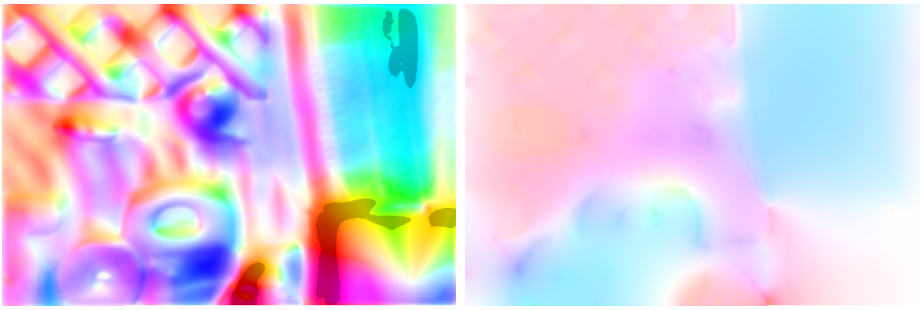
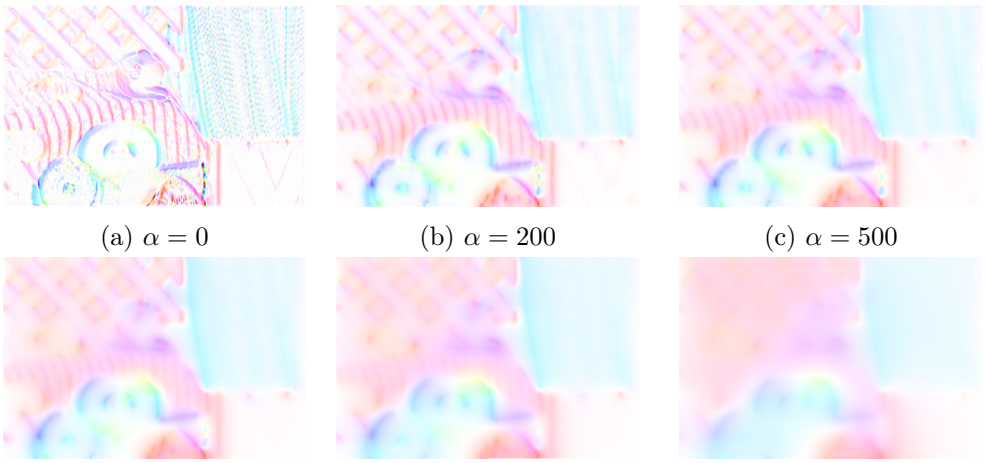


FIGURE 3.8 – Estimation du flot optique pour le test de RubberWhale avec variations de luminosité. A gauche, en utilisant l'hypothèse de la luminosité constante. A droite, en considérant une variation affine de la luminosité.

3.2.2.2 Influence du paramètre de régularisation α

Sur la figure 3.10, nous présentons l'effet de la régularisation pour différentes valeurs de α, lorsque ce paramètre est une constante. Nous pouvons voir qu'une régularisation trop faible ne lisse pas suffisamment le champ de vecteurs et fait apparaître des détails de textures non désirés. Cependant, si le paramètre α est trop grand, le champ est trop lissé et nous perdons les informations sur les bords des objets de l'image.



(a) $\alpha = 0$ (b) $\alpha = 200$ (c) $\alpha = 500$

(d) $\alpha = 1000$ (e) $\alpha = 2000$ (f) $\alpha = 10000$

FIGURE 3.9 – Estimation du flot optique pour différentes valeurs du paramètre α constant.

Lorsque ce choix est fixé de manière constante dans l'algorithme, il faut alors faire le meilleur compromis. Dans ce cas, nous remarquons que le meilleur candidat pour ce test est $\alpha = 2000$. Cependant, avec la stratégie LUM-REGU$_{\text{loc}}$, nous allons pouvoir le réévaluer localement à chaque itération en fonction de la géométrie des différents objets.

3.2.2.3 Applications sur différents cas tests

Afin de vérifier la validité de notre méthode, nous utilisons plusieurs cas tests couramment utilisés dans la littérature. Ces tests proviennent tous du site de l'université de Middleburry. Ils sont fournis avec les solutions exactes, nous permettant le calcul des erreurs. Dans le tableau 3.1, nous présentons brièvement quelles sont les particularités de chaque test.

	Cas test	Spécificité
RubberWhale		Textures importantes sur différents plans
Hydrangea		
Dimetrodon		
Grove2		Images synthétiques présentant de nombreux détails

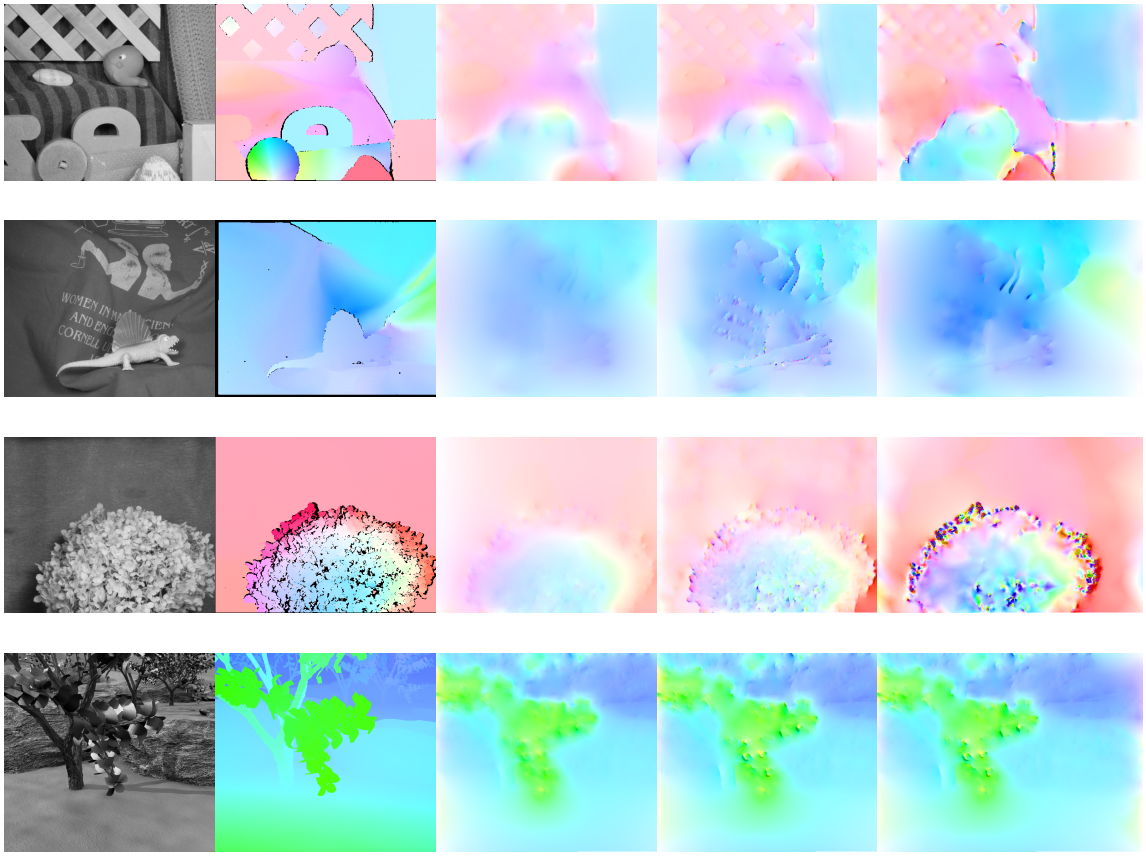
TABLE 3.1 – Particularités des différents cas tests de *Middleburry*.

Sur la figure 3.10, nous présentons les résultats de la méthode prenant en compte les variations de luminosité sans adaptation (c), puis avec l'adaptation du paramètre de régularité α : méthode LUM-REGU$_{\text{loc}}$. Nous donnons les résultat pour les formules d'adaptation du α (3.18) (d) et (3.16) (e).

Bien que la solution obtenue sans adaptation ne soit pas mauvaise, nous remarquons que le contrôle adaptatif du paramètre α permet d'améliorer l'estimation du flot optique sur les zones discontinues. Pour déterminer la précision de la solution estimée, nous utilisons deux méthodes d'évaluation proposées dans la littérature. Dans un premier temps, nous calculons l'erreur angulaire moyenne (*Average Angular Error*, notée AAE). Ce calcul d'erreur permettant la mesure de la précision d'un champ de vecteurs, a été proposé par Fleet et Jepsen [47]. Il se fait à l'aide de la formule

$$AAE = arccos \frac{u_{1,h}u_{1,e} + u_{2,h}u_{2,e} + 1}{\sqrt{(u_{1,h}^2 + u_{2,h}^2 + 1)(u_{1,e}^2 + u_{2,e}^2 + 1)}} \tag{3.19}$$

où $\mathbf{u}_h = (u_{1,h}, u_{2,h})$ est une approximation du champ de vecteurs obtenue après calculs et $\mathbf{u}_e = (u_{1,e}, u_{2,e})$ représente le flot optique exact.



(a) Image test (b) Solution exacte (c) Estimation sans adaptation (d) Méthode LUM-REGU$_{\text{loc}}$, adaptation (3.18) (e) Méthode LUM-REGU$_{\text{loc}}$, adaptation (3.16)

FIGURE 3.10 – Estimation du flot optique obtenue avec et sans adaptation de α pour différents cas tests du site de *Middleburry*.

Cette méthode d'évaluation présente l'avantage de pénaliser les vecteurs de grandes amplitudes face à ceux de petites amplitudes. En revanche, comme le soulignent Otte et Nagel [98], un même angle d'erreur présent symétriquement peut donner des erreurs angulaires différentes (figure 3.11).

FIGURE 3.11 – Différence d'erreur angulaire pour deux vecteurs de même angle mais avec des normes différentes.

Malgré cet inconvénient, nous utilisons cette évaluation car nous souhaitons principalement vérifier que le champ de vecteurs obtenu soit bien orienté. C'est actuellement la plus utilisée pour les estimations de flots optiques. Le calcul de cette erreur permettra ainsi de comparer nos résultats à ceux de la littérature.

La deuxième méthode que nous utilisons permet d'évaluer le champ de vecteurs de manière plus précise à l'aide de la norme L^2. Cette erreur est également appelée *Endpoint Error* (notée EE). Elle ne permet pas d'obtenir une evaluation concernant la direction du champ de vecteurs mais elle n'est pas affectée par le problème cité ci-dessus. Elle est déterminée à l'aide de la formule

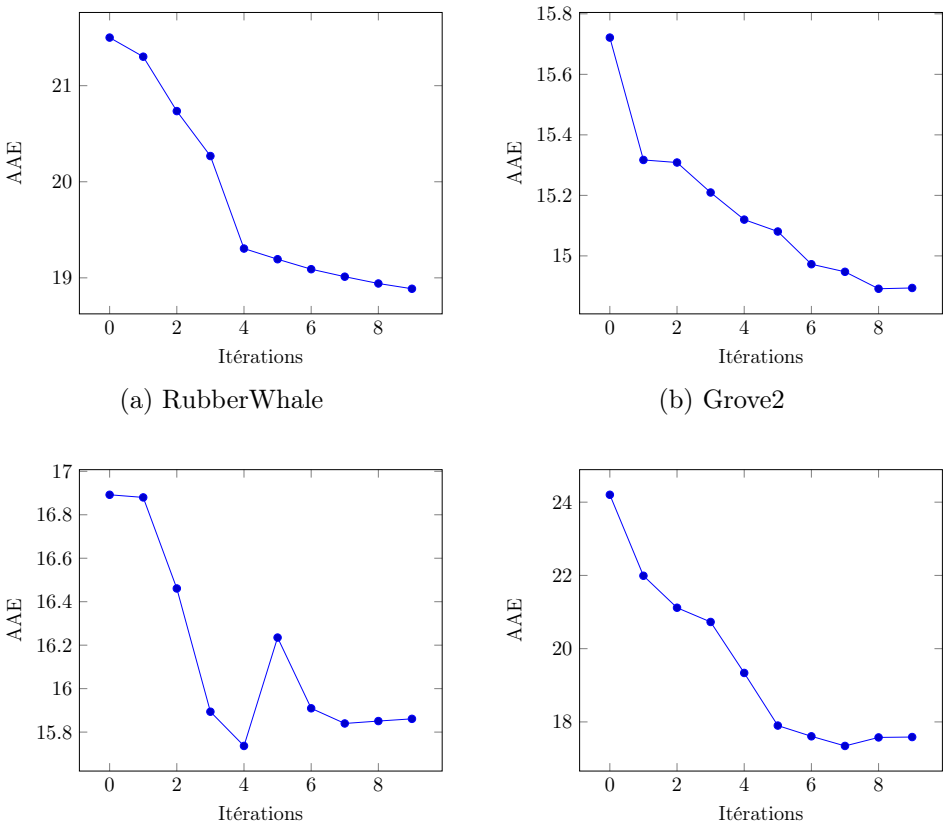
$$EE = \sqrt{(u_{1,h} - u_{1,e})^2 + (u_{2,h} - u_{2,e})^2}. \tag{3.20}$$

Afin d'observer l'évolution de la précision de l'estimation au cours de l'adaptation du paramètre de régularisation α, nous présentons sur les figures 3.12 et 3.13 l'évolution de l'erreur angulaire moyenne (AAE) et l'erreur moyenne L^2 (EE).

Notre but étant d'obtenir un champ de vecteurs lisse sur les zones homogènes tout en conservant les discontinuités du flot optique, nous débutons l'algorithme avec une valeur de α relativement grande. Ce choix a pour effet de fortement lisser le champ. Nous remarquons qu'au cours des itérations d'adaptation, les erreurs AAE et EE diminuent. Les zones discontinues étant minoritaires, cette diminution est limitée. Malgré cela, nous remarquons visuellement que la solution est plus proche de la solution exacte et que les bords des éléments de l'image sont mieux définis.

Lors des itérations pour le contrôle local de la régularisation, nous effectuons également une adaptation du maillage. Sur la figure 3.14, nous donnons les maillages adaptés obtenus. Le maillage initial est fin, il comporte deux cellules par pixel. Les régions où le paramètre α est grand sont des régions qui ne présentent pas de bord et où le champ sera lissé. Avec l'adaptation du maillage, nous pouvons donc réduire le nombre de mailles dans ces zones et l'augmenter dans les zones où la valeur de η_K est grande, c'est à dire au niveau des contours des objets.

Grâce à cette adaptation, le nombre de degrés de liberté au cours des itérations réduit de manière significative dès la deuxième itération puis garde une valeur stable (voir figure 3.15). Il n'est donc pas nécessaire de faire de nombreuses itérations d'adaptation du maillage.La diminution du nombre de degrés de liberté implique également un gain de temps de calcul. Pour s'en convaincre, dans le tableau 3.2, nous montrons l'influence de l'adaptation du maillage sur le temps de calcul en comparant les temps d'exécution de l'algorithme pour quatre itérations d'adaptation de la régularisation, avec et sans adaptation de maillage. Sur 4 itérations, le gain de temps est de plus de 50%. Il est aussi important de noter que la première itération effectuée sur le maillage initial représente la



(a) RubberWhale (b) Grove2

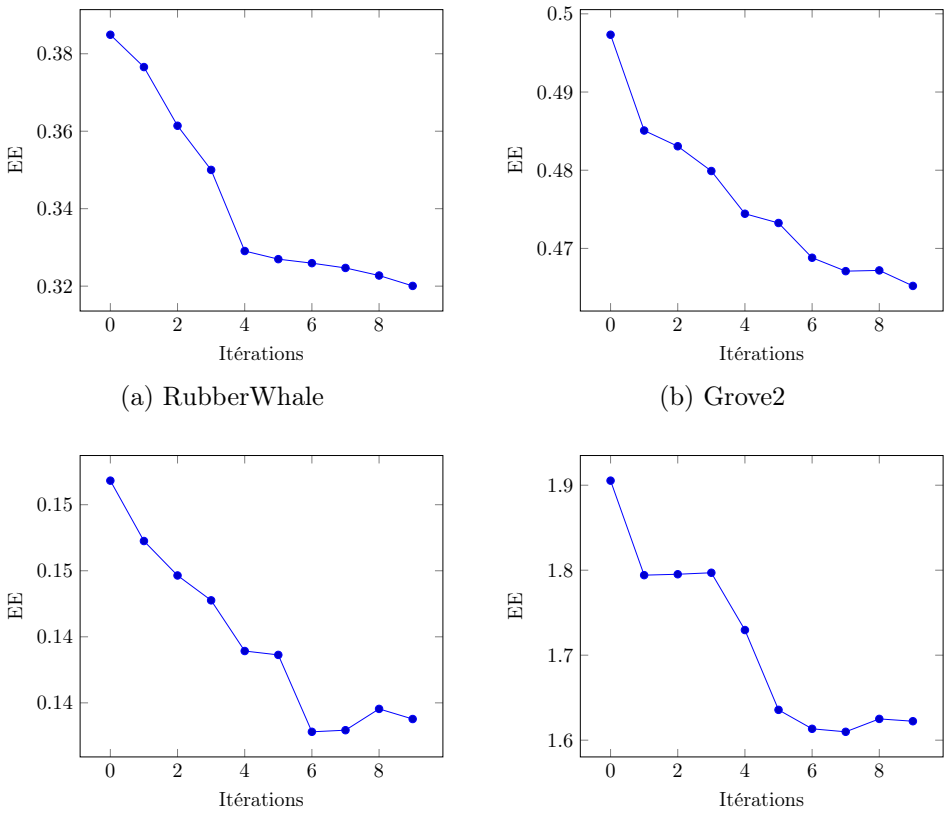
(c) Dimetrodon (d) Hydrangea

FIGURE 3.12 – Evolution de l'erreur angulaire de LUM-REGU$_{\text{loc}}$ en fonction des itérations d'adaptation pour les différents cas tests.

majorité du temps de calcul. Nous présentons ce temps entre parenthèses afin de pouvoir comparer. Ainsi, le temps ajouté pour chaque itération supplémentaire est de moins en moins important face au temps de la première itération.

	Avec adaptation du maillage	Sans adaptation du maillage
RubberWhale	2min 48s (1min 18s)	5min 45s
Dimetrodon	2min 11s (1min 17s)	5min 43s
Hydrangea	2min 45s (1min 15s)	5min 33s

TABLE 3.2 – Temps d'exécution de l'algorithme avec et sans adaptation du maillage pour quatre itérations d'adaptation du paramètre de régularisation. Entre parenthèses, nous donnons le temps de la premiere itération.



(a) RubberWhale (b) Grove2

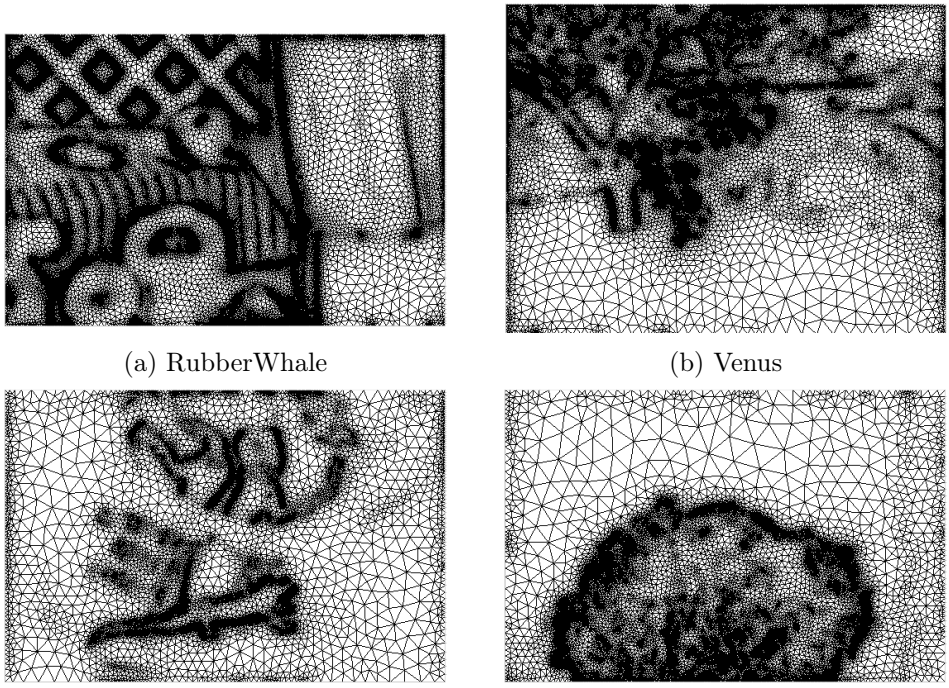
(c) Dimetrodon (d) Hydrangea

FIGURE 3.13 – Evolution de l'erreur L^2 de LUM-REGU$_{\text{loc}}$ en fonction des itérations d'adaptation pour les différents cas tests.

3.3 Optimisation des résultats par rapprochement itératif

Dans la section 1.4, nous avons présenté une méthode permettant d'optimiser l'estimation du flot optique à l'aide d'un procédé de rapprochement itératif et d'une pyramide gaussienne. Contrairement au contrôle adaptatif de la régularisation, cet algorithme concerne l'amélioration du terme correspondant à l'adéquation aux données du problème. Nous allons à présent appliquer cette méthode au modèle LUM-REGU$_{\text{loc}}$.

Soient f_1 et f_2 les deux images de la séquence. Nous commençons par appliquer un flou gaussien de paramètre ρ sur ces deux images et déterminons le flot optique $\mathbf{U}$ entre $K_\rho \star f_1$ et $K_\rho \star f_2$. L'approximation $\mathbf{U}$ ainsi obtenue est ensuite utilisée afin de déterminer la nouvelle image $f_1' = f_1(\mathbf{x} + \mathbf{U})$ à l'aide de l'interpolation bilinéaire



(a) RubberWhale (b) Venus

(c) Dimetrodon (d) Hydrangea

FIGURE 3.14 – Maillages adaptés donnés par LUM-REGU$_{\text{loc}}$ pour les différents cas tests de *Middleburry*.

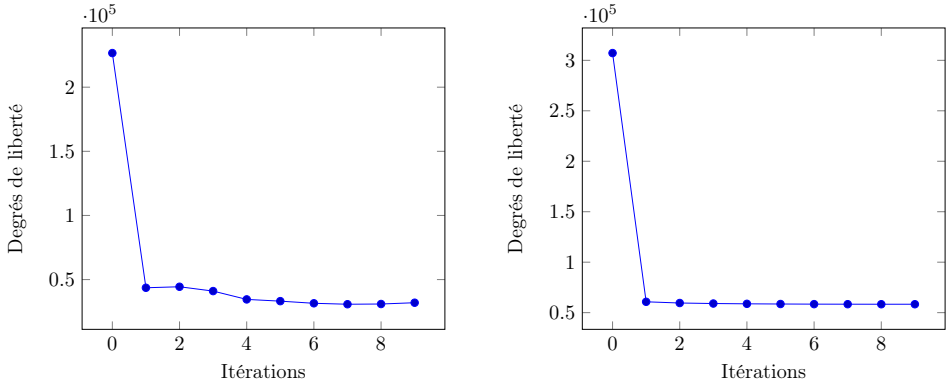
$$f_1'(x,y) \approx \frac{1}{(x_2-x_1)(y_2-y_1)}(f_1(x_1,y_1)(x_2-x)(y_2-y) + f_1(x_2,y_1)(x-x_1)(y_2-y) + f_1(x_1,y_2)(x_2-x)(y-y_1) + f_1(x_2,y_2)(x-x_1)(y-y_1))$$

où

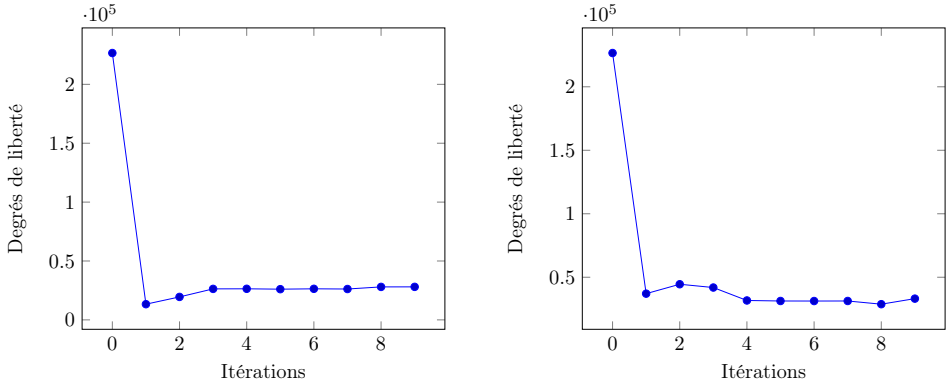
$$\begin{aligned} x_1 &= \lfloor x-u_1 \rfloor \\ x_2 &= \lceil x-u_2 \rceil \\ y_1 &= \lfloor y-u_1 \rfloor \\ y_2 &= \lceil y-u_2 \rceil. \end{aligned}$$

Après avoir réduit la valeur du paramètre ρ, nous réitérons le processus avec f_1' et f_2.

Le flot optique final est alors donné par la somme de tous les $\mathbf{U}^n$. La méthode peut être résumée à l'aide du pseudo-code suivant.



(a) RubberWhale (b) Grove2



(c) Dimetrodon (d) Hydrangea

FIGURE 3.15 – Evolution du nombre de degrés de liberté au cours des itérations de LUM-REGU$_{\text{loc}}$ pour les différents cas tests.

Initialisation : $f_1^0 = f_1$ et $\mathbf{U}^0 = 0$.
Pour n allant de 0 à N
- Déterminer le flot optique $\mathbf{U}^{n+1}$ entre les images $K_\rho \star f_1^n$ et $K_\rho \star f_2$.
- Déterminer $f_1^{n+1}(\mathbf{x}) = f_1^n(\mathbf{x} + \mathbf{U}^{n+1})$.
- Diminuer le paramètre ρ.

Fin pour

Nous combinons cette méthode avec la stratégie adaptative du paramètre de régularisation et du maillage. Les deux adaptations se font alors au même moment que l'étape d'interpolation bilinéaire. Le pseudo-code de l'algorithme complet s'écrit alors

Dans la suite, nous nommerons OF-Rapp cette méthode incluant le rapprochement itératif ainsi que la pyramide gaussienne. Nous donnons les résultats obtenus avec cette

Initialisation : $f_1^0 = f_1$, $\mathbf{U}^0 = 0$, et $\mathcal{T}_h^0$ un maillage cartésien régulier.
Pour n allant de 0 à N
- Déterminer le flot optique $\mathbf{U}^{n+1}$ entre les images $K_\rho \star f_1^n$ et $K_\rho \star f_2$ sur le maillage $\mathcal{T}_h^n$.
- Construire le maillage adapté $\mathcal{T}_h^{n+1}$.
- Choisir localement $\alpha^{n+1}(x, y)$ sur $\mathcal{T}_h^{n+1}$.
- Déterminer $f_1^{n+1}(\mathbf{x}) = f_1^n(\mathbf{x} + \mathbf{U}^{n+1})$.
- Diminuer le paramètre ρ.

Fin pour

méthode sur la figure 3.16 et les erreurs en fonction des itérations d'adaptation pour l'erreur angulaire (figure 3.17) et l'erreur L^2 (figure 3.18). En comparant ces deux dernières figures aux figures 3.12 et 3.13, nous constatons que les pics d'erreurs, visibles notamment sur le cas test *Dimetrodon*, sont stabilisés.

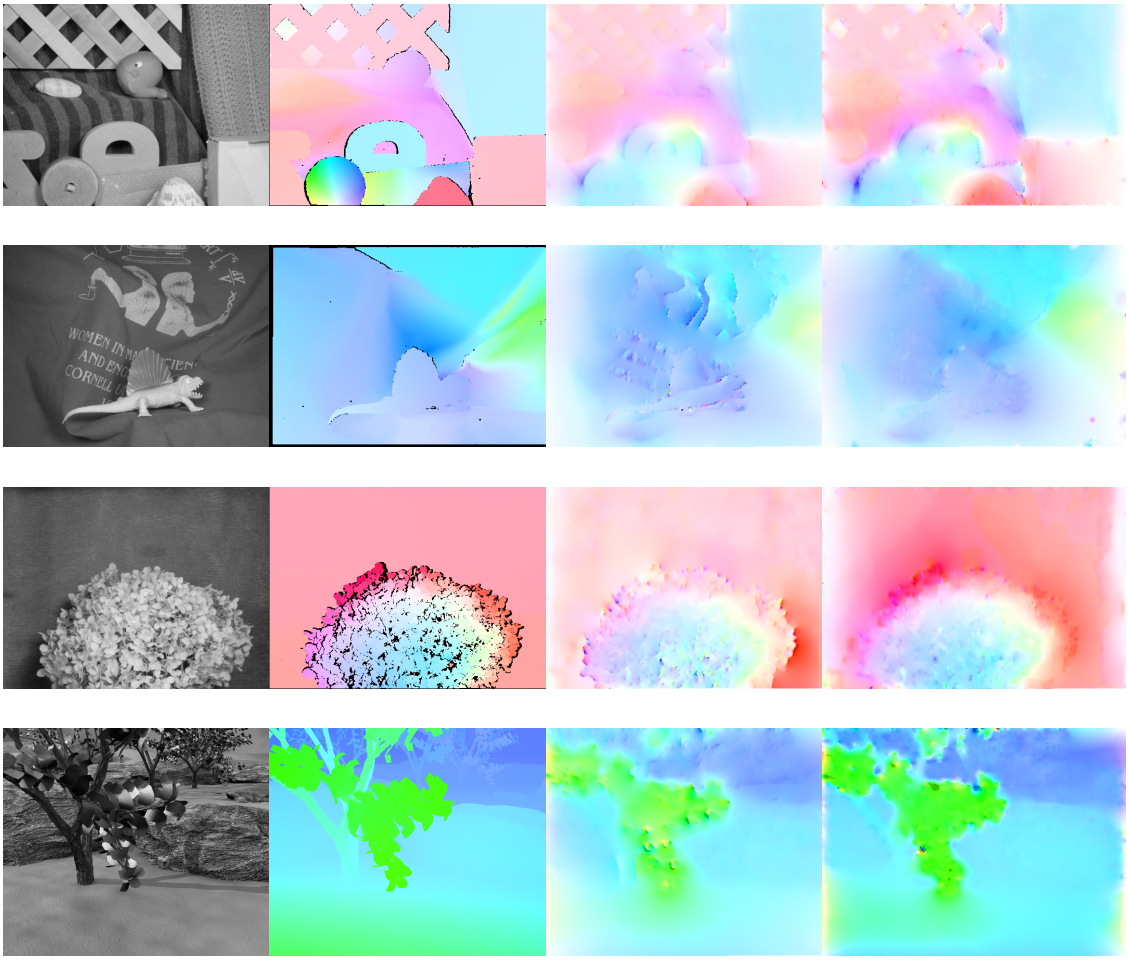
Pour tous les cas testés, nous observons une diminution des erreurs angulaires et L^2. Nous remarquons visuellement les améliorations sur la figure 3.16. Le calcul avec la stratégie OF-Rapp propose effectivement des résultats plus proches des solutions exactes.

Dans le tableau 3.3, nous comparons les temps que nous obtenons avec notre méthode de régularisation locale, sans puis avec la méthode de rapprochement itératif. Nous comparons donc LUM-REGU$_{\text{loc}}$ et OF-Rapp. Dans les deux cas, nous faisons quatre itérations d'adaptation.

	Sans rapprochement itératif LUM-REGU$_{\text{loc}}$	Avec rapprochement itératif OF-Rapp
RubberWhale	2min 48s	3min 31s
Dimetrodon	2min 11s	3min 01s
Hydrangea	2min 45s	3min 18s

TABLE 3.3 – Temps d'exécution de l'algorithme avec et sans rapprochement itératif pour quatre itérations d'adaptation du paramètre de régularisation et quatre itérations de rafinement.

Combinée à l'adaptation de la régularisation, cette méthode permet donc de gagner encore un niveau dans la précision des résultats. Cependant, malgré le fait que nous ayons pu amortir le coût des itérations ajoutées en les combinants à celles de l'adaptation, le tableau 3.3 nous montre que le programme reste plus lourd en termes de temps de calcul. En effet, il faut ajouter au temps de la résolution du problème, le temps de l'interpolation bilinéaire ainsi que celui nécessaire pour appliquer un nouveau flou gaussien à chaque itération.

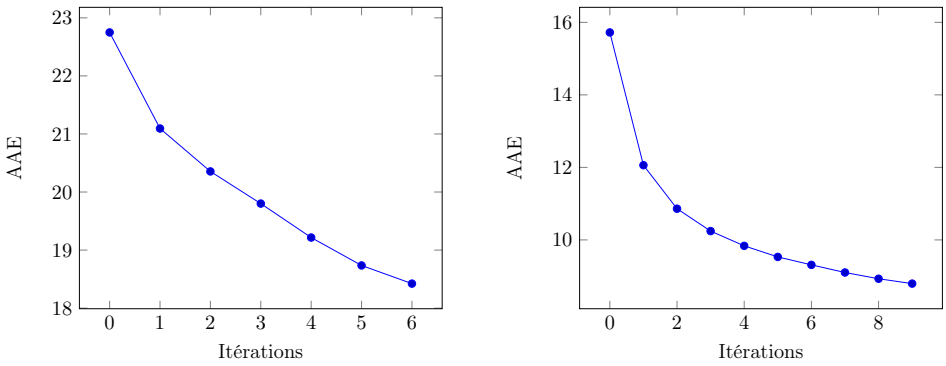


(a) Image test (b) Solution exacte (c) Sans rapprochement itératif ni filtre gaussien : LUM-REGU$_{\text{loc}}$ (d) Avec rapprochement itératif et filtre gaussien : OF-Rapp

FIGURE 3.16 – Estimation du flot optique obtenue avec et sans raffinement itératif et pyramide gaussienne pour les différents cas tests avec adaptation du paramètre de régularisation et prise en compte de la variation de la luminosité.

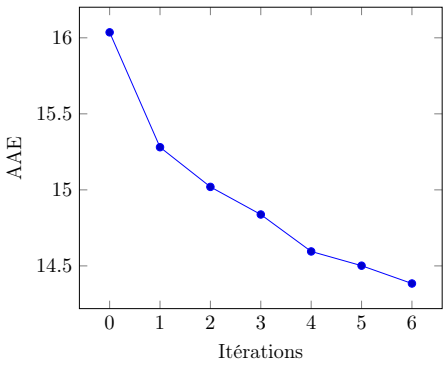
3.4 Contrôle adaptatif du paramètre λ

Comme nous l'avons vu dans dans ce chapitre, la méthode LUM-REGU$_{\text{loc}}$ permet d'obtenir une bonne approximation du champ de vecteurs même lorsque la luminosité varie dans la séquence. Toutefois, dans l'exemple présenté, la variation de luminosité était relativement uniforme. Nous présentons dans cette partie un nouvel exemple dans lequel la variation de luminosité est localisée (voir figure 3.19).

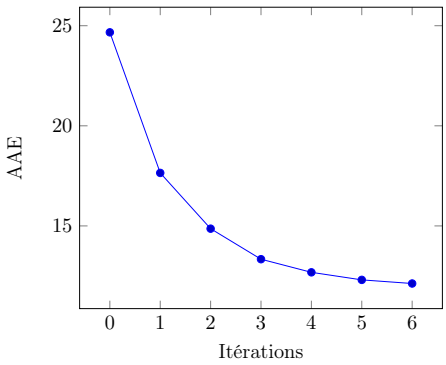


(a) RubberWhale

(b) Grove2



(c) Dimetrodon

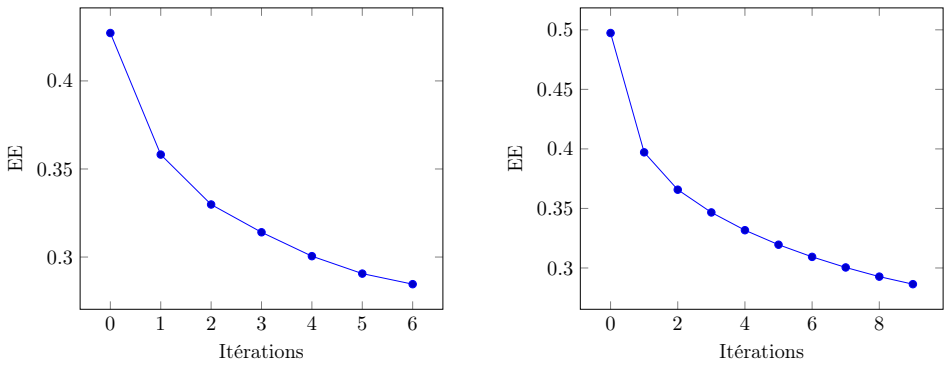


(d) Hydrangea

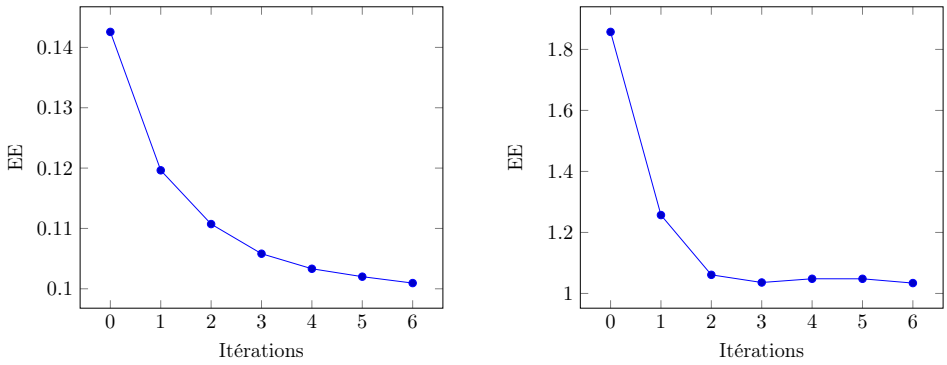
FIGURE 3.17 – Evolution de l'erreur angulaire de OF-Rapp en fonction des itérations d'adaptation pour les différents cas tests.

Sur la figure 3.20, nous présentons les résultats obtenus à l'aide de la méthode LUM-REGU$_{\text{loc}}$ pour trois valeurs constantes du paramètre λ : 10α, 1000α et 10000α. Dans le cas d'une valeur trop faible comme 10α, les zones où la luminosité a subi des variations sont bien traitées mais le champ de vecteurs est fortement lissé, ce qui implique des résultats peu précis au niveau des discontinuités. Les valeurs 1000α et 10000α ne permettent pas d'obtenir une bonne approximation des zones à luminosité variable.

Afin de conserver la qualité de l'approximation dans les zones où la luminosité est constante tout en corrigeant le champ de vecteurs dans les zones où la luminosité est variable, nous choisissons d'utiliser une méthode de contrôle local du paramètre λ basée sur le même principe que l'adaptation du paramètre α. La valeur de m_t du flot optique $(u_1, u_2, m_t)^T$ nous indique l'importance de la variation de luminosité. Lorsque cette valeur est grande, cela signifie que la variation de luminosité est grande. Elle nous permet ainsi, après un premier calcul, de déterminer deux types de zones : celles où la luminosité varie et celles où elle ne varie pas. Pour améliorer l'estimation du flot optique, nous vou-



(a) RubberWhale (b) Grove2



(c) Dimetrodon (d) Hydrangea

FIGURE 3.18 – Evolution de l'erreur L^2 de OF-Rapp en fonction des itérations d'adaptation pour les différents cas tests.

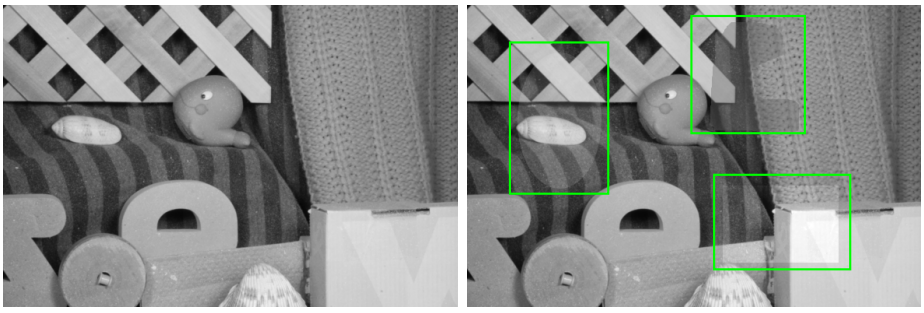


FIGURE 3.19 – Séquence de RubberWhale avec une variation de luminosité non régulière.

lons alors adapter le paramètre de régularisation λ afin de réduire l'influence de m_t dans les zones où il n'y a pas de variation de la luminosité en faisant tendre le système (3.5) vers le système (3.2) mais également de garder le poids de l'hypothèse (3.3) sur les zones où la luminosité varie. Pour cela, nous considérons le nouvel indicateur d'erreur $\eta_K(m_t)$

(a) $\lambda = 10\alpha$

(b) $\lambda = 1000\alpha$

(c) $\lambda = 10000\alpha$

FIGURE 3.20 – Estimation du flot optique pour une variation non uniforme de la luminosité à l'aide de la méthode REGU_{loc} (a) et de la méthode $\text{LUM-REGU}_{\text{loc}}$ pour différentes valeurs de λ constant (b, c et d).

défini pour chaque élément $K \in \mathcal{T}_h$ par

$$\eta_K(m_t) = \lambda_K^{-\frac{1}{2}} h_K \left\| \nabla m_t \right\|_{L^2(K)} \tag{3.21}$$

et nous utilisons une formule équivalente à celle utilisée pour l'adaptation du paramètre α

$$\lambda_K^{n+1} = \max\left(\frac{\lambda_K^n}{1 + \kappa \max\left(\frac{\eta_K(m_t)}{\|\eta(m_t)\|_\infty} - \frac{\zeta}{100}, 0 \right)}, \lambda_s \right) \tag{3.22}$$

Ce choix implique une réduction lente et régulière de la régularisation λ dans les zones où le terme $\eta_K(m_t)$ est grand. Afin de ne pas obtenir un champ discontinu, nous appliquons également une légère diffusion sur λ.

Sur la figure 3.21, nous présentons la fonction $\lambda(x, y)$ après adaptation. Les parties jaunes, correspondantes aux zones où la lumière a varié, représentent là où λ a été réduit. Les parties roses sont celles où λ n'a pas été modifié et a gardé une grande valeur. Sur la même figure, nous présentons l'estimation du flot optique obtenue avec l'ajout de cette adaptation.

Grâce à cette nouvelle adaptation, nous obtenons une bonne estimation du flot optique

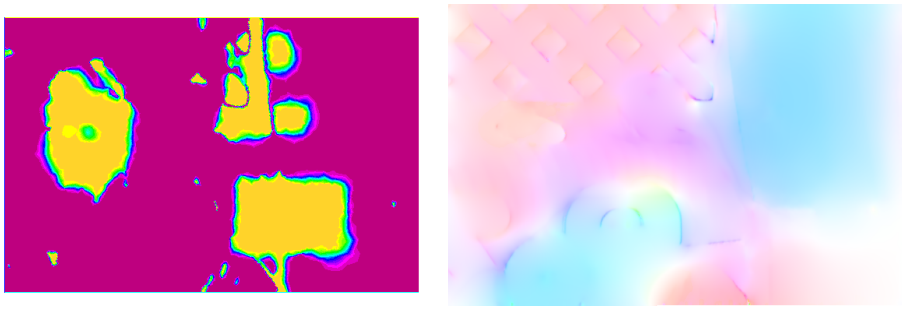


FIGURE 3.21 – Gauche : fonction $\lambda(x, y)$ après adaptation locale. Droite : estimation de mouvement obtenue avec l'adaptation locale du paramètre λ.

malgré les variations locales de luminosité perturbant l'assimilation des pixels correspondants d'une image à l'autre.

3.5 Conclusion

Dans ce chapitre, nous avons développé des modèles variationnels utilisant la méthode des éléments finis pour la résolution du problème de l'estimation du flot optique en petits déplacements. Nous avons d'abord présenté un modèle dans lequel la luminosité est supposée constante d'une image à l'autre de la séquence. Cette méthode nous a permis d'obtenir de bonnes estimations sur les cas tests du site de Middleburry. Cependant, lorsque nous avons fait varier la luminosité, nous avons vu que ce modèle perd de son efficacité. Pour dépasser ce problème, nous avons donc créé un deuxième modèle prenant en compte la variation de la luminosité. Les résultats numériques ont montré que le modèle parvient alors à garantir la précision du résultat, même lorsque l'intensité lumineuse varie dans la séquence. Pour améliorer l'estimation de la solution sur les arêtes de l'image, nous avons implémenté une méthode de contrôle adaptatif de la régularisation. Nous avons montré que l'algorithme adaptatif consiste à construire une suite (U_n, α_n) qui minimise une famille d'énergies discrètes qui Γ- converge vers la fonctionnelle de Mumford-Shah. Autrement dit, $(U_n)_n$ converge vers U, minimiseur de Mumford-Shah et (α_n) converge vers une fonction $\alpha = 1$ sauf sur l'ensemble S_u où elle vaut 0. Cette méthode nous a permis de gagner en précision au niveau des contours des objets et ainsi de diminuer l'erreur angulaire. Enfin, nous avons proposé un contrôle adaptatif pour le terme concernant la luminosité. Cela a, en effet, amélioré l'estimation du flot optique dans le cas où la variation lumineuse serait localisée.

Chapitre 4

Méthodes parallèles pour l'estimation du flot optique

Sommaire

4.1 Méthode pararéelle

Comme nous l'avons vu dans le chapitre précédent, l'adaptation du paramètre α permet d'obtenir une caractérisation plus fine de la géométrie de l'image. Cela permet d'avoir

une estimation du flot optique plus précise et de réduire fortement le nombre de degrés de liberté. Nous voulons à présent améliorer le temps de calcul en parallélisant notre programme. Parce que nous sommes dépendants de la géométrie et que la parallélisation sur carte graphique n'est pas la plus adaptée pour l'adaptation du maillage, nous avons choisi dans un premier temps d'implémenter un algorithme parallèle en temps. Dans le chapitre 1, nous avons introduit cette méthode à l'aide d'un exemple simple. Dans ce chapitre, nous reprenons cette méthode et l'appliquons dans le cas du flot optique.

4.1.1 Application de la méthode pararéelle pour le problème du flot optique

Dans le but d'utiliser le pararéel, nous avons dans un premier temps ajouté une variation en temps au problème. Ce travail est préparatif à la résolution du problème du flot optique en grands déplacements (voir chapitre 5). Le problème à résoudre devient alors

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} - \operatorname{div}(\alpha \nabla \mathbf{u}) + J_\rho \mathbf{u} = \mathbf{f} \text{ dans } \Omega, \quad t \in [0,T] & (4.1) \\ \mathbf{u}(0) = 0. & (4.2) \end{cases}$$

Pour une méthode sans contrôle local du paramètre de régularisation, l'algorithme utilisé est rigoureusement identique à la méthode proposée dans l'exemple (2.2). Nous commençons par définir la subdivision $0 = T_0 < T_1 < \cdots < T_N = T$ de l'intervalle de temps $[0,T]$ puis nous définissons le solveur grossier pour $n = 0, \ldots, N-1$,

$$\begin{cases} \mathbf{u}^g_{n+1} = \mathcal{G}(\mathbf{u}^g_n) = \mathbf{u}^g_n + \Delta T \left(\operatorname{div}(\alpha \nabla \mathbf{u}^g_{n+1}) - J_\rho \mathbf{u}^g_{n+1} + \mathbf{f}\right) \\ \mathbf{u}^g_0 = \mathbf{u}^g(T_0 = 0) = 0. \end{cases}$$

Chacune des solutions obtenues est ensuite utilisée comme état initial de la résolution fine sur l'intervalle correspondant. Ainsi, en parallèle pour chaque CPU_i, le problème à résoudre sur le sous intervalle $[T_k, T_{k+1}]$ est donné par

$$\begin{cases} \mathbf{u}^f_{n,m+1} = \mathcal{F}(\mathbf{u}^f_{n,m}) = \mathbf{u}^f_{n,m} + \Delta t \left(\operatorname{div}(\alpha \nabla \mathbf{u}^f_{n,m+1}) - J_\rho \mathbf{u}^f_{n,m+1} + \mathbf{f}\right) \\ \mathbf{u}^f_{n,0} = \mathbf{u}^g_n \end{cases}$$

où $m = 0, \ldots, M-1$.

Une fois les deux solveurs définis, la phase de mise à jour de la solution à l'aide de la prédiction-correction est identique à celle présentée dans le chapitre 2.

Pour appliquer le contrôle adaptatif de la régularisation ainsi que l'adaptation du maillage, soit nous effectuons l'adaptation lors des itérations grossières à l'initialisation, soit nous le faisons à chaque itération fine. Avoir un paramètre de régularisation différent entre deux CPU n'est pas gênant, nous n'avons donc pas la nécessité de faire le transfert de ces valeurs entre les processeurs. Toutefois, il serait très couteux d'effectuer le contrôle du paramètre à chaque itération fine, de plus il peut ne pas être bon de faire trop d'étapes d'adaptation du paramètre de régularisation. Pour ces raisons, nous effectuons donc ces contrôles lors des étapes de résolutions grossières.

4.1.2 Résultats numériques

Dans un premier temps, nous allons vérifier que les résultats obtenus à l'aide de la méthode pararéelle correspondent à ceux obtenus avec le même algorithme en séquentiel. Nous déterminons donc à chaque itération l'erreur angulaire entre les deux solutions et nous présentons les résultats obtenus à chaque itération globale sur la figure 4.1.

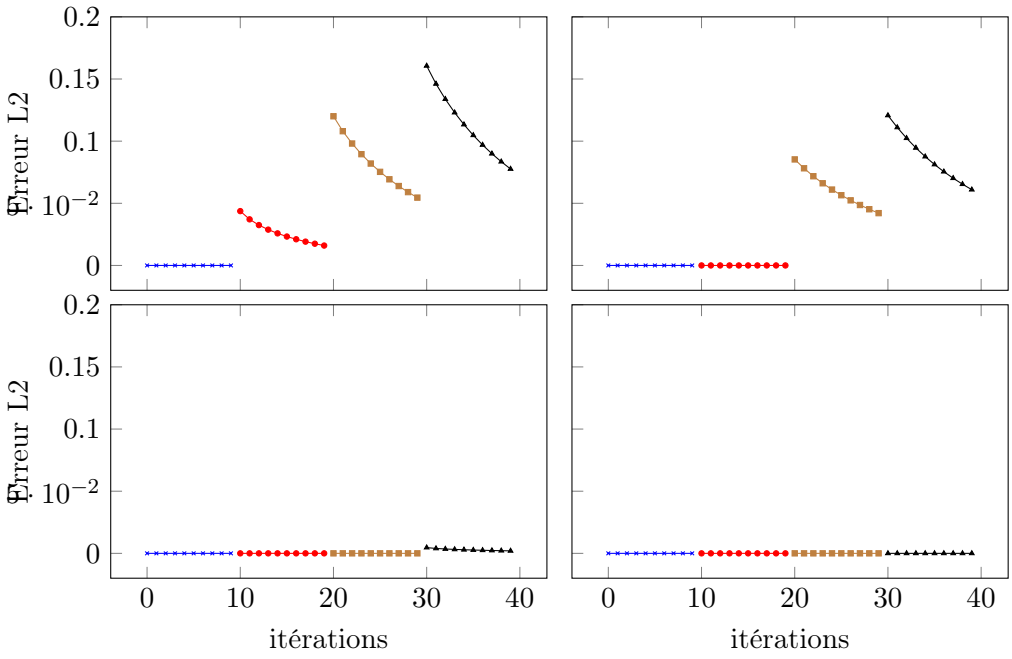


FIGURE 4.1 – Evolution de l'erreur L2 entre la méthode pararéelle et de celle séquentielle à chaque itération globale.

Comme dans l'exemple (2.2), nous constatons que l'étape de mise à jour entre les itérations globales a pour effet de réduire de manière significative l'écart entre l'estimation donnée par la méthode séquentielle et celle donnée par la méthode pararéelle et donc d'accélérer la convergence de cette dernière. Nous présentons sur la figure 4.2 les solutions

obtenues avec la méthode pararéelle, avec et sans adaptation.

FIGURE 4.2 – Gauche : Solution du code pararéel sans adaptation. Droite : Solution du code pararéel avec adaptation.

Comme dans le chapitre précédent, l'adaptation du paramètre α préserve la qualité de la solution générale et améliore la précision de l'estimation sur les arêtes de l'image.

Concernant le temps d'exécution de l'algorithme, nous comparons les temps obtenus avec la méthode pararéelle et avec celle séquentielle. Pour la méthode pararéelle, nous testons plusieurs configurations permettant d'obtenir un nombre d'itérations équivalent à la méthode séquentielle afin de déterminer laquelle propose une meilleure accélération. Cette accélération est donnée par le rapport du temps séquentiel sur le temps parallèle. De plus, nous pouvons constater sur la figure 4.1 que l'étape de mise a jour permet d'atteindre un résultat proche de celui donné par l'algorithme séquentiel sans avoir à effectuer toutes les itérations globales. Nous prenons comme base le programme séquentiel avec 40 itérations. Afin de garder la cohérence de la comparaison avec le programme séquentiel, les différentes configurations pararéelles choisies pour ces tests sont 10 itérations grossières et 4 fines, 5 itérations grossières et 8 fines et enfin 2 itérations grossières et 20 fines. Les résultats sont donnés sur la figure 4.3.

Comme nous pouvons le voir, l'algorithme est non-scalable. Ce qui veut dire que nous ne gagnons pas autant de temps que le nombre de processeurs utilisés. Sur le même graphique nous présentons également le détail des temps d'exécutions des principales parties de l'algorithme : la résolution grossière, la résolution fine, l'étape de mise à jour et les communications. D'après ces résultats, les temps de communications sont négligeables dans cet algorithme. La non-scalabilité réside dans l'étape de mise à jour qui nécessite la résolution de problèmes grossiers supplémentaires, proportionnellement au nombre de processeurs utilisés. Sur la figure 4.3 où nous donnons les temps d'exécution des principales parties de l'algorithme pararéel, nous pouvons constater que l'étape de mise à jour est une étape très coûteuse lorsqu'il y a un grand nombre de processeurs et donc un grand nombre d'itérations grossières. Or, ces étapes sont nécessaires à l'algorithme et permettent de converger vers l'algorithme séquentiel de manière plus rapide.

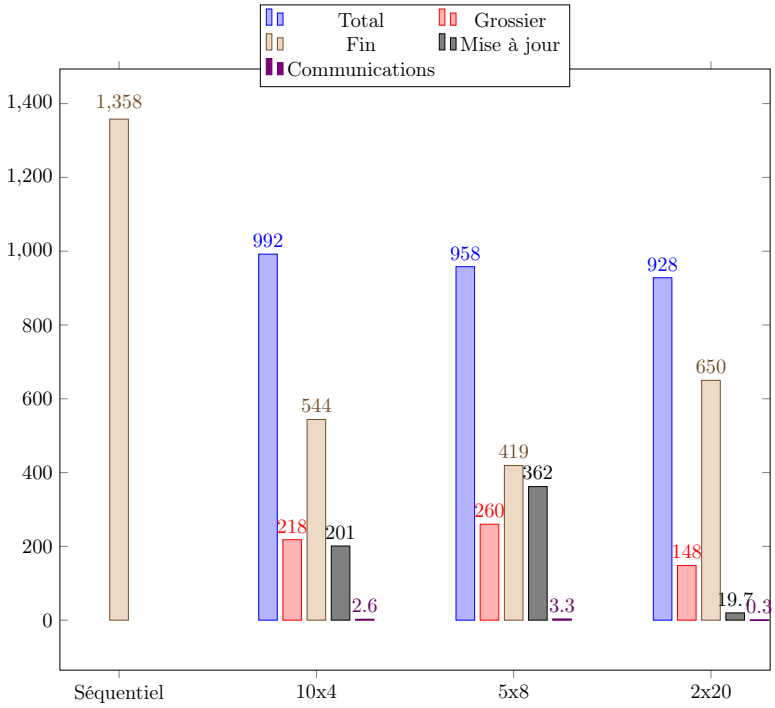


FIGURE 4.3 – Temps de calcul détaillés des principales tâches de l'implémentation pour différentes configurations de l'algorithme pararéel : 10 itérations grossières et 4 fines, 5 grossières et 8 fines, 2 grossières et 20 fines.

4.1.3 Vers une amélioration de la méthode pararéelle

Afin de réduire les temps de calcul de la méthode pararéelle, un algorithme proposant une modification de l'ordre des opérations est présenté dans [7]. Il consiste à ne pas effectuer l'ensemble de la résolution grossière en amont. Nous présentons dans cette partie les différences entre les deux algorithmes.

Sur la figure 4.4, nous schématisons l'organisation des différentes étapes de la méthode pararéelle classique présentée dans le chapitre 2.

Après une première étape pendant laquelle tous les processeurs effectuent les itérations grossières, débute l'étape des itérations globales. Chaque itération globale comprend une résolution fine partant de l'état initial $\mathbf{u}_i^g$ pour le processeur i, puis la mise à jour des états initiaux suivants. Dans cet algorithme, les processeurs effectuent les mêmes étapes en même temps.

Les étapes de la résolution grossière et de la mise à jour étant effectuées par tous les processeurs, les communications nécessaires sont minimales puisque seuls les résultats

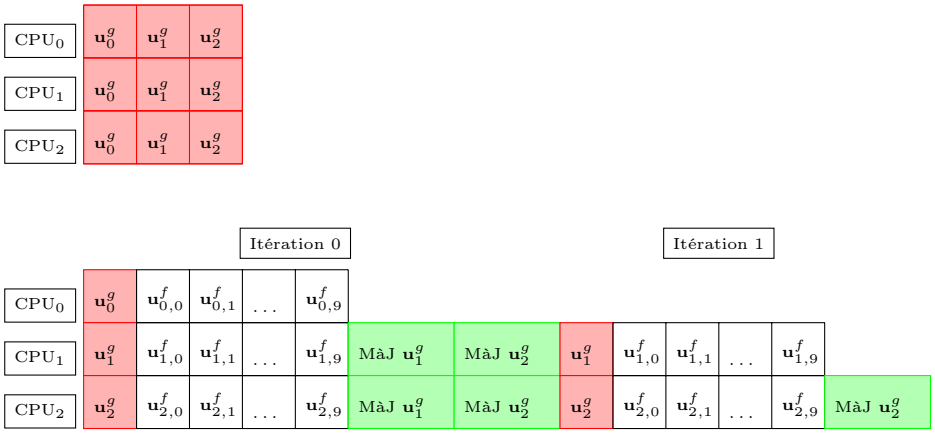


FIGURE 4.4 – Déroulement des étapes de la méthode pararéelle classique.

$\mathbf{u}^f n, M$ doivent être partagés pour la mise à jour. En revanche, parce que le processeur i n'a besoin que de l'état $\mathbf{u}_i^g$ pour pouvoir effectuer sa résolution fine, de nombreux calculs sont inutiles.

Sur la figure 4.5, nous représentons schématiquement le déroulement de la version modifiée de l'algorithme pararéel proposée dans [7] puis dans [14].

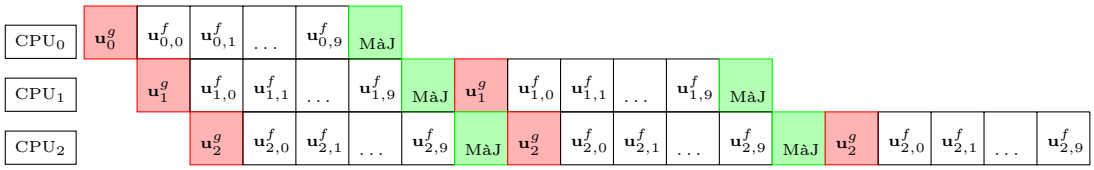


FIGURE 4.5 – Réorganisation de la méthode pararéelle présentée dans [7] puis dans [14].

Contrairement à la version initiale de l'algorithme, cette organisation permet d'optimiser la gestion de la résolution grossière afin de réduire l'attente des processeurs et d'éviter les calculs inutiles. Pour résoudre le problème fin, le processeur i n'a besoin que de l'état $\mathbf{u}_i^g$ afin de déterminer les états $\mathbf{u}_{i,m}^f$. Dans cette version de l'algorithme, un processeur i ne calcule que le i^{e} pas grossier. L'état précédent lui est directement envoyé par le procésseur $i-1$. De même, pour les mises à jour, le processeur i détermine la quantité

$$\mathbf{u}_{i,M}^f - \mathcal{G}(\mathbf{u}_i^g) \tag{4.3}$$

et l'envoie au processeur $i+1$ qui, à son tour, met à jour sa valeur $\mathbf{u}_{i+1}^g$ en ajoutant la quantité (4.3) reçue. Dans cet algorithme, tous les processeurs travaillent de manière déphasée, c'est à dire qu'un processeur ne doit pas démarrer une nouvelle itération s'il n'a pas reçu la mise à jour du processeur précédent. Les communications utilisées sont

donc de type bloquantes. Le nombre de communications entre les processeurs est plus important que dans la version classique de l'algorithme mais le nombre d'étapes de calcul est bien inférieur.

Nous comparons dans le diagramme 4.6 les temps obtenus avec les deux méthodes parareelles sans effectuer l'adaptation du maillage. Pour cela, nous utilisons quatre itérations pour la résolution grossière et quatre itérations pour la résolution fine. Dans les deux cas, nous effectuons les mesures sur quatre processeurs. Nous donnons également les temps correspondants pour la version séquentielle de l'algorithme.

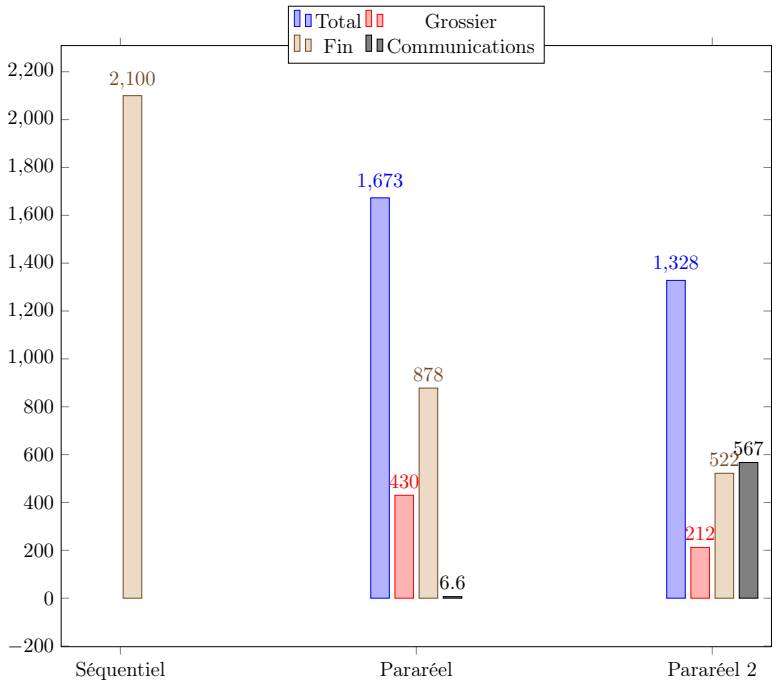


FIGURE 4.6 – Comparaison des temps de calcul pour les deux algorithmes pararéels sans adaptation de maillage pour 4 itérations grossières et 4 itérations fines.

La nouvelle organisation des opérations de la méthode pararéelle permet ainsi d'obtenir un gain de temps de 25% par rapport à la méthode pararéelle classique. Toutefois, nous montrons dans le tableau 4.7 que lorsque nous utilisons l'adaptation du maillage, ce nouvel algorithme est moins efficace que le premier. Comme nous l'avions dit précédemment, les étapes de communications sont plus nombreuses et nécessitent des communications bloquantes. Lorsque les calculs sont nombreux, le temps de ces communications n'est pas une étape principale en termes de temps de calcul. Cependant, lorsque nous effectuons l'adaptation du maillage, nous réduisons fortement la taille du problème (voir figure 3.15) et ainsi le nombre de calculs. Les communications deviennent alors une part importante du temps d'exécution total, ce qui justifie les mesures.

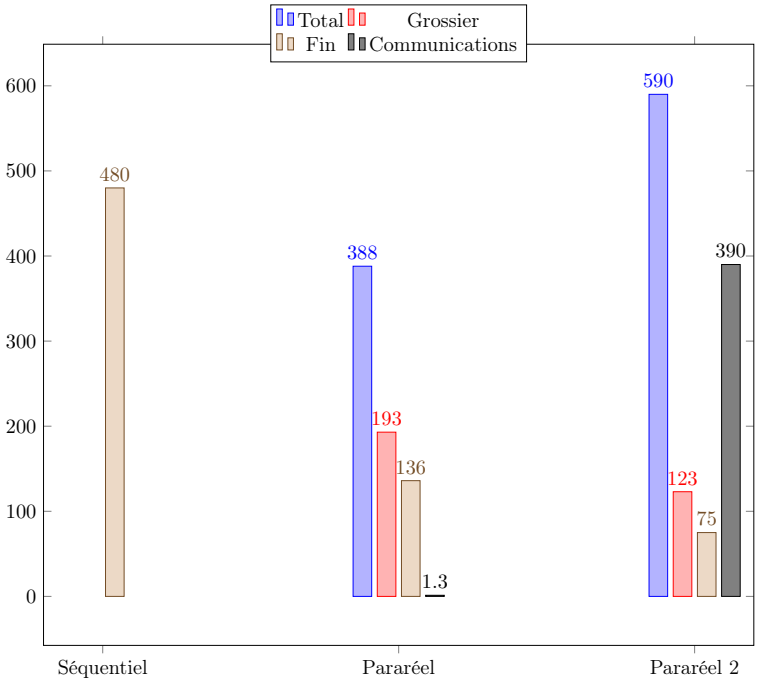


FIGURE 4.7 – Comparaison des temps de calcul pour les deux méthodes pararéelles présentées avec adaptation de maillage pour 4 itérations grossières et 4 itérations fines.

L'adaptation du maillage étant un point essentiel de notre méthode, nous utiliserons donc la méthode pararéelle classique dans la suite des travaux.

4.2 Parallélisation de l'estimation sur une séquence d'image

Afin d'obtenir une meilleure accélération du temps de calcul, nous avons implémenté une méthode de parallélisation de l'estimation du flot optique pour une séquence de plusieurs images successives. En effet, si nous avons un nombre important d'images et que nous souhaitons déterminer le flot optique entre chaque couple d'images, chaque estimation est indépendante. Nous pouvons alors partager les estimations entre les processeurs disponibles comme le montre la figure 4.8.

Pour une séquence de N images, il y a $N-1$ flots optiques à estimer. Donc il est inutile d'effectuer la parallélisation sur $M > N-1$ processeurs sauf si nous souhaitons utiliser plusieurs niveaux de parallélisme comme nous le verrons dans la suite. Pour un nombre $M \leq N-1$ de processeurs, la répartition des tâches se fait comme sur la figure 4.9. Chaque couple d'images $(i, i+1)$, $i = 0, \ldots, M-1$ est envoyé au processeur i qui effectue l'estimation du flot optique. Une file d'attente est ensuite créée pour préparer l'estimation

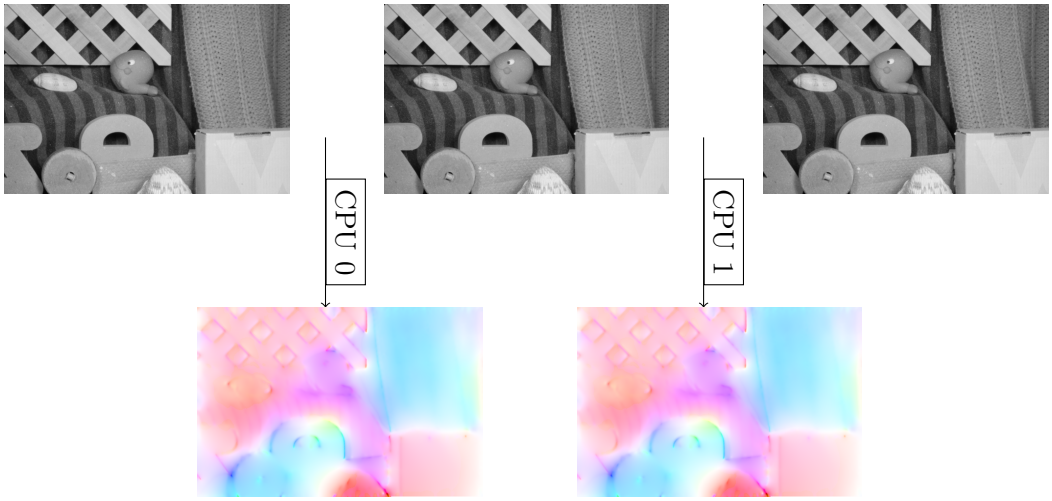


FIGURE 4.8 – Partage des calculs de plusieurs estimations du flot optique dans une séquence de plus de deux images.

des flots optiques pour les images suivantes lorsque les processeurs seront disponibles. Au deuxième lancement des résolutions, les couples d'images $((M-1)+i, (M-1)+(i+1))$ sont envoyés au processeur i.

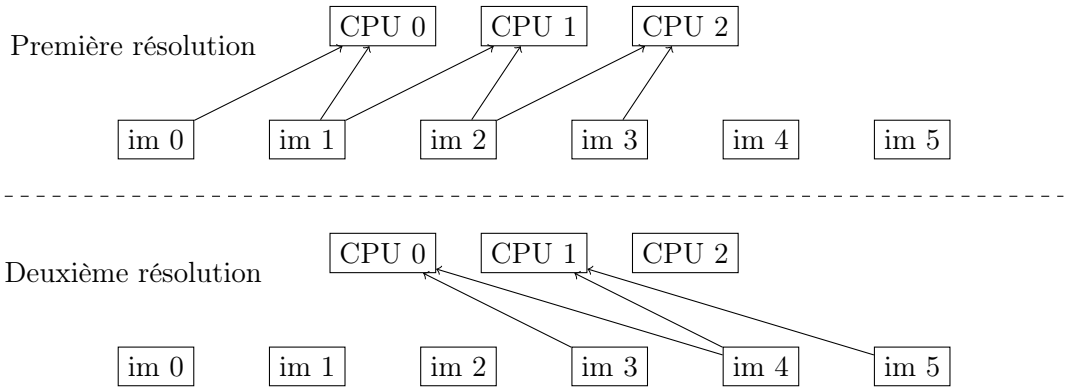


FIGURE 4.9 – Répartition des tâches pour l'estimation de cinq flots optiques sur trois processeurs.

Les estimations étant complétement indépendantes, il n'y a aucune communication entre les différents processeurs. L'accélération théorique ne dépend donc que du nombre de processeurs et du nombre d'images de la séquence. Elle est donnée par

$$\text{Speedup}_{th} = (\text{nbImages-1}) \cdot \left\lceil \frac{\text{nbImages} - 1}{\text{nbCPU}} \right\rceil^{-1}$$

Nb CPU	Speedup
1	1
2	1.7
3	2.5
4	2.5
5	5

FIGURE 4.10 – Temps de calcul pour la parallélisation d'une séquence de 6 images (5 flots optiques à calculer).

où $\lceil \cdot \rceil$ représente la partie entière supérieure.

Dans le tableau 4.10, nous présentons les speedups obtenus avec une séquence de six images (donc cinq estimations du flot optique à calculer). On peut voir que ces accélérations sont égales à la valeur théorique. Cela veut dire qu'il n'y a aucun temps de latence, chaque processeur travaille à tout moment, le gain de temps est donc maximal.

4.3 Décomposition de domaine

4.3.1 Découpage de l'image

Nous venons de voir une méthode simple et efficace pour la parallélisation de l'estimation de plusieurs flots optiques dans une séquence d'images. Une autre problématique à prendre en compte concerne la dimension des images à traiter. En effet, la résolution du problème est dépendante de la dimension des images. Les photos prises par des appareils modernes peuvent atteindre plusieurs millions de pixels. L'estimation du flot optique sur de telles images se révèle donc très coûteuse, c'est pourquoi nous avons implémenté une méthode de parallélisation par décomposition de domaine. Le but est de découper l'image en sous-parties puis chaque processeur reçoit la tâche de déterminer le champ de vecteurs sur une seule sous partie (voir figure 4.11). L'utilisation de cette méthode nous permet d'obtenir des performances difficiles à atteindre avec une méthode variationnelle classique avec en plus un contrôle adaptatif du paramètre de régularisation.

Contrairement au cas précédent, la décomposition de domaine impose une étape de communication entre les différents processeurs. Cette étape a un coût et plus le découpage est important, plus le temps de communication est important. De plus le problème du flot optique est sensible aux conditions de bords donc plus il y a de sous-parties, plus il y a de bords à traiter. Pour ces deux principales raisons, il est important de découper l'image de manière à ce qu'il y ait le moins de pixels sur les frontières des sous parties. Pour cela, on cherche le plus grand ratio $\frac{\text{aire}}{\text{périmètre}}$ où *aire* est le nombre total de pixels dans le sous-domaine et *périmètre* est le nombre de pixels des frontières. Sur la figure 4.12, nous

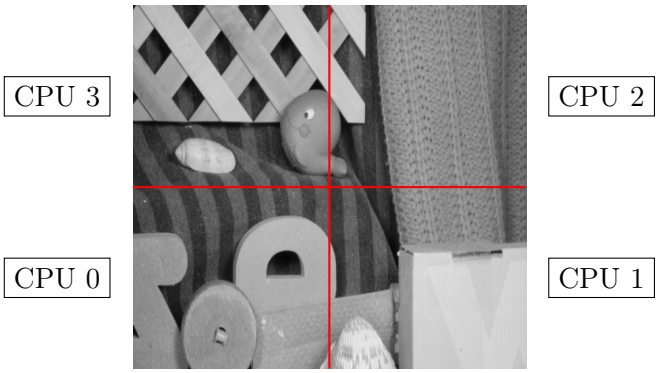


FIGURE 4.11 – Décomposition d'une image pour 4 processeurs.

montrons l'évolution de ce ratio sur une grille de taille 48×48 avec une décomposition pour douze processeurs. Nous avons dans ce cas six choix possibles pour le découpage de l'image : 1×12, 2×6, 3×4, 12×1, 6×2 et 4×3. Dans notre exemple, la grille étant carrée, nous pourrions ignorer les décompositions symétriques puisque le ratio sera le même. Toutefois, nous les conservons dans le but de rester général. Nous pouvons voir, après calcul des différents ratios, que dans le cas d'un découpage d'une image de 48×48 pixels le meilleur découpage en 12 parties est le découpage de l'image en 4×3.

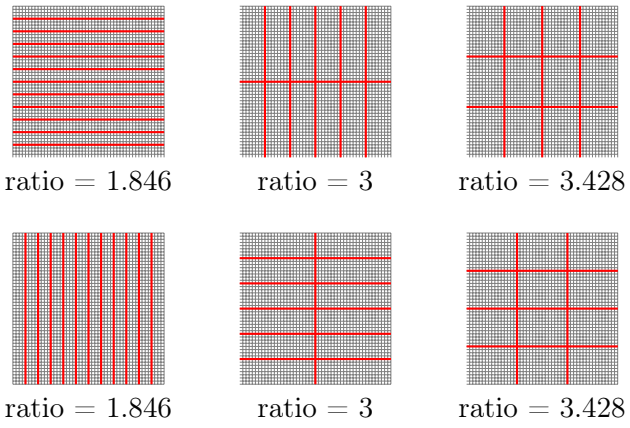


FIGURE 4.12 – Evolution du ratio sur une grille 48×48 pour 12 processeurs.

Comme l'estimation du flot optique est sensible au niveau des frontières, nous allons utiliser la méthode additive de Schwartz avec recouvrement pour améliorer la solution aux interfaces entre les différentes parties. L'utilisation de cette méthode requiert une superposition des sous-domaines, voir figure 4.13.

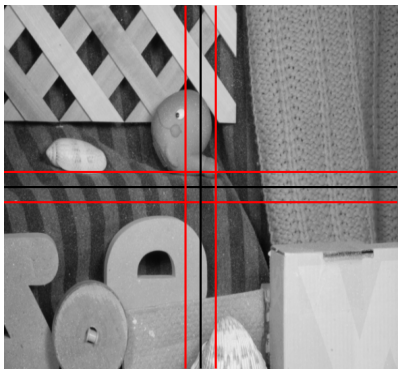


FIGURE 4.13 – Décomposition avec recouvrement pour quatre processeurs.

4.3.2 Décomposition du modèle

On note Ω_i la partie de l'image attribuée au CPU$_i$ et $\mathcal{J}_i$ l'ensemble de tous les indices j voisins de i. On définit $\Sigma_{i,j} = \Omega_i \cap \Omega_j$ et $\Gamma_{i,j} = \partial\Sigma_{i,j} \backslash \partial\Omega_j$ (voir Fig. 4.14).

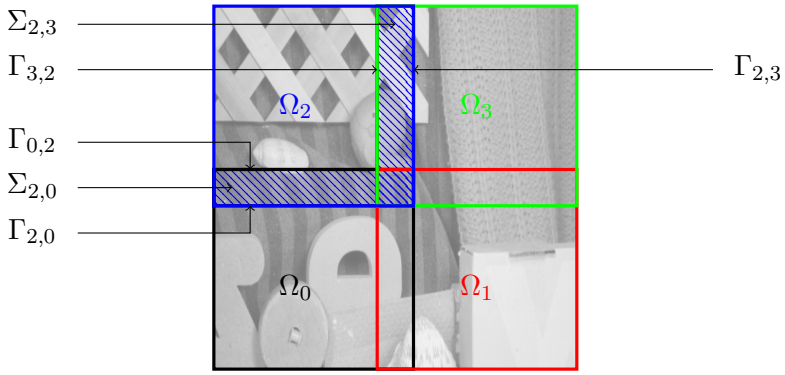


FIGURE 4.14 – Exemple de notations pour le processeur $i = 2$.

En utilisant la méthode additive de Schwarz, la décomposition du problème (3.2) s'écrit sous la forme d'un nouveau système dépendant des sous-domaines. Ce problème consiste dorénavant à trouver une estimation du flot optique $\mathbf{u}_i$ sur le sous-domaine Ω_i. On écrit donc

$$\begin{cases} -\mathrm{div}(\alpha^k \nabla \mathbf{u}_i^k) + J_\rho \mathbf{u}_i^k = \mathbf{f} \text{ sur } \Omega_i \\ \dfrac{\partial \mathbf{u}_i^k}{\partial n} = 0 \text{ sur } \partial\Omega_i \cap \Gamma_{j,i}, \quad \forall j \\ \mathbf{u}_i^k = \mathbf{u}_j^{k-1} \text{ sur } \Gamma_{i,j}. \end{cases} \tag{4.4}$$

Grâce à la méthode de Schwartz, les séquences $(\mathbf{u}_i^k)$ convergent vers $\mathbf{u}_{|\Omega_i}$. Le taux de convergence augmente en fonction de la taille de $\Sigma_{i,j}$. Cependant, nous verrons dans la suite, que le choix de la taille du recouvrement est importante puisqu'une intersection trop grande implique nécessairement un coût de calcul plus important. Il existe une méthode de Schwarz ne nécessitant pas de recouvrement et qui pourrait potentiellement être plus rapide, toutefois cette méthode est plus complexe à mettre en place et nous verrons plus tard que nous n'avons besoin que d'un très petit recouvrement pour obtenir une convergence rapide. Nous pouvons aussi citer une méthode de décomposition de domaine présentée par P.-L. Lions [78] utilisant des conditions aux bords de Robin.

4.3.3 Résultats numériques

Dans cette partie, nous présentons les résultats obtenus pour la décomposition de domaine. Nous avons fait quatre itérations de la méthode de Schwartz en prenant un recouvrement de cinq pixels. Sur les figures 4.15 et 4.16, nous présentons les résultats que nous obtenons en décomposant le domaine en quatre et huit parties. A gauche, nous donnons les résultats après la première itération. Nous pouvons voir que sur les frontières entre les découpages, l'estimation est à améliorer. A droite, nous remarquons qu'après cinq itérations de la méthode de Schwartz, le résultat obtenu est bien meilleur.

FIGURE 4.15 – Estimation du flot optique découpée en quatre sous-domaines. Gauche : Résultat obtenu sans la méthode de Schwartz. Droite : Résultat obtenu après cinq itérations de Schwartz.

Comme nous l'avons dit précédemment, si nous augmentons la taille du recouvrement, nous pouvons réduire le nombre d'itérations de Schwartz. Cependant un grand recouvrement implique un plus gros problème à résoudre et donc un temps d'exécution plus important. Nous montrons sur la figure 4.17 l'effet de la taille du recouvrement sur le temps d'exécution de l'algorithme pour un découpage en quatre sous-domaines.

Pour finir, nous présentons dans le tableau 4.18 les accélérations obtenues avec la même configuration que précédemment mais avec différents découpages. Lorsque le nombre de processeurs est très grand, le gain de temps est proportionnellement moins important

FIGURE 4.16 – Estimation du flot optique découpée en huit sous-domaines. Gauche : Résultat obtenu sans la méthode de Schwartz. Droite : Résultat obtenu après cinq itérations de Schwartz.

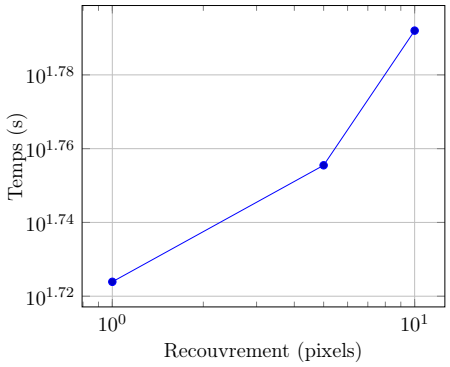


FIGURE 4.17 – Temps d'exécution de l'estimation du flot optique par rapport à la taille du recouvrement.

parce que le nombre de communications augmente.

En plus de fortement réduire les temps de calcul, la méthode de décomposition de domaine permet de résoudre le problème du flot optique sur des images en haute définition. En effet, tandis que les images testées jusqu'ici font partie de séquences de référence utilisées dans la littérature qui ne dépassent pas le million de pixels, les images prises avec les appareils photo actuels, elles, peuvent facilement atteindre plusieurs millions de pixels. Ces images peuvent s'avérer trop lourdes pour les solveurs mathématiques utilisés. En permettant de subdiviser le domaine et d'attribuer le calcul d'un sous-domaine à un processeur, la décomposition de domaine autorise le traitement d'images bien plus grandes. Le travail étant ainsi réparti entre chacun des processeurs, le problème à résoudre se trouve alors réduit. Grâce aux séquences fournies par le laboratoire de Strasbourg du Centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement (Cerema), nous testons notre algorithme sur des images réelles en haute résolution (2028×1098 pixels). Le test présenté sur la figure 4.19 est une série de deux photographies d'un mur à l'intérieur d'un tunnel. Le déplacement y est homogène mais la texture du

Nb CPU	Accélérations
1	1
2	2
4	3.2
8	5.4
16	7.5

FIGURE 4.18 – Accélérations obtenues pour la décomposition de domaine pour cinq pixels de recouvrement et cinq itérations de Schwartz.

mur ainsi que les fortes zones d'occlusions causées par les plaques sur le mur présentent un défi pour ce test. Le second test, figure 4.20, est une séquence en haute définition représentant des véhicules sur l'autoroute. Le mouvement est rapide et les zones d'occlusions sont donc très nombreuses. Sur les figures 4.19 et 4.20, nous présentons également les solutions obtenues avec notre modèle pour une décomposition en 16 sous-parties. Nous donnons les résultats des programmes avec et sans prise en compte de la luminosité. Nous pouvons alors observer que les solutions obtenues sans la prise en compte de la luminosité ne sont pas satisfaisantes. L'algorithme traitant la variation de luminosité donne des résultats plus proche de la solution recherchée.

FIGURE 4.19 – Haut : cas test fourni par Cerema représentant le mur d'un tunnel. Bas : estimation du flot optique obtenue avec (droite) et sans (gauche) traitement de la luminosité. Décomposition en 16 sous-parties.

Enfin, sur la figure 4.21, nous donnons les temps d'exécution de l'algorithme pour ces deux tests en fonction du nombre de sous-parties utilisées. Le graphique montre que la

FIGURE 4.20 – Haut : cas test fourni par Cerema représentant des véhicules sur une autoroute. Bas : estimation du flot optique obtenue avec (droite) et sans (gauche) traitement de la luminosité. Décomposition en 16 sous-parties.

décomposition de domaine en 16 sous-parties a permis d'obtenir une accélération d'un facteur 10 sur le temps de calcul.

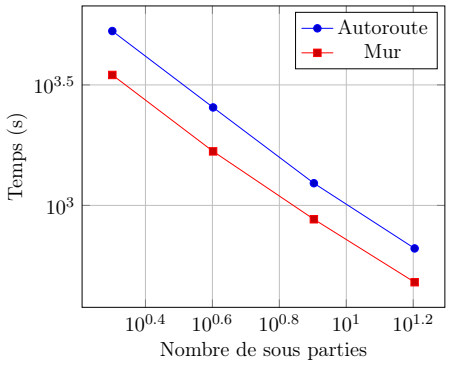


FIGURE 4.21 – Temps d'exécution pour les cas tests de Cerema en fonction du nombre de sous-parties.

4.4 Parallélisme multi-niveaux : vers une méthode de calcul intensif

Nous venons de voir que la méthode de décomposition de domaine est une bonne stratégie en termes de scalabilité. Toutefois, une image ne pouvant pas etre décomposée un trop grand nombre de fois, la méthode, bien que très efficace, reste limitée par ce cri-

tère. Afin d'arriver à une méthode parallèle multi-niveaux, nous allons opérer plusieurs couplages. Dans un premier temps, nous étudierons le couplage de la méthode avec un solveur massivement parallèle, le solveur MUMPS (MUltifrontal Massively Parallel sparse direct Solver). Ensuite, nous couplerons la méthode de décomposition de domaine avec la méthode pararéelle présentée précedemment dans ce chapitre.

4.4.1 Couplage entre la décomposition de domaine et le solveur MUMPS

Nous avons vu que la décomposition de domaine permet d'obtenir une bonne accélération du temps de calcul. Cependant, elle permet également de traiter des images beaucoup plus grandes. En effet, avec le logiciel FreeFem++, le solveur par défaut pour les systèmes linéaires est UMFPACK (Unsymmetric MultiFrontal method). C'est un ensemble de fonctions pour résoudre des systèmes d'équations linéaires de la forme

$$Ax = b$$

même si la matrice A n'est pas symétrique. L'inconvénient de cette librairie est qu'il n'est possible de résoudre que des problèmes dont le stockage des données est plus petits que 4GB. Ainsi, si l'on veut travailler avec des images réelles de 2000×1000 pixels, il nous faut passer par la décomposition de domaine.

Cependant, il existe un solveur massivement parallèle appelé MUMPS (MUltifrontal Massively Parallel sparse direct Solver). Cette librairie permet de résoudre des problèmes linéaires creux et n'est pas limitée par la taille du problème. Cela veut dire que nous pouvons estimer le flot optique pour des images aussi grandes que l'on souhaite en parallèle sans faire de travail particulier. L'inconvénient est que nous avons besoin de pré-conditionner le système en faisant une décomposition LU de la matrice A. Avec un problème assez large, cette décomposition pourrait devenir longue et annuler le gain de temps obtenu grâce au solveur MUMPS.

En couplant la décomposition de domaine appliquée à l'image et le solveur MUMPS, on peut alors résoudre de manière parallèle de nombreux problèmes plus petits et ainsi avoir de plus petites matrices à pré-conditionner. Dans la suite, nous appellerons cette méthode, la méthode DDM-MUMPS. Dans la décomposition de domaine classique présentée précédemment, nous avons été limités par le nombre de processeurs car nous ne pouvions pas couper notre image en parties trop petites. A présent que l'on sépare l'image comme présenté sur la Fig 4.22, nous pouvons alors réellement augmenter le parallélisme.

L'image est découpée de la même manière que dans la partie précédente, c'est à dire en fonction du ratio maximal entre le périmètre et l'aire de la sous-partie. On garde également le même recouvrement entre les parties et pour des raisons pratiques, nous

FIGURE 4.22 – Décomposition de l'image pour quatre groupes de processeurs.

avons gardé le même nombre de groupes.

La distribution des processeurs dans les groupes est faite comme suit

$$\text{groupe}(\text{CPU}_i) = \frac{\text{CPU}_i}{\text{nbPart}}$$

où nbPart est le nombre de parties composant l'image découpée. Le reste du travail reste le même que la précédente décomposition de domaine, excepté le fait que nous agissons sur un groupe de processeurs et non plus sur un seul processeur. L'ordre des processeurs à l'intérieur d'un même groupe peut être aléatoire.

4.4.1.1 Temps d'exécution

Afin de tester l'efficacité de la méthode DDM-MUMPS, nous avons résolu une estimation de flot optique dans différentes configurations. Pour différents nombres de sous-domaines (2, 4 et 8), nous avons exécuté le code avec 1 ou 4 processeurs par sous-domaines. Pour des raisons d'égalité dans la méthode de calcul du temps, nous avons toujours fait dix itérations de la méthode de Schwarz et nous avons utilisé un recouvrement de cinq pixels.

Sur la figure 4.23, on commence par présenter les temps de calcul globaux observés.

Afin de comprendre pourquoi l'utilisation de quatre processeurs par sous-domaine n'est pas aussi efficace que l'on pourrait s'y attendre lorsqu'on a un grand nombre de sous-domaines, on présente sur la figure 4.24 les temps de communication obtenus selon les configurations. Il est évident que si nous augmentons le nombre de processeurs, les temps de communications sont plus importants puisqu'il y a plus de communications à effectuer.

L'augmentation des temps de communication n'est pas suffisante pour expliquer le faible

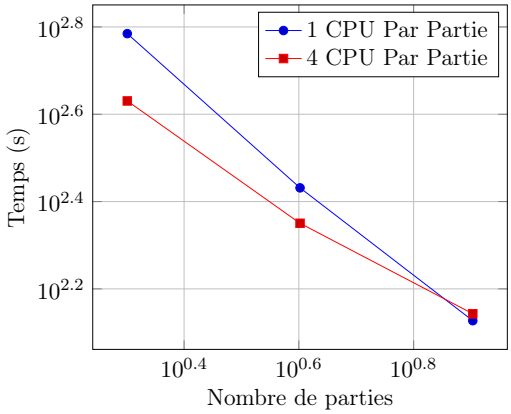


FIGURE 4.23 – Temps d'exécution de l'algorithme parallèle multi-niveaux pour différentes configurations.

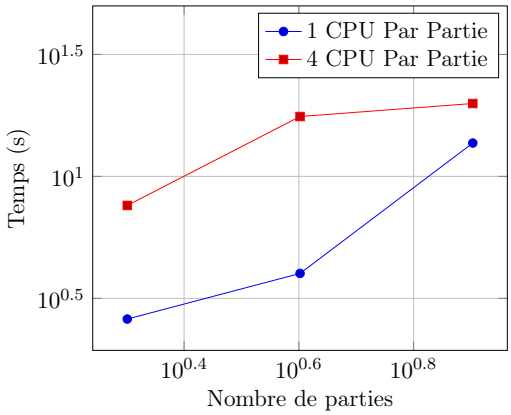


FIGURE 4.24 – Temps de communication en fonction de la configuration. Ces temps n'incluent pas les temps de communications dus au solveur MUMPS.

gain de temps de l'algorithme. En fait dans toute implémentation parallèle, le coût des communications a un effet négatif sur la performance de l'implémentation. Cependant, en règle générale le temps utilisé pour cette étape doit être compensé par le gain de temps obtenu par la méthode parallèle, dans notre cas le multi-niveaux. Afin d'étudier en détail le temps consommé par les différentes parties de l'algorithme, on présente dans la figure 4.25 les temps des deux principales parties de notre code, à savoir la construction de la matrice de masse et le préconditionnement de cette matrice avec la résolution du système linéaire. On observe que l'ajout de processeurs à un sous-domaine réduit légèrement le temps de construction de la matrice. En revanche, le gain de temps est plus significatif dans la résolution du système et le préconditionnement de la matrice. Si nous augmentons le nombre de sous-domaines, la partie principale du calcul se trouvera dans la construction de la matrice. Donc en ajoutant un parallélisme dans les sous-parties, l'étape de résolution du système n'étant plus l'étape la plus coûteuse, la proportion de temps gagné sur cette

partie n'est plus suffisante pour compenser l'augmentation du temps de calcul causé par les communications.

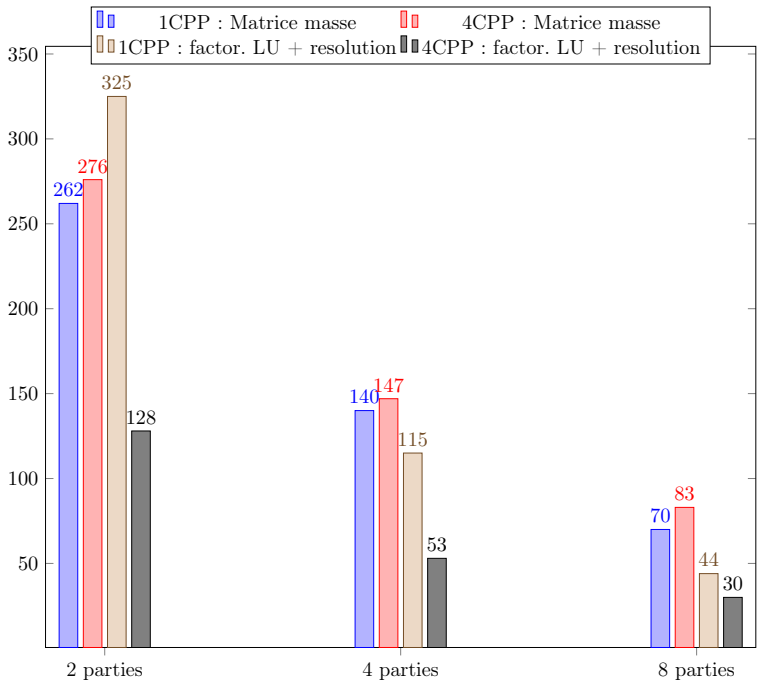


FIGURE 4.25 – Temps de calcul détaillés pour 2, 4 et 8 séparations avec un processeur par partie (1 CPP) ou quatre processeurs par partie (4 CPP).

4.4.2 Couplage entre la décomposition de domaine et la méthode parareelle

Pour finir ce chapitre et dans le but de préparer le chapitre 5 sur les grands déplacements, nous présentons une méthode de couplage entre la méthode de décomposition de domaine et la méthode parareelle que nous appellerons la méthode DDM-Para. Ce couplage a pour but d'effectuer une résolution parareelle à l'intérieur des sous-domaines. Comme dans le cas de la méthode DDM-MUMPS, nous attribuons plusieurs processeurs à chaque sous-domaine. Nous réutilisons la même méthode pour la distribution des processeurs entre les différents sous-domaines, le numéro du groupe pour le CPU_i est donné par

$$\text{groupe}(\text{CPU}_i) = \frac{\text{CPU}_i}{\text{nbPart}}.$$

Concernant la méthode parareelle, nous gardons la numérotation des processeurs entre 0 et $N-1$ à l'intérieur d'un sous-domaine. Dans ce but, nous définissons une nouvelle nu-

mérotation locale. Sur la figure 4.26 nous représentons la répartition de seize processeurs sur quatre sous-domaines.

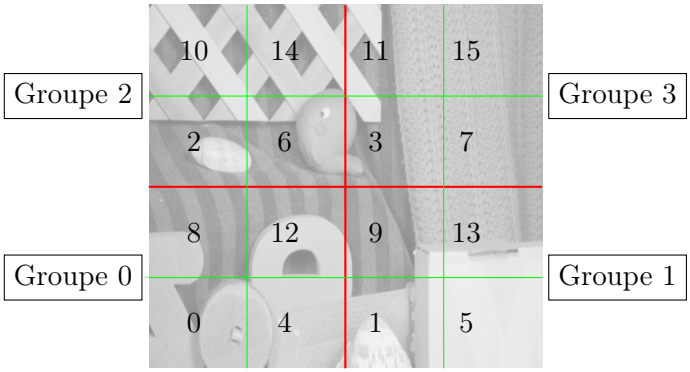


FIGURE 4.26 – Numérotation globale pour 16 processeurs sur 4 sous-domaines.

Le passage à une numérotation locale se fait à l'aide de la formule de changement de numérotation suivante

$$\mathrm{CPU}_i^{\mathrm{loc}} = \mathrm{CPU}_i[\mathrm{nbPart}]$$

La numérotation locale des processeurs est représentée sur la figure 4.27.

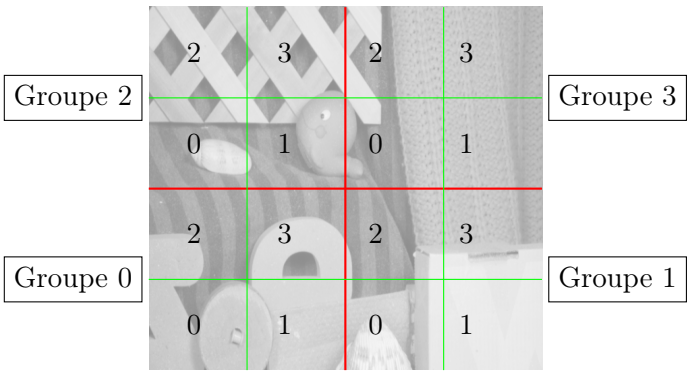


FIGURE 4.27 – Numérotation locale pour 16 processeurs sur 4 sous-domaines.

Afin d'évaluer le gain de temps obtenu avec cette méthode, nous faisons un test avec quatre itérations pour le solveur grossier et quatre itérations pour le solveur fin. Les temps d'exécution en fonction du nombre de sous-domaines sont représentés sur la figure 4.28.

Pour un découpage en deux parties, nous obtenons un gain de 33% par rapport à la méthode pararéelle. Les communications ainsi que la nécessité de synchroniser les pro-

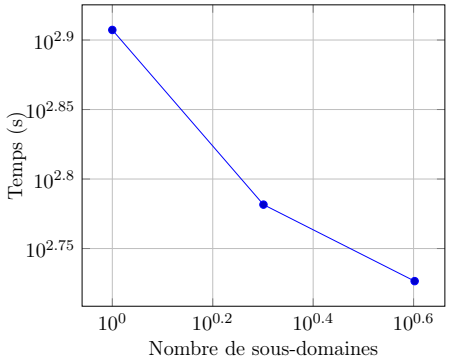


FIGURE 4.28 – Temps d'exécution de calcul du flot optique par la méthode DDM-Para en fonction du nombre de sous-domaines.

cesseurs au cours du calcul provoquent des temps de latence lors de l'exécution de l'algorithme empêchant de parvenir à une scalabitité. Pour quatre sous-parties le gain de temps est ainsi de 50%. Néanmoins, le temps de calcul continue de diminuer au fur et à mesure que l'on augmente le découpage. L'insertion d'une décomposition de domaine à l'intérieur d'un algorithme pararéel permet donc un gain de temps considérable comparé à la méthode pararéelle implémentée seule.

4.5 Conclusion

Dans ce chapitre, nous avons présenté plusieurs méthodes de parallélisation pour l'estimation du flot optique. Les méthodes présentées sont indépendantes de la méthode de résolution choisie. Elles peuvent s'appliquer aussi bien à des méthodes variationnelles qu'à des méthodes indirectes ou des méthodes de filtrages. La parallélisation d'estimations de plusieurs flots optiques dans une séquence vidéo est la plus intuitive mais elle est aussi la plus efficace en termes de nombre de processeurs et accélération. En effet, de par le fait que les estimations sont complétement indépendantes, il n'y a aucune perte de temps consacré à des communications, donc les processeurs sont constamment utilisés.

La décomposition de domaine, avec une gestion efficace des communications, a permis d'obtenir de bonnes accélérations de la résolution. Elle permet également le traitement d'images dont la résolution est plus élevée, ce qui se rapproche plus des images aquises avec les technologies actuelles. Nous avons aussi vu que sur une architecture disposant de nombreux processeurs, l'utilisation d'une parallélisation multi-niveaux à l'aide du solveur parallèle MUMPS permet d'augmenter encore l'accélération du programme. Toutefois, nous avons mis en évidence la nécessité d'un contrôle efficace de la répartition des processeurs. En effet, un découpage trop fin de l'image en sous-domaines peut entraîner d'une part une perte de la qualité des résultats due à la sensibilité de l'estimation sur les bords

des sous-domaines, mais également une perte de temps causée par une augmentation du temps des communications qui n'est plus compensée par le gain de temps donné par la parallélisation de la méthode.

Enfin, nous avons commencé ce chapitre par la présentation détaillée de la méthode pararéelle. Nous avons pu voir que, dans le cadre de l'estimation du flot optique, les résultats obtenus ne sont pas assez satisfaisants en termes de temps de calcul. La cause n'étant pas un problème d'implémentation, ni un problème de méthode mais un problème lié à la taille du problème du flot optique. En effet, nous avons vu que la méthode pourrait être efficace pour un grand nombre de processeurs et à condition de trouver un solveur grossier qui soit rapide et suffisamment précis. Un couplage de cette méthode avec la méthode de décomposition de domaine a cependant permis de réduire les temps de calcul.

Chapitre 5

Estimation du flot optique en grands déplacements

Sommaire

Parmi les hypothèses initiales du flot optique, deux d'entre elles menent à des restrictions concernant les images à traiter. La première de ces hypothèses (1.1) nous imposait de ne travailler qu'avec des séquences d'images dont la luminosité est constante. Dans la partie 3.1.2, nous avons vu une méthode permettant de contourner cette restriction. La deuxième hypothèse a été de considérer que les déplacements sont petits. Cette hypothèse nous a permis de supposer que la fonction f était suffisamment régulière et ainsi de linéariser l'équation (1.1) pour obtenir la contrainte fondamentale du flot optique (1.3). Nous allons à présent étudier le cas où les déplacements ne se limitent pas qu'à quelques pixels. En effet, lors de l'aquisition d'une séquence vidéo il peut être intéressant, pour des raisons de stockage mémoire, d'effectuer un enregistrement où le nombre d'images par seconde est faible. Le mouvement entre deux images successives pourrait alors être relativement grand. Il en est de même dans le cas d'une vidéo d'une scène où les éléments

sont rapides comme le trafic sur une autoroute par exemple. Nous nous intéressons dans ce chapitre à l'estimation du flot optique pour des grands déplacements. Nous suivons un plan équivalent à celui du chapitre 2.3. Après avoir présenté le problème du flot optique dans le cas des grands déplacements, sans puis avec prise en compte de la variation de la luminosité, nous étudions l'application de la méthode de contrôle adaptatif vue dans le chapitre 2.3. Nous terminons ce chapitre par la présentation des résultats numériques que nous comparons avec ceux de la méthode LUM-REGU$_{\text{loc}}$.

5.1 Problème du flot optique en grands déplacements

5.1.1 Luminosité constante

Pour l'estimation du flot optique en grands déplacements, nous ne pouvons plus supposer que la fonction est régulière. De ce fait, nous ne pouvons plus utiliser un développement de Taylor afin de linéariser l'équation sur la constance de la luminosité

$$f(x, y, t) = f(x + u_1, y + u_2, t + 1)$$

pour obtenir la contrainte fondamentale du flot optique

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} = 0.$$

L'équation (3.1) s'écrit donc

$$\int_\Omega \underbrace{K_\rho \star (f(x + u_1, y + u_2, t + 1) - f(x, y, t))^2 dxdy}_{\text{non-linéaire}} + \int_\Omega \underbrace{\alpha(\mathbf{x})(|\nabla u_1|^2 + |\nabla u_2|^2)}_{\text{linéaire}}. \quad (5.1)$$

Par souci de lisibilité, dans la suite la fonction f désignera $K_\rho \star f$. Le système d'Euler-Lagrange associé au problème (5.1) s'écrit

$$\begin{cases} -\operatorname{div}(\alpha(\mathbf{x})\nabla u_1) + (f(x+u_1, y+u_2, t+1) - f(x,y,t)) f_x(x+u_1, y+u_2, t+1) = 0 \\ -\operatorname{div}(\alpha(\mathbf{x})\nabla u_2) + (f(x+u_1, y+u_2, t+1) - f(x,y,t)) f_y(x+u_1, y+u_2, t+1) = 0. \end{cases} \tag{5.2}$$

Afin de lever la non-linéarité de ce système, nous utilisons la méthode itérative du point fixe. Cette méthode consiste à construire une suite de solutions convergeant vers la solution du problème non-linéaire.

En discrétisant le terme non linéaire semi-implicitement, le système (5.2) s'écrit

$$\begin{cases} -\operatorname{div}(\alpha(\mathbf{x})\nabla u_1^{k+1}) + (f(x+u_1^{k+1}, y+u_2^{k+1}, t+1) - f(x,y,t)) f_x(x+u_1^{k}, y+u_2^{k}, t+1) = 0 \\ -\operatorname{div}(\alpha(\mathbf{x})\nabla u_2^{k+1}) + (f(x+u_1^{k+1}, y+u_2^{k+1}, t+1) - f(x,y,t)) f_y(x+u_1^{k}, y+u_2^{k}, t+1) = 0. \end{cases} \tag{5.3}$$

Ce système restant non-linéaire à cause du terme $f(x+u_1^{k+1}, y+u_2^{k+1}, t+1)$, nous décomposons les inconnues à l'état $k+1$ de la manière suivante

$$\begin{aligned} u_1^{k+1} &= u_1^k + du_1^k \\ u_2^{k+1} &= u_2^k + du_2^k. \end{aligned}$$

Nous linéarisons ensuite le système à l'aide d'un développement de Taylor. Cette linéarisation est maintenant compatible avec le problème du flot optique en grands déplacements puisqu'elle est appliquée non pas sur le problème initial mais sur le problème discrétisé (5.3). Nous obtenons

$$\begin{aligned} f(x+u_1^{k+1}, y+u_2^{k+1}, t+1) \approx\ & f(x+u_1^k, y+u_2^k, t+1) \\ & + f_x(x+u_1^k, y+u_2^k, t+1) du_1^k \\ & + f_y(x+u_1^k, y+u_2^k, t+1) du_2^k. \end{aligned}$$

Après avoir injecté cette approximation dans le système (5.3), nous obtenons finalement le système d'Euler-Lagrange linéarisé correspondant au problème de minimisation de la fonctionnelle (5.1)

$$\begin{cases} -\operatorname{div}(\alpha(\mathbf{x})\nabla du_1^k) + (f_x(x+u_1^k, y+u_2^k, t+1) du_1^k + f_y(x+u_1^k, y+u_2^k, t+1) du_2^k) f_x(x+u_1^k, y+u_2^k, t+1) \\ \quad = \operatorname{div}(\alpha(\mathbf{x})\nabla u_1^k) - (f(x+u_1^k, y+u_2^k, t+1) - f(x,y,t)) f_x(x+u_1^k, y+u_2^k, t+1) \\ -\operatorname{div}(\alpha(\mathbf{x})\nabla du_2^k) + (f_x(x+u_1^k, y+u_2^k, t+1) du_1^k + f_y(x+u_1^k, y+u_2^k, t+1) du_2^k) f_y(x+u_1^k, y+u_2^k, t+1) \\ \quad = \operatorname{div}(\alpha(\mathbf{x})\nabla u_2^k) - (f(x+u_1^k, y+u_2^k, t+1) - f(x,y,t)) f_y(x+u_1^k, y+u_2^k, t+1). \end{cases}$$

Sous forme vectorielle, ce système s'écrit

$$-\operatorname{div}(\alpha(\mathbf{x})\nabla d\mathbf{u}^k) + \mathbf{J}^k d\mathbf{u}^k = \operatorname{div}(\alpha(\mathbf{x})\nabla \mathbf{u}^k) + \mathbf{F}^k$$

avec

$$d\mathbf{u}^k = \begin{pmatrix} du_1^k \\ du_2^k \end{pmatrix}, \ \mathbf{J}^k = \begin{pmatrix} J_{1,1}^k & J_{1,2}^k \\ J_{1,2}^k & J_{2,2}^k \end{pmatrix},$$

où

$$\begin{aligned} J_{1,1}^k &= f_x^2(x+u_1^k, y+u_2^k, t+1) \\ J_{2,2}^k &= f_y^2(x+u_1^k, y+u_2^k, t+1) \\ J_{1,2}^k &= f_y(x+u_1^k, y+u_2^k, t+1) f_x(x+u_1^k, y+u_2^k, t+1) \end{aligned}$$

et

$$\mathbf{F}^k = \begin{pmatrix} -(f(x+u_1^k, y+u_2^k, t+1) - f(x,y,t)) f_x(x+u_1^k, y+u_2^k, t+1) \\ -(f(x+u_1^k, y+u_2^k, t+1) - f(x,y,t)) f_y(x+u_1^k, y+u_2^k, t+1) \end{pmatrix}.$$

Les inconnues de ce système sont à présent du_1^k et du_2^k, nous construisons alors la suite de solutions $\mathbf{u}^k$ en appliquant à chaque fin d'itération la mise à jour

$$\mathbf{u}^{k+1} = \mathbf{u}^k + d\mathbf{u}^k.$$

La solution u^k étant connue, nous déterminons la fonction $f(x+u_1^k, y+u_2^k, t+1)$ à l'aide d'une interpolation bilinéaire comme dans le cas de la méthode de rapprochement itératif présentée dans le chapitre 2.3. Nous nommons cette méthode GD-REGU$_{\text{loc}}$.

5.1.2 Prise en compte de la luminosité

Dans le chapitre 2.3, nous avons vu que dans le cas où la séquence d'images présente des variations lumineuses, la méthode se basant sur l'hypothèse fondamentale du flot optique (1.1) ne suffit plus. Nous avons alors considéré que la luminosité pouvait varier de manière linéaire et donc remplacé cette contrainte par l'équation (3.3).

Dans le cas de l'étude des grands déplacements, l'image est plus sujette à des variations importantes de luminosité ou à des occlusions causées par des mouvements rapides sur les arêtes de la scène. Pour ces raisons, nous reprenons dans ce chapitre, le modèle présenté dans la section 3.1.2 afin de prendre en compte les variations de lumière. Nous appellerons ce nouvel algorithme GD-LUM-REGU$_{\text{loc}}$. La fonctionnelle (5.1) devient alors

$$\int_\Omega \underbrace{K_\rho \star (f(x+u_1, y+u_2, t+1) - Mf(x,y,t))^2 dxdy}_{\text{non-linéaire}} + \int_\Omega \underbrace{\alpha(|\nabla u_1|^2 + |\nabla u_2|^2) + \lambda |\nabla M|^2 dxdy}_{\text{linéaire}}. \quad (5.4)$$

En considérant $f = K_\rho \star f$, nous écrivons le système associé au problème de minimisation de cette fonctionnelle

$$\begin{cases} -\operatorname{div}(\alpha(\mathbf{x})\nabla u_1) + (f(x+u_1, y+u_2, t+1) - Mf(x,y,t)) f_x(x+u_1, y+u_2, t+1) = 0 \\ -\operatorname{div}(\alpha(\mathbf{x})\nabla u_2) + (f(x+u_1, y+u_2, t+1) - Mf(x,y,t)) f_y(x+u_1, y+u_2, t+1) = 0 \\ -\operatorname{div}(\lambda(\mathbf{x})\nabla M) - (f(x+u_1, y+u_2, t+1) - Mf(x,y,t)) f(x,y,t) = 0. \end{cases} \quad (5.5)$$

Il est possible de résoudre la troisième équation de ce système afin de déterminer M puis d'utiliser ce résultat afin de résoudre la première et la seconde équation de façon indépendante. Cependant, cette solution est relativement grossière, c'est pourquoi nous choisissons d'écrire le schéma itératif pour la variable étendue $\mathbf{U} = (u_1, u_2, M)$.

Ce système étant non-linéaire, nous allons utiliser le même procédé itératif que celui présenté dans la section précédente. Le passage de l'itération k à $k+1$ sera considéré comme un petit déplacement. En effet, le choix de cette méthode itérative est une semi-linéarisation qui va permettre d'utiliser les chapitres 2.3 et 4.

La discrétisation semi-implicite nous donne le système

$$\begin{cases} -\operatorname{div}(\alpha(\mathbf{x})\nabla u_1^{k+1}) + (f(x+u_1^{k+1}, y+u_2^{k+1}, t+1) - M^{k+1} f(x,y,t)) f_x(x+u_1^k, y+u_2^k, t+1) = 0 \\ -\operatorname{div}(\alpha(\mathbf{x})\nabla u_2^{k+1}) + (f(x+u_1^{k+1}, y+u_2^{k+1}, t+1) - M^{k+1} f(x,y,t)) f_y(x+u_1^k, y+u_2^k, t+1) = 0 \\ -\operatorname{div}(\lambda(\mathbf{x})\nabla M^{k+1}) - (f(x+u_1^{k+1}, y+u_2^{k+1}, t+1) - M^{k+1} f(x,y,t)) f(x,y,t) = 0. \end{cases} \quad (5.6)$$

En posant

$$
\begin{aligned}
u_1^{k+1} &= u_1^k + du_1^k \\
u_2^{k+1} &= u_2^k + du_2^k \\
M^{k+1} &= M^k + dM^k
\end{aligned}
$$

et en utilisant un développement de Taylor, nous obtenons le système linéarisé de la fonctionnelle (5.4)

$$
\left\{
\begin{array}{l}
(f_x(x+u_1^k, y+u_2^k, t+1)du_1^k + f_y(x+u_1^k, y+u_2^k, t+1)du_2^k - dM^k f(x,y,t))f_x(x+u_1^k, y+u_2^k, t+1) \\
-\operatorname{div}(\alpha(\mathbf{x})\nabla du_1^k) = \operatorname{div}(\alpha(\mathbf{x})\nabla u_1^k) - (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t))f_x(x+u_1^k, y+u_2^k, t+1) \\
\\
(f_x(x+u_1^k, y+u_2^k, t+1)du_1^k + f_y(x+u_1^k, y+u_2^k, t+1)du_2^k - dM^k f(x,y,t))f_y(x+u_1^k, y+u_2^k, t+1) \\
-\operatorname{div}(\alpha(\mathbf{x})\nabla du_2^k) = \operatorname{div}(\alpha(\mathbf{x})\nabla u_2^k) - (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t))f_y(x+u_1^k, y+u_2^k, t+1) \\
\\
-(f_x(x+u_1^k, y+u_2^k, t+1)du_1^k + f_y(x+u_1^k, y+u_2^k, t+1)du_2^k - dM^k f(x,y,t))f(x,y,t) \\
-\operatorname{div}(\lambda(\mathbf{x})\nabla dM^k) = \operatorname{div}(\lambda(\mathbf{x})\nabla M^k) + (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t))f(x,y,t).
\end{array}
\right.
$$

Sous forme vectorielle, ce système s'écrit

$$
-\operatorname{div}(\mathbf{\Lambda}(\mathbf{x})\nabla d\mathbf{U}^k) + \mathbf{A}^k d\mathbf{U}^k = \operatorname{div}(\mathbf{\Lambda}(\mathbf{x})\nabla \mathbf{U}^k) + \mathbf{F}^k \tag{5.7}
$$

avec

$$
d\mathbf{U}^k = \begin{pmatrix} du_1^k \\ du_2^k \\ dM^k \end{pmatrix}, \ \mathbf{A}^k = \begin{pmatrix} A_{1,1}^k & A_{1,2}^k & A_{1,3}^k \\ A_{1,2}^k & A_{2,2}^k & A_{2,3}^k \\ A_{1,3}^k & A_{2,3}^k & A_{3,3}^k \end{pmatrix},
$$

où

$$
\begin{aligned}
A_{1,1}^k &= f_x^2(x+u_1^k, y+u_2^k, t+1) \\
A_{2,2}^k &= f_y^2(x+u_1^k, y+u_2^k, t+1) \\
A_{3,3}^k &= f^2(x,y,t) \\
A_{1,2}^k &= f_y(x+u_1^k, y+u_2^k, t+1)f_x(x+u_1^k, y+u_2^k, t+1) \\
A_{1,3}^k &= -f(x,y,t)f_x(x+u_1^k, y+u_2^k, t+1) \\
A_{2,3}^k &= -f(x,y,t)f_y(x+u_1^k, y+u_2^k, t+1),
\end{aligned}
$$

et

$$
\mathbf{F}^k = \begin{pmatrix} -(f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t))f_x(x+u_1^k, y+u_2^k, t+1) \\ -(f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t))f_y(x+u_1^k, y+u_2^k, t+1) \\ (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t))f(x,y,t) \end{pmatrix}.
$$

Notons que, la troisième équation de (5.6) peut être résolue indépendemment des deux autres. Cependant, nous choisissons de la conserver dans le système puisque le cas où M est linéaire couvre plus d'applications dans la pratique.

Comme dans le cas des petits déplacements, lorsque la fonction f est nulle, la matrice $\mathbf{A}_\rho$ n'est plus définie positive. Afin d'anticiper ce problème, dans une pareille situation, nous ajoutons articiellement un paramètre ε de l'ordre 10^{-8} et considérons ainsi le terme $f^2 + \varepsilon$ à la place de f^2.

Contrairement au problème du flot optique en petits déplacements (chapitre 2.3), l'étude du problème pour les grands déplacements est plus complexe puisque l'énergie est non convexe. Nous ne pouvons donc pas prouver de manière analytique la convergence du modèle. Nous étudierons alors l'évolution du schéma pour un grand nombre d'opérations dans la section des résultats numériques.

5.1.3 Discrétisation

Soit $\mathcal{T}_h$, un maillage tel que présenté sur la figure 3.1. Nous définissons l'espace d'approximation de la solution

$$\mathcal{V}_h = \{\mathbf{V}_h \in (C(\overline{\Omega}))^3, \quad \mathbf{V}_h|_K \in (P_1(K))^3\}$$

où, h représente le diamètre maximal des cellules K, $C(\overline{\Omega})$ représente l'ensemble des fonctions continues sur $\overline{\Omega}$ et $P_1(K)$ est l'ensemble des fonctions polynômiales de degré inférieur ou égal à 1 sur K.

Dans cet espace, nous notons $\mathbf{A}_h$ et $a_{\mathbf{\Lambda},h}$, les approximations respectives de $\mathbf{A}$ et $a_{\mathbf{\Lambda}}$, et $L_h(\mathbf{V})$ l'approximation de $L(\mathbf{V})$, où

$$a_{\mathbf{\Lambda}}(d\mathbf{U}_{\mathbf{\Lambda}}, \mathbf{V}) = \int_\Omega \mathbf{\Lambda}(\mathbf{x}) \nabla d\mathbf{U}_{\mathbf{\Lambda}} \nabla \mathbf{V} d\mathbf{x} + \int_\Omega \mathbf{V}^t \mathbf{A} d\mathbf{U}_{\mathbf{\Lambda}} d\mathbf{x}$$

et

$$L(\mathbf{V}) = \int_\Omega \mathbf{F}\mathbf{V} + \mathbf{\Lambda}(\mathbf{x}) \nabla \mathbf{U} \nabla \mathbf{V} d\mathbf{x}.$$

Lors de l'implémentation numérique, nous choisirons $\mathbf{\Lambda}(\mathbf{x}) = \mathbf{\Lambda}_0$ dans le second membre. En effet, nous pouvons considérer que pour l'adaptation de $\mathbf{\Lambda}$, l'ensemble $|\{|\mathbf{\Lambda}| = \min(\alpha, \lambda)\}| < \epsilon$.

La formulation variationnelle discrétisée consiste à trouver la solution faible $d\mathbf{U}_{\mathbf{\Lambda},h} \in \mathcal{V}_h$ telle que

$$a_{\mathbf{\Lambda},h}(d\mathbf{U}_{\mathbf{\Lambda},h}, \mathbf{V}_h) = L_h(\mathbf{V}_h) \quad \forall \mathbf{V}_h \in \mathcal{V}_h.$$

5.1.4 Variation locale et adaptative du paramètre α et implémentation

Les déplacements étant plus importants, nous allons permettre à u_1 et u_2 d'être adaptés séparément en définissant deux indicateurs d'erreurs, η_K^1 et η_K^2. Cela nous permettra de considérer l'anisotropie. Se basant sur la méthode d'adaptation présentée dans le chapitre 2.3, sans changer la formule, nous définissons ainsi les indicateurs de manière indépendante. En notant $F_{1,h}$ et $F_{2,h}$ les approximations des deux premières composantes de $\mathbf{F}$ et $A_{ij,h}$, les approximations des composantes (i,j) de la matrice $\mathbf{A}$, les indicateurs d'erreurs sont donnés par

$$\eta_K^1 = \alpha_K^{-\frac{1}{2}} h_K \left\| F_{1,h} + \alpha_K \Delta u_{1,h} + (A_{11,h} u_{1,h}) + (A_{12,h} u_{2,h}) \right\|_{L^2(K)} + \frac{1}{2} \sum_{e \in \varepsilon_K} \alpha_e^{-\frac{1}{2}} h_e^{\frac{1}{2}} \left\| [\alpha \nabla u_{1,h} \cdot \mathbf{n}_e]_e \right\|_{L^2(e)} \quad (5.8)$$

et

$$\eta_K^2 = \alpha_K^{-\frac{1}{2}} h_K \left\| F_{2,h} + \alpha_K \Delta u_{2,h} + (A_{12,h} u_{1,h}) + (A_{22,h} u_{2,h}) \right\|_{L^2(K)} + \frac{1}{2} \sum_{e \in \varepsilon_K} \alpha_e^{-\frac{1}{2}} h_e^{\frac{1}{2}} \left\| [\alpha \nabla u_{2,h} \cdot \mathbf{n}_e]_e \right\|_{L^2(e)} . \quad (5.9)$$

Le choix local des paramètres de régularisation α^1 et α^2 est ensuite donné pour $\ell = 1, 2$ par la formule

$$(\alpha_K^\ell)^{n+1} = \max \left(\frac{(\alpha_K^\ell)^n}{1 + \kappa \max \left(\frac{\eta_K^\ell}{\|\eta_K^\ell\|_\infty} - \frac{\eta}{100}, 0 \right)}, \alpha_s \right) . \quad (5.10)$$

Dans ce chapitre, nous utilisons la même méthode de contrôle local du paramètre de régularisation que celle utilisée dans le chapitre des petits déplacements. Toutefois, la

méthode itérative utilisée pour la linéarisation du problème implique une implémentation différente de la méthode. Nous explicitons ci-dessous les différentes étapes de l'algorithme implémenté à l'aide de *FreeFem++*.

1. Initialisation : $\mathbf{U}^0 = 0$, $i = 0$.
2. Calcul de $f(\mathbf{x} + \mathbf{U}^i)$ et $\mathbf{A}^i$.
3. Résolution du problème (5.7) pour l'itération i.
4. Choix local du paramètre α et adaptation du maillage.
5. Mise à jour de la solution : $\mathbf{U}^{i+1} = \mathbf{U}^i + d\mathbf{U}^i$.
6. $i = i + 1$, retour à l'étape 2.

L'algorithme s'arrête lorsque l'erreur L^2 entre $\mathbf{U}^i$ et $\mathbf{U}^{i+1}$ est inférieure à 10^{-2}.

5.2 Résultats numériques

5.2.1 Evaluation du modèle

Pour l'évaluation de notre modèle d'estimation du flot optique en grands déplacements, nous utilisons une nouvelle fois des séquences proposées sur le site de Middleburry. Les cas tests considérés dans cette partie sont présentés dans le tableau 5.1. Afin de travailler sur des séquences en grands déplacements, nous utilisons des couples d'images plus espacées : les images numérotées 7 et 14.

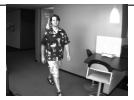
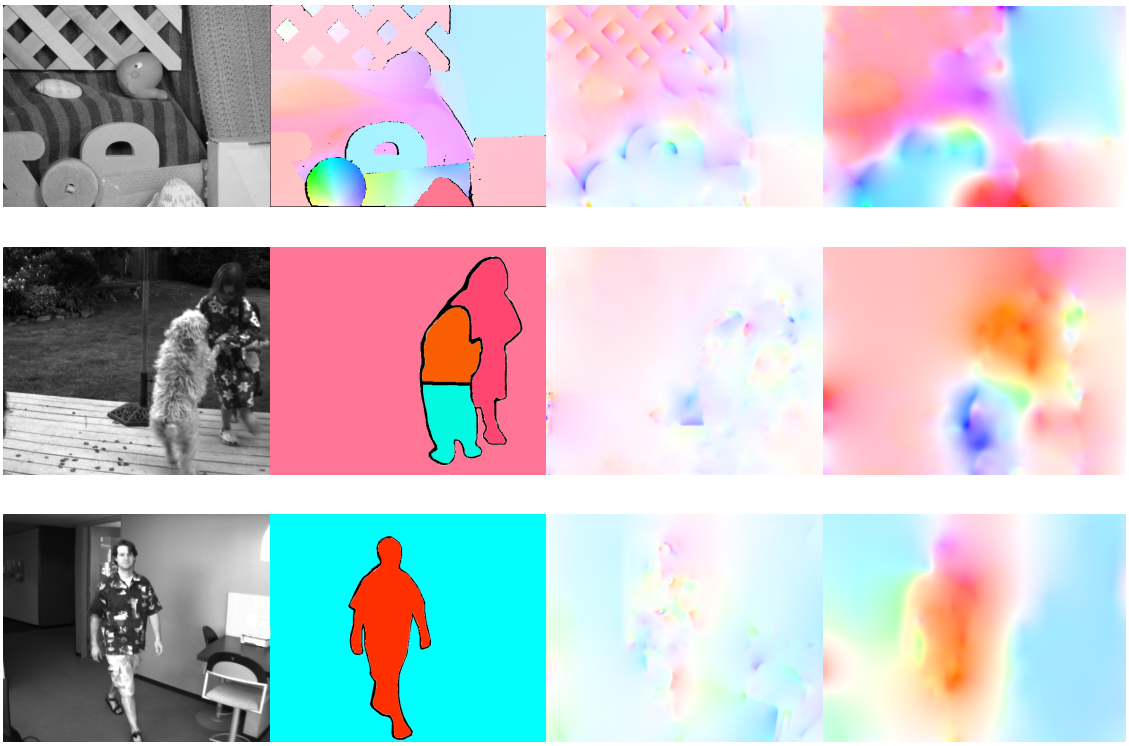
	Cas test	Spécificité
RubberWhale		Images synthétiques présentant de nombreux détails
DogDance		Photographies réelles présentant de larges occlusions
Walking		

TABLE 5.1 – Particularités des différents cas tests utilisés.

Ces séquences mettent en évidence le problème principal de l'estimation du flot optique en grands déplacements, les occlusions. De plus, la séquence *Walking* présente une variation de la luminosité causée par le rapprochement du personnage vers une source lumineuse (luminosité du personnage et ombre sur le mur).

Bien qu'il n'existe actuellement pas de séquence test disposant d'une solution exacte pour des grands déplacements, nous savons que le flot optique de la séquence *RubberWhale* est une extension de la séquence utilisée dans le cas des petits déplacements. Le mouvement est le même, il est simplement prolongé. La solution exacte recherchée reste donc très similaire. Concernant les cas tests *DogDance* et *Walking*, nous analysons visuellement les mouvements afin de créer une représentation grossière de la solution exacte. Ces solutions ne nous permettent pas d'effectuer des calculs d'erreurs mais elles permettent en revanche de vérifier la validité de l'orientation du champs de vecteurs.

Sur les résultats de la figure 5.1, nous remarquons que la méthode LUM-REGU$_{\text{loc}}$ présentée dans le chapitre 2.3 n'est pas en mesure de donner une bonne approximation du flot optique. En revanche, avec la méthode GD-LUM-REGU$_{\text{loc}}$, bien que les éléments ne soient pas définis précisément, nous obtenons une approximation proche de la solution recherchée.

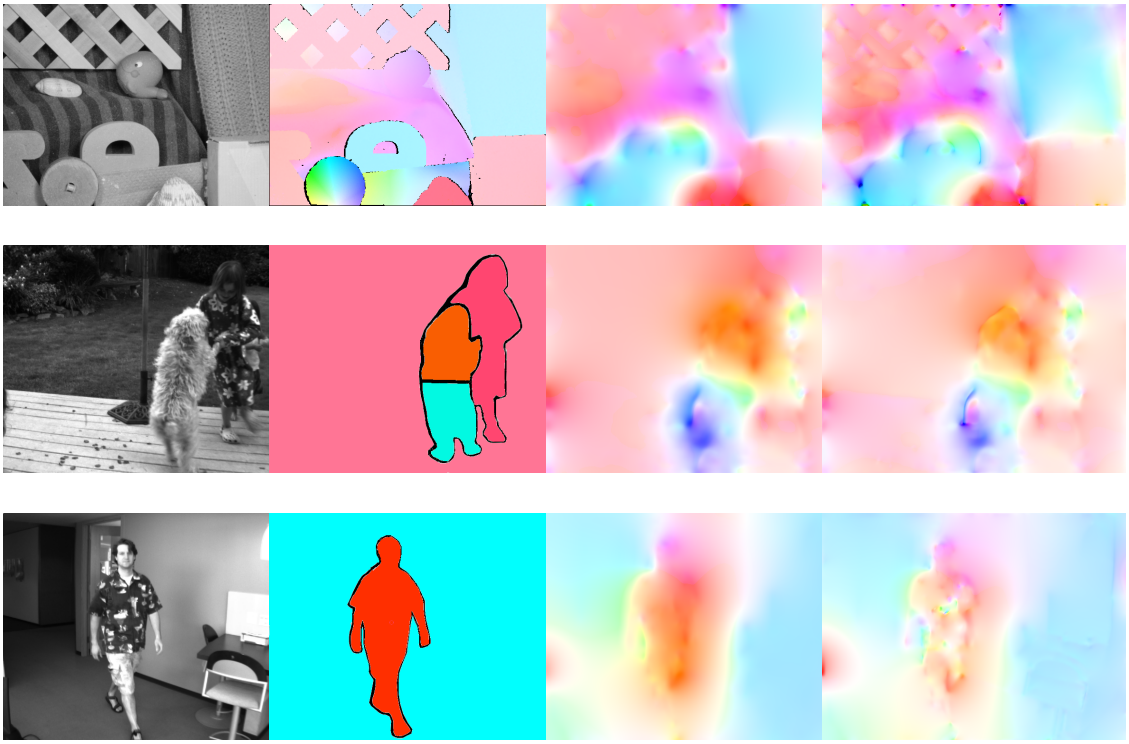


(a) Image test (b) Solution recherchée schématique (c) Algorithme des petits déplacements LUM-REGU$_{\text{loc}}$ (d) Algorithme des grands déplacements GD-LUM-REGU$_{\text{loc}}$

FIGURE 5.1 – Estimation du flot optique pour différents cas tests de grands déplacements du site de *Middleburry*, avec la méthode LUM-REGU$_{\text{loc}}$ (c) puis avec la méthode GD-LUM-REGU$_{\text{loc}}$ (d).

L'utilisation du contrôle local du paramètre de régularisation α (voir figure 5.2) nous

permet une fois encore de mieux déterminer les contours des objets et donc d'obtenir une meilleure approximation du flot optique. En revanche, sur la séquence *Walking*, les motifs de la chemise impliquent une diminution de ce paramètre donc une perte de précision de l'estimation dans cette zone. Nous remarquons également que dans les deux cas *DogDance* et *Walking*, les zones d'occlusions à gauche sont moins bien estimées.



(a) Image test (b) Solution recherchée schématique (c) Sans adaptation (d) Avec adaptation

FIGURE 5.2 – Estimation du flot optique obtenue avec (d) et sans (c) adaptation de α pour les différents cas tests du site de *Middleburry* avec l'algorithme GD-LUM-REGU$_{\text{loc}}$.

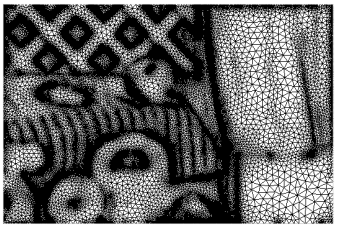
Enfin, bien que le test *RubberWhale* ne présente pas de variation de luminosité, nous montrons sur la figure 5.3, que dans le cas des grands déplacements, les zones d'occlusions agissent comme des zones où la luminosité est variable. La prise en compte de la luminosité aide ainsi à mieux traîter ces zones.

5.2.2 Adaptation du maillage

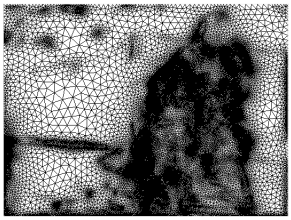
Sur la figure 5.4, nous présentons les maillages adaptés après 10 itérations d'adaptation. Ceux-ci ont bien été raffinés autour des objets en mouvement de l'image. Cependant, nous remarquons que les textures des vêtements et du pelage du chien entrainent un


FIGURE 5.3 – Estimation du flot avec les méthodes GD-REGU$_{\text{loc}}$ (gauche) et GD-LUM-REGU$_{\text{loc}}$ (droite) dans le cas des grands déplacements.

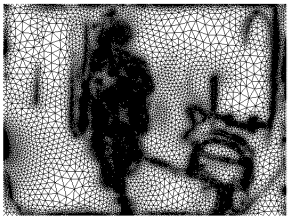
raffinement du maillage non désiré.



(a) RubberWhale



(b) DogDance



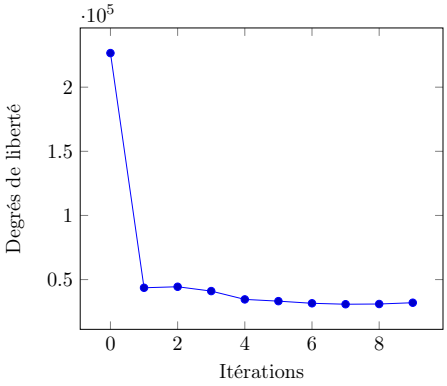
(c) Walking

FIGURE 5.4 – Maillages adaptées obtenu avec l'algorithme GD-LUM-REGU$_{\text{loc}}$.

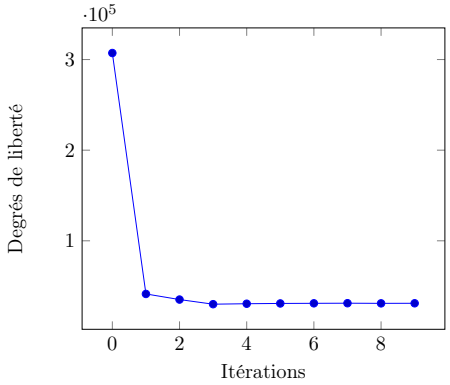
Malgré cela, comme on le voit sur la figure 5.5, le profil de diminution du nombre de degrés de liberté est le même que dans le chapitre sur les petits déplacements, il est fortement diminué dès la deuxième itération.

5.2.3 Convergence de la méthode

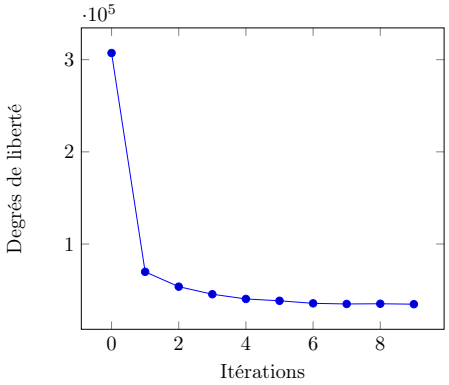
La fonctionnelle considérée dans le cas des grands déplacements n'est pas une fonction convexe. Par conséquent, nous ne pouvons pas pas prouver analytiquement la convergence de la méthode. Afin de valider numériquement la méthode, nous effectuons la simulation sur un grand nombre d'itérations afin d'observer la convergence de l'algorithme. Les



(a) RubberWhale



(b) DogDance



(c) Walking

FIGURE 5.5 – Evolution du nombre de degrés de liberté au cours des itérations d'adaptation.

erreurs causées par l'interpolation bilinéaire et les erreurs du modèle entrainent à long terme une mauvaise adaptation du maillage. Nous choisissons donc d'arrêter l'adaptation du maillage lorsque le modèle commence à diverger. Sur la figure 5.6, nous affichons l'évolution de l'erreur

$$\left\|\mathbf{U}^{n+1} - \mathbf{U}^{n}\right\|_{L^2}.$$

5.2.4 Temps d'exécution

Bien que la méthode utilisée pour la résolution du flot optique en grands déplacements soit proche de celle utilisée dans le chapitre 2.3, son implémentation FreeFem++ nécessite plusieurs modifications. Nous pouvons toutefois remarquer que, contrairement à

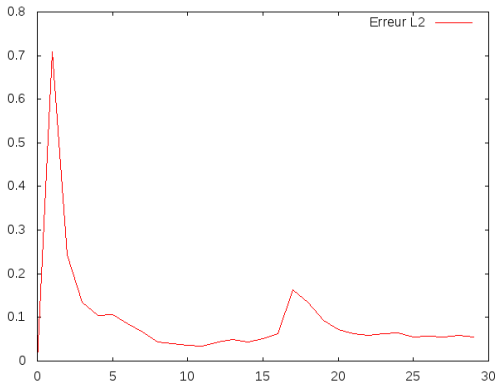


FIGURE 5.6 – Evolution de l'erreur $\left\|\mathbf{U}^{n+1} - \mathbf{U}^{n}\right\|_{L^2}$.

	LUM-REGU$_{\text{loc}}$	GD-LUM-REGU$_{\text{loc}}$
Walking	252s	339s
DogDance	247s	336s

FIGURE 5.7 – Comparaison des temps de calcul des méthodes LUM-REGU$_{\text{loc}}$ et GD-LUM-REGU$_{\text{loc}}$ pour les cas tests Walking et DogDance.

la méthode des petits déplacements, l'algorithme des grands déplacements est déjà un algorithme itératif dans sa version séquentielle. Comme nous l'avons fait dans le cas de la méthode utilisant le rapprochement itératif, nous pouvons combiner ces itérations ainsi que celles des étapes d'adaptation du paramètre α. Malgré cela, l'algorithme reste plus coûteux en temps de calcul par rapport à la méthode des petits déplacements en raison des étapes d'interpolations bilinéaires et de lissages gaussiens supplémentaires. Dans cette section, nous comparons les temps de calcul entre les méthodes LUM-REGU$_{\text{loc}}$ et GD-LUM-REGU$_{\text{loc}}$. Dans le tableau 5.7, nous donnons les temps d'exécution des deux méthodes pour chacun des deux tests utilisés dans cette partie. Nous effectuons pour chacune des méthodes 4 itérations.

En étudiant en détail le temps des calculs de la méthode GD-LUM-REGU$_{\text{loc}}$, nous observons que le temps nécessaire pour effectuer l'interpolation bilinéaire pour ces tests est en moyenne de 50 secondes. Le temps d'exécution des lissages gaussiens est également plus important que dans le cas des petits déplacements. Compte tenu de la structure des implémentations, aucune optimisation n'est a priori possible pour réduire le temps de calcul de la convolution. Ainsi, nous constatons que la méthode GD-LUM-REGU$_{\text{loc}}$ traitant les grands déplacements ainsi que le contrôle local du paramètre de régularisation α demande environ 35% de temps supplémentaire que la méthode LUM-REGU$_{\text{loc}}$ dans notre cas.

5.3 Conclusion

Dans ce chapitre, nous avons adapté au cas des grands déplacements les méthodes étudiées dans le chapitre 2.3. Bien que le problème soit différent puisque l'hypothèse fondamentale du flot optique n'est plus applicable, nous avons pu résoudre la non linéarité à l'aide d'une méthode itérative de point fixe. Afin d'obtenir un meilleur traitement des zones d'occlusions et des variations de luminosité, la prise en compte de la luminosité a également été ajoutée à notre modèle. Nous avons pu voir que l'implémentation de la méthode de contrôle adaptatif du paramètre α est transposable à la méthode GD-LUM-REGU$_{\text{loc}}$ et qu'elle nous permet d'obtenir les mêmes améliorations sur les bords des objets. Bien que moins rapide que la méthode LUM-REGU$_{\text{loc}}$ à cause d'un nombre plus important de convolutions gaussiennes à effectuer, l'algorithme GD-LUM-REGU$_{\text{loc}}$ regroupe les itérations permettant la linéarisation du flot optique avec celles concernant l'adaptation de la régularisation, rendant l'influence du contrôle local de la régularisation et l'adaptation du maillage négligeables en termes de temps de calcul.

Chapitre 6

Méthodes parallèles pour l'estimation du flot optique en grands déplacements

Sommaire

Dans le chapitre précédent, nous avons etudié des méthodes d'estimation du flot optique pour des séquences en larges déplacements. Nous avons constaté que bien que nous ayons pu intégrer l'étape de contrôle local de la régularisation dans le calcul itératif de la méthode de point fixe, le temps de calcul de l'algorithme GD-LUM-REGU$_{\text{loc}}$ est environ 35% suppérieur à celui de la méthode LUM-REGU$_{\text{loc}}$. Dans ce chapitre, nous reprenons les méthodes de calcul parallèle utilisées dans le chapitre 4 afin de les implémenter dans le cas de l'estimation de mouvements en grands déplacements et ainsi réduire le temps d'exécution du programme.

6.1 Méthode de décomposition de domaine

Nous implémentons la méthode de décomposition de domaine (que l'on appelera GD-DDM) dans le cas de l'algorithme GD-LUM-REGU$_{\text{loc}}$ en effectuant un découpage du domaine Ω similaire à celui du chapitre 4. Nous effectuons les tests de temps en utilisant les images 7 et 14 de la séquence *RubberWhale*. Pour toutes les configurations, nous faisons quatre itérations de la méthode de Schwarz et donc quatre itérations d'adaptation du paramètre α. Sur la figure 6.1, nous donnons les temps obtenus. Nous détaillons sur ce graphique les temps d'exécution des principales parties de l'algorithme, c'est à dire la construction de la matrice de masse du système, le conditionnement de cette matrice, les communications et les convolutions.

Comme nous l'avons dit dans le chapitre précédent, la méthode GD-LUM-REGU$_{\text{loc}}$ nécessite des étapes d'interpolation et de lissage supplémentaires. La convolution peut être effectuée en parallèle sur chaque sous-domaine, mais l'étape d'interpolation doit être réalisée de manière séquentielle sur la totalité de l'image. Cela a pour conséquence de diminuer la scalabilité de la méthode. Cependant, les résultats présentés sur la figure 6.1 témoignent de l'efficacité de la méthode de décomposition de domaine.

Sur la figure 6.2, nous pouvons constater qu'après quatre itérations la solution à l'interface des sous-parties a été correctement fusionnée grâce à la méthode de Schwarz.

6.2 Méthode pararéelle

Nous nous proposons dans cette section de nous servir des avantages de la méthode pararéelle afin de proposer différents schémas pour les résolutions grossière et fine nous permettant, sous certaines hypothèses, de nous défaire du problème de la non-linéarité posé par les grands déplacements. En effet, le pararéel proposant un algorithme visant à résoudre deux schémas sur deux échelles, grossier et fin, nous pouvons ainsi tourner à notre avantage la partie fine en proposant par exemple une résolution principalement explicite ou encore rappeler l'algorithme des petits déplacements pour cette partie, la partition du temps étant assez fine pour considérer les sous-résolutions fines comme des petits déplacements.

Afin d'utiliser la méthode pararéelle, nous commençons par ajouter le terme de variation temporelle. Le problème (5.5) avec la variation en temps s'écrit sous forme vectorielle

$$\frac{d\mathbf{U}}{dt} - \operatorname{div}(\mathbf{\Lambda}(\mathbf{x})\nabla\mathbf{U}) + (\mathbf{f}(\mathbf{x}+\mathbf{U}) - M\mathbf{f}(\mathbf{x}))\tilde{\mathbf{f}}(\mathbf{x}+\mathbf{U}) = 0 \tag{6.1}$$

avec

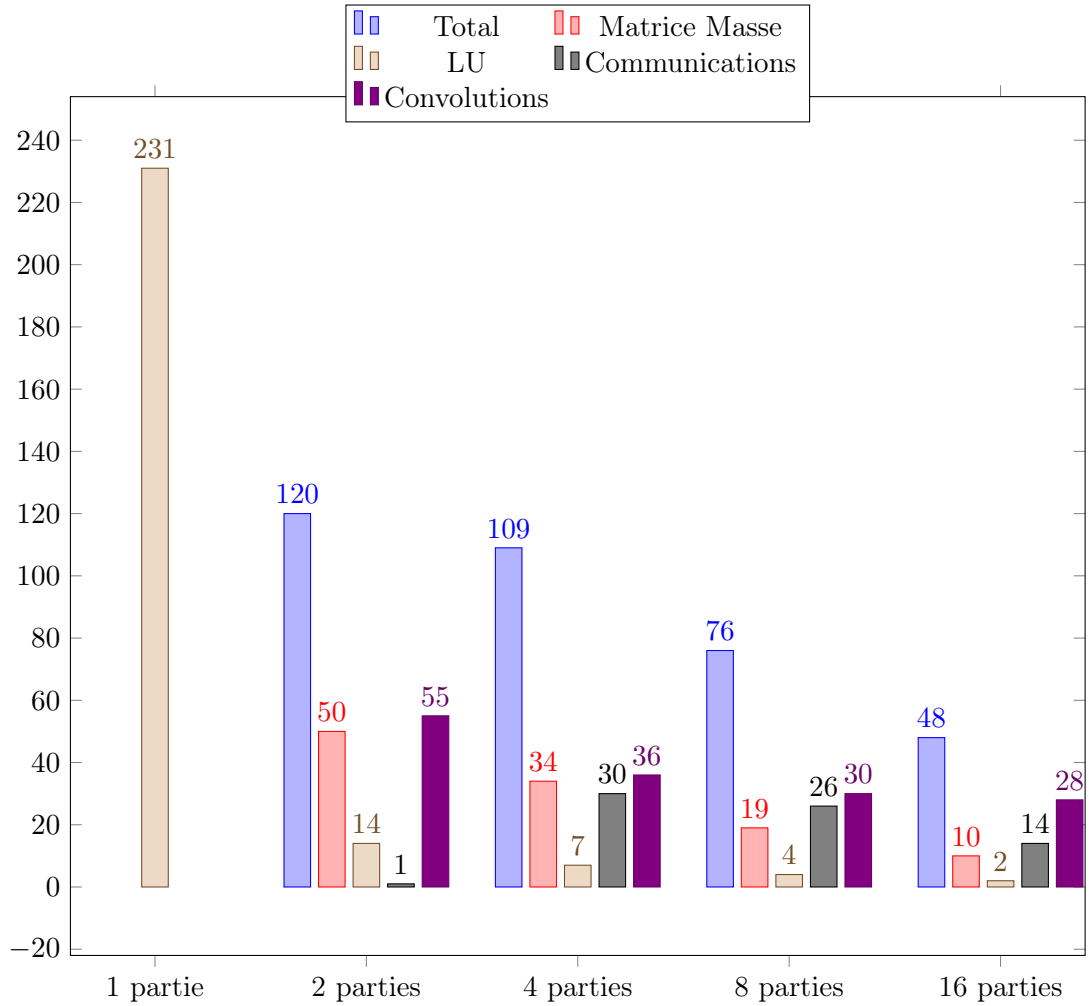


FIGURE 6.1 – Temps de calcul détaillés de la décomposition de domaine GD-DDM dans le cas des grands déplacements pour 4 itérations de la méthode GD-LUM-REGU$_{\text{loc}}$ et 2, 4, 8 et 16 séparations.

$$\tilde{\mathbf{f}}(\mathbf{x}+\mathbf{U}) = \begin{pmatrix} f_x(x+u_1, y+u_2, t+1) \\ f_y(x+u_1, y+u_2, t+1) \\ f(x,y,t) \end{pmatrix}.$$

Afin de décrire la méthode, nous commençons par considérer les paramètres de régularisation α et λ constants. Soit $0 = T_0 < T_1 < \cdots < T_N = T$ une subdivision de l'intervalle de temps $[0, T]$, nous définissons un solveur grossier

FIGURE 6.2 – Solution obtenue avec la méthode de décomposition de domaine pour les grands déplacements GD-DDM après 4 itérations de la méthode de Schwarz.

$$\begin{cases} \mathbf{U}^g_{n+1} = \mathcal{G}(\mathbf{U}^g_n), & n = 0, \dots, N-1 \\ \mathbf{U}^g_0 = \mathbf{U}^g(T_0 = 0) = 0. \end{cases}$$

Chacune des solutions grossières $\mathbf{U}^g_n$ obtenues est ensuite utilisée comme état initial de la résolution fine du problème (6.1) sur l'intervalle $[T_n, T_{n+1}]$ correspondant. Ainsi, le problème fin à résoudre par le processeur n sur le sous-intervalle $[T_n, T_{n+1}]$ est donné par

$$\begin{cases} \mathbf{U}^f_{n,m+1} = \mathcal{F}(\mathbf{U}^f_{n,m}), & m = 0, \dots, M-1 \\ \mathbf{U}^f_{n,0} = \mathbf{U}^g_n. \end{cases}$$

Nous appliquons ensuite l'étape de mise à jour de la solution en utilisant la prédiction-correction présentée dans le chapitre 2.

Pour les schémas grossiers et fins, nous choisissons de tester différentes méthodes afin de déterminer quelle discrétisation représente le meilleur choix pour le terme non-linéaire.

6.2.1 Discrétisation semi-implicite

Dans un premier temps, nous choisissons le même schéma pour le solveur grossier et le solveur fin, une discrétisation semi-implicite. Dans le chapitre précédent, nous avons décrit ce schéma dans le cas de l'équation (5.5). Nous reprenons ici le même schéma d'itération de point fixe afin de linéariser le problème. En reprenant les notations de (5.7), le problème (6.1) s'écrit dans le cas du solveur grossier

$$\frac{d\mathbf{U}^n}{\Delta T} - \operatorname{div}(\mathbf{\Lambda}(\mathbf{x})\nabla d\mathbf{U}^n) + \mathbf{A}^n d\mathbf{U}^n = \operatorname{div}(\mathbf{\Lambda}(\mathbf{x})\nabla \mathbf{U}^n) + \mathbf{F}^n. \tag{6.2}$$

De même, pour le solveur fin, sur chaque intervalle grossier $[T_n, T_{n+1}]$ de taille ΔT nous résolvons le problème

$$\frac{d\mathbf{U}^m}{\Delta t} - \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla d\mathbf{U}^m) + \mathbf{A}^m d\mathbf{U}^m = \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla \mathbf{U}^m) + \mathbf{F}^m. \tag{6.3}$$

avec $\Delta t < \Delta T$. Dans la suite, nous nommerons cet algorithme GDSI.

6.2.2 Discrétisation explicite de la non-linéarité pour le solveur fin

Dans un deuxième temps, nous proposons de garder le schéma semi-implicite pour le solveur grossier

$$\frac{d\mathbf{U}^n}{\Delta T} - \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla d\mathbf{U}^n) + \mathbf{A}^n d\mathbf{U}^n = \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla \mathbf{U}^n) + \mathbf{F}^n$$

et de discrétiser le terme non-linéaire avec une méthode explicite pour la résolution fine. Pour cela, sur chaque intervalle grossier $[T_n, T_{n+1}]$, nous résolvons le système linéarisé

$$\left\{ \begin{array}{l} -\mathrm{div}(\alpha(\mathbf{x})\nabla u_1^{k+1}) + (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t)) f_x(x+u_1^k, y+u_2^k, t+1) = 0 \\ -\mathrm{div}(\alpha(\mathbf{x})\nabla u_2^{k+1}) + (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t)) f_y(x+u_1^k, y+u_2^k, t+1) = 0 \\ -\mathrm{div}(\lambda(\mathbf{x})\nabla M^{k+1}) - (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t)) f(x,y,t) = 0. \end{array} \right.$$

Après avoir ajouté et discrétisé le terme temporel, nous obtenons le problème linéaire pour la résolution fine

$$\frac{\mathbf{U}^{m+1} - \mathbf{U}^m}{\Delta t} - \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla \mathbf{U}^{m+1}) = \mathbf{F}^m \tag{6.4}$$

où

$$\mathbf{F}^m = \begin{pmatrix} -(f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t)) f_x(x+u_1^k, y+u_2^k, t+1) \\ -(f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t)) f_y(x+u_1^k, y+u_2^k, t+1) \\ (f(x+u_1^k, y+u_2^k, t+1) - M^k f(x,y,t)) f(x,y,t) \end{pmatrix}.$$

Cette méthode sera appelée GDEx.

6.2.3 Appel de la résolution en petits déplacements pour le solveur fin

Nous proposons finalement de considérer la résolution fine comme une résolution en petits déplacements. Pour la résolution grossière, nous gardons la résolution semi-implicite initiale du problème

$$\frac{d\mathbf{U}^n}{\Delta T} - \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla d\mathbf{U}^n) + \mathbf{A}^n d\mathbf{U}^n = \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla \mathbf{U}^n) + \mathbf{F}^n.$$

Pour la résolution fine de chaque intervalle grossier $[T_n, T_{n+1}]$, nous supposons le pas assez petit pour considérer la fonction f de classe $\mathcal{C}^1$ et ainsi, nous placer dans le cas des petits déplacements. Dans ce cas, le problème s'écrit

$$\frac{\mathbf{U}^{m+1} - \mathbf{U}^m}{\Delta t} - \mathrm{div}(\mathbf{\Lambda}(\mathbf{x})\nabla \mathbf{U}^{m+1}) + \mathbf{J}_\rho \mathbf{U}^{m+1} = \mathbf{F}^m. \tag{6.5}$$

Cette méthode sera notée GDPD.

6.2.4 Adaptation du paramètre α pour le solveur grossier

L'application de la méthode de contrôle local du paramètre de régularisation α et l'adaptation du maillage étant faites a posteriori et indépendemment du schéma choisi, nous appliquons la même méthode que celle présentée dans le chapitre 2.3. Nous proposons d'effectuer les étapes d'adaptation que dans le cas de la résolution grossière. Le nouveau maillage ainsi que la fonction $\alpha(\mathbf{x})$ sont communiqués entre les processeurs en même temps que les solutions grossières du problème.

6.2.5 Résultats numériques

Nous souhaitons tester et comparer l'efficacité des différents schémas présentés précédemment. Pour cela, nous reprenons le cas test *RubberWhale* de Middleburry en considérant les images 7 et 14 de la séquence afin d'obtenir un large déplacement.

Sur la figure 6.3, nous donnons les solutions obtenues en traitant la non-linéarité du solveur fin avec le schéma semi-implicite (GDSI), puis avec le schéma explicite (GDEx) et enfin, en utilisant le schéma reprenant la méthode des petits déplacements (GDPD).

Nous remarquons sur la figure 6.3 que, bien que les solutions obtenues à l'aide des méthodes GDEx et GDPD présentent certaines arêtes bien définies, la solution globale est

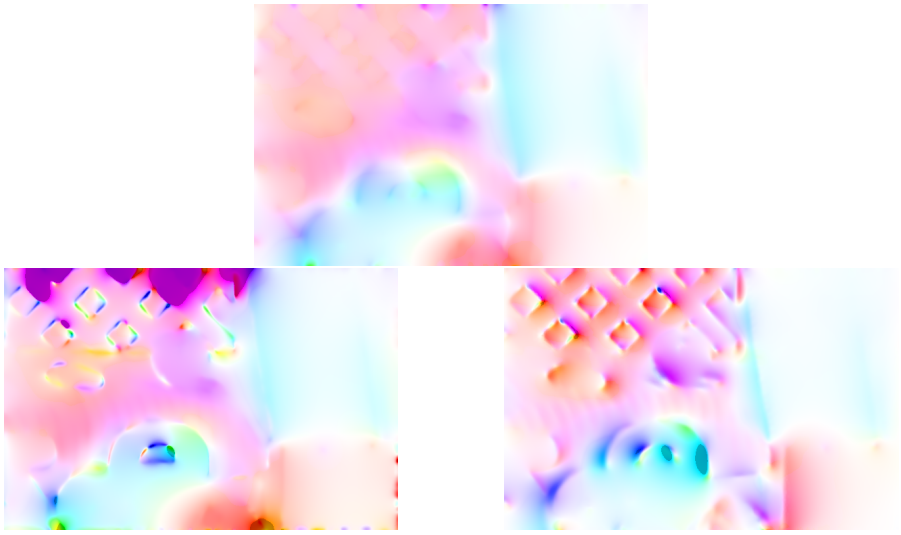


FIGURE 6.3 – Estimation du flot optique pour une séquence de larges déplacements avec, en haut, les résultats du schéma GDSI et en bas, les méthodes GDEx (gauche) et GDPD (droite).

de meilleure qualité avec la méthode semi-implicite GDSI. Pour montrer l'effet de l'adaptation, nous donnons le résultat de GDSI sans stratégie adaptative du paramètre α sur la figure 6.4. Nous remarquons ainsi que l'adaptation du maillage et du paramètre α ont eu, ici encore, un effet positif sur l'estimation de la solution au niveau des arêtes de l'image.

FIGURE 6.4 – Estimation du flot optique pour une séquence de larges déplacements avec la méthode GDSI sans adaptation.

6.2.5.1 Temps d'exécution

Dans le tableau 6.5, nous comparons les temps des méthodes GDSI, GDEx et GDPD testées en utilisant quatre processeurs. Nous effectuons quatre itérations grossières et quatre itérations de la résolution fine. La résolution du problème du flot optique pour les grands déplacements se fait dans notre exemple avec quatre itérations de point fixe. Nous

Méthode	Temps (s)
Seq (6.1)	1955
GDSI (6.3)	1774
GDEx (6.4)	1534
GDPD (6.5)	1300

FIGURE 6.5 – Comparaison des temps de calcul pour les méthodes GDSI, GDEx et GDPD par rapport à un algorithme de référence séquentiel (Seq).

comparons ces résultats avec un algorithme de grands déplacements séquentiel effectuant un total de 16 pas de temps et de quatre itérations de point fixe.

6.3 Couplage entre la décomposition de domaine et la méthode pararéelle

Nous avons vu dans la section précédente que la méthode pararéelle nous permettait d'obtenir de bons résultats mais que c'était également une méthode dont les temps de calculs pouvaient être longs. Nous allons donc dans cette section utiliser un couplage entre la méthode pararéelle et la méthode de décomposition de domaine (appelé GD-DDM-Para). Nous employons la même stratégie que celle utilisée dans le chapitre 4, nous attribuons plusieurs processeurs à chaque sous-domaine. La même méthode de distribution des processeurs entre les différents sous-domaines est utilisée, le CPU_i est attribué au groupe i grâce à la formule

$$\mathrm{groupe}(\mathrm{CPU}_i) = \frac{\mathrm{CPU}_i}{\mathrm{nbPart}}.$$

A l'intérieur d'un sous-domaine, la numérotation des processeurs reste identique à celle de la méthode pararéelle seule.

Sur la figure 6.6, nous présentons la solution obtenue à l'aide de ce couplage. Nous utilisons pour ces tests la méthode GDSI. Nous constatons que la solution hérite de la bonne régularité de la solution du problème par la méthode pararéelle et d'une bonne estimation sur les arêtes de l'image dûe au traitement par sous-domaine du contrôle local de la régularisation.

En plus de l'amélioration de l'estimation de la solution, la figure 6.7 montre que ce couplage entre les deux méthodes parallèles utilisées dans ce chapitre permet d'obtenir une réduction du temps de calcul de la méthode GDSI.

FIGURE 6.6 – Estimation du flot optique obtenue avec le couplage de la méthode parareelle et de la méthode de décomposition de domaine en larges déplacements GD-DDM-Para.

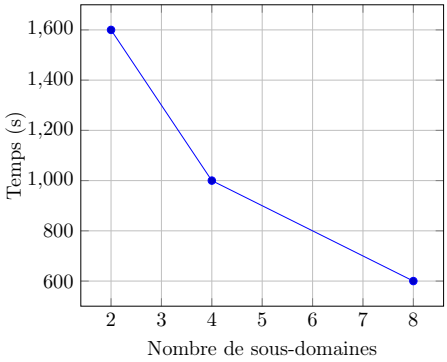


FIGURE 6.7 – Temps d'exécution de calcul du flot optique par la méthode GD-DDM-Para en fonction du nombre de sous-domaines.

Sur le tableau 6.8, nous proposons une synthèse des différentes méthodes parallèles utilisées pour les grands déplacements.

Nous pouvons ainsi conclure que si le but principal est de gagner du temps, la méthode la plus efficace est la décomposition de domaine seule (DDM-GD). Dans le cas où l'on souhaite tester l'efficacité de plusieurs schémas et utiliser les avantages que propose la séparation des deux solveurs grossier et fin afin d'améliorer la précision, nous nous tournerons vers le couplage du pararéel et de la décomposition de domaine. En effet, comme nous l'avons vu, le pararéel est un algorithme permettant d'utiliser plusieurs schémas mais également de modifier le modèle grâce à ce couplage entre résolution grossière et résolution fine. Cependant, utiliser le pararéel pour le problème du flot optique s'est révélé couteux en termes de temps de calcul. Pour améliorer ce point, un couplage avec la méthode de décomposition de domaine représente une bonne option.

Méthode	Description	Temps (s)
GD-DDM (6.1)	Méthode de décomposition de domaine (4 parties)	109
GDSI (6.2.1)	Méthode pararéelle. Décomposition semi-implicite pour le solveur grossier et pour le solveur fin (4 itérations grossières et 4 itérations fines)	1774
GDEx (6.2.2)	Méthode pararéelle. Décomposition semi-implicite pour le solveur grossier et explicite pour le solveur fin (4 itérations grossières et 4 itérations fines)	1534
GDPD (6.2.3)	Méthode pararéelle. Décomposition semi-implicite pour le solveur grossier et estimation du flot optique en petits déplacements pour le solveur fin (4 itérations grossières et 4 itérations fines)	1300
GD-DDM-Para (6.3)	Couplage de la méthode GD-DDM et de la méthode pararéelle GDSI (4 parties, 4 itérations grossières et 4 itérations fines).	600

FIGURE 6.8 – Comparaison des temps de calcul des différentes méthodes parallèles utilisées en grands déplacements pour le test de RubberWhale.

6.4 Conclusion

Dans ce chapitre, nous avons implémenté la méthode de décomposition de domaine dans le cas des grands déplacements afin de réduire les temps de calcul de la méthode GD-LUM-REGU$_{\text{loc}}$. Bien que notre modèle ne soit pas scalable à cause de certain facteurs tels que l'étape d'interpolation bilinéaire, l'algorithme est efficace et permet de diminuer les temps de calcul d'un facteur 2 en partageant l'image entre deux processeurs ou d'un facteur 5 en la décomposant entre 16 processeurs.
Nous avons ensuite implémenté la méthode pararéelle étudiée dans le chapitre 4 et comparé différents schémas de discrétisation de la non-linéarité pour le solveur fin. En plus de la discrétisation semi-implicite précédemment étudiée, nous avons utilisé une discrétisation explicite puis une méthode faisant intervenir le modèle développé pour le problème en petits déplacements. Après avoir ajouté l'adaptation de la régularisation et du maillage, nous avons pu améliorer la précision des résultats. Enfin, comme nous l'avions fait dans le chapitre 4, nous avons proposé un couplage entre la méthode de décomposition de domaine et la méthode pararéelle. Ce couplage nous a permis de garder la souplesse offerte par la méthode pararéelle tout en réduisant les temps d'exécution de l'algorithme grâce à la méthode de décomposition de domaine.

Chapitre 7

Application du flot optique

Sommaire

7.1 Restauration de films

Dans le domaine du traitement d'images, il existe une catégorie concernant la restauration d'images. Cette branche comporte de nombreuses applications dans la restauration de séquences vidéos abîmées, la retouche de photos ou la décompression d'images dans le cas où des blocs seraient perdus lors de la compression par exemple. Nous n'allons pas détailler ici les méthodes utilisées pour réaliser ces restaurations. Dans un premier temps, nous présentons de manière synthétique les équations de Ginzburg-Landau utilisées dans le domaine de l'inpainting puis nous verrons par quel moyen nous pouvons utiliser l'estimation du flot optique dans cette branche de l'imagerie.

7.1.1 Les équations de Ginzburg-Landau

Les équations de Ginzburg-Landau sont à l'origine des équations créées pour la physique des supraconducteurs [55, 56]. Pour une fonction $u(\mathbf{x})$, $\mathbf{x} \in \Omega$ à valeurs dans $[0, 1]$, on veut alors résoudre

$$\Delta u + \frac{1}{\varepsilon^2} F'(u) = 0$$

où ε est un réel positif et où $F(u) = (1 - \mid u^2 \mid)^2$.

L'utilisation de ces équations dans le domaine de la restauration d'image est détaillée dans un article de Harald Grossauer [60] dans lequel il utilise une version complexe de ces équations. Pour cela, il construit une fonction complexe g à partir de la fonction f définissant l'intensité des pixels de l'image à restaurer

$$\begin{cases} \Re(g) = f \\ \Im(g) = \sqrt{(1 - f^2)}. \end{cases}$$

Les équations de Ginzburg-Landau complexes modifiées pour l'application à la restauration d'image consistent à résoudre le système

$$\begin{cases} \frac{\partial v}{\partial t} - \Delta v + \frac{1}{\varepsilon^2}(1 - \mid v \mid^2)^2 + (v - g) = 0 \text{ in } \Sigma, \quad t \in [0, T] \\ \frac{\partial v}{\partial n} = 0 \end{cases} \tag{7.1}$$

où v est la solution complexe à déterminer et où $\Sigma \subset \Omega$ représente le sous-ensemble de Ω où l'image à besoin d'être restaurée. Pour plus de détails sur ces équations nous renvoyons le lecteur vers [55, 56, 60, 112].

7.1.2 Utilisation du flot optique

Afin d'utiliser les équations de Ginzburg-Landau (7.1), nous devons préalablement déterminer le domaine Σ où la restauration doit avoir lieu. Cette zone, que nous appellerons dans la suite le masque de dégradation peut, dans le cas d'une séquence d'au moins trois images, être déterminée à partir d'une estimation de flots optiques. L'utilisation d'un calcul de flot optique dans la détermination du masque de dégradation provient de Harald Grossauer [59]. Nous avons implémenté cette méthode dans le cas de notre estimation de flot optique avec contrôle du paramètre de régularisation. L'idée consiste, à partir de trois images successives g_{n-1}, g_n et g_{n+1} dans une séquence, à déterminer le masque de dégradation de l'image g_n.

Pour cela, on commence par déterminer les flots optiques h_b et h_f liés aux images f_{n-1} et f_{n+1}. On a donc les approximations

$$g_{n-1}(\mathbf{x} + h_b) \approx g_n(\mathbf{x}) \approx g_{n+1}(\mathbf{x} + h_f).$$

En effet, le champ de vecteurs h_b transporte les pixels de g_{n-1} vers g_n et de la même manière h_f transposte les pixels de g_{n+1} vers g_n. En évaluant les différences $|g_{n-1}(\mathbf{x}+h_b) - g_n(\mathbf{x})|$ et $|g_{n-1}(\mathbf{x}+h_b) - g_n(\mathbf{x})|$, on peut déterminer quelles sont les différences entre les images. Afin de s'assurer que la différence observée est bien une zone dégradée de l'image g_n et non un objet apparaissant ou disparaissant de la séquence, on considère qu'un pixel fait partie du masque de dégradation si pour un $\varepsilon > 0$ fixé, on a

$$|g_{n-1}(\mathbf{x}+h_b) - g_n(\mathbf{x})| > \varepsilon \text{ et } |g_{n-1}(\mathbf{x}+|h_b) - g_n(\mathbf{x})| > \varepsilon.$$

Afin d'expliquer de manière plus simple le procédé, on rappelle dans la figure 7.1 les principales étapes de la méthode pour déterminer le masque de dégradation. Sur cet exemple, on traite une séquence de trois images successives représentant un sapin se déplacant de la gauche vers la droite. Les images 2 et 3 présentent chacune des dégradations. L'application des flots optiques nous permet d'obtenir les images présentent sur la deuxième ligne. Nous comparons alors l'image à restaurer avec ces estimations. Nous présentons sur la troisième ligne les résultats de ces comparaisons. On remarque qu'il y a une seule dégradation identique sur les deux différences. Cette dégradation constitue donc le masque de dégradation de l'image g_n.

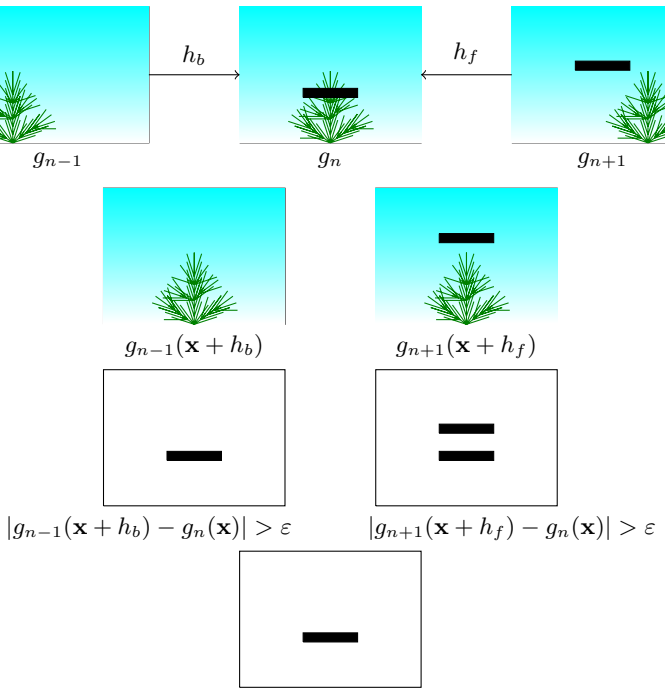


FIGURE 7.1 – Exemple simplifié de la méthode de détermination du masque de dégradation à partir d'une estimation de flots optiques.

7.1.3 Résultats numériques

Dans cette partie, nous présentons un résultat obtenu à partir d'une séquence de trois images. Nous avons, pour cela, considéré la séquence *Venus* donnée par le site de Middelburry `www.vision.middleburry.edu/flow/`. Nous avons sélectionné trois images de cette séquence et nous avons inséré artificiellement une dégradation sur l'image centrale (figure 7.2).

FIGURE 7.2 – Image 11 de la séquence de Venus à laquelle nous avons ajouté une dégradation.

L'utilisation de la méthode de détermination du masque à l'aide des flots optiques nous permet de retrouver le masque présenté sur la figure 7.3. Comme nous pouvons le remarquer, le masque obtenu dépend fortement de la précision de l'estimation du flot optique puisque des zones où il est estimé grossièrement apparaissent alors dans le masque. Sur notre exemple, ces zones sont principalement concentrées sur les bords des objets.

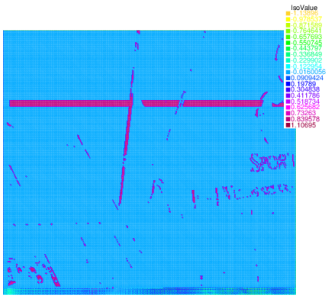


FIGURE 7.3 – Masque de la dégradation obtenu par estimations de flots optiques.

Une fois le masque obtenu, il nous reste alors à résoudre le problème de Ginzburg-Landau afin de reconstituer l'image sous le masque. Le résultat est présenté dans la figure 7.4.

FIGURE 7.4 – Résultat après reconstitution de l'image à partir masque obtenu.

7.1.4 Parallélisation

Comme nous l'avons fait dans la section 4.2, il est possible d'utiliser une parallélisation sans communication pour ce type de structure. Pour cela, on procède d'une manière analogue à celle de la section 4.2. On distribue cette fois-ci les images par paquets de trois, suivant le schéma présenté sur la figure 7.5.

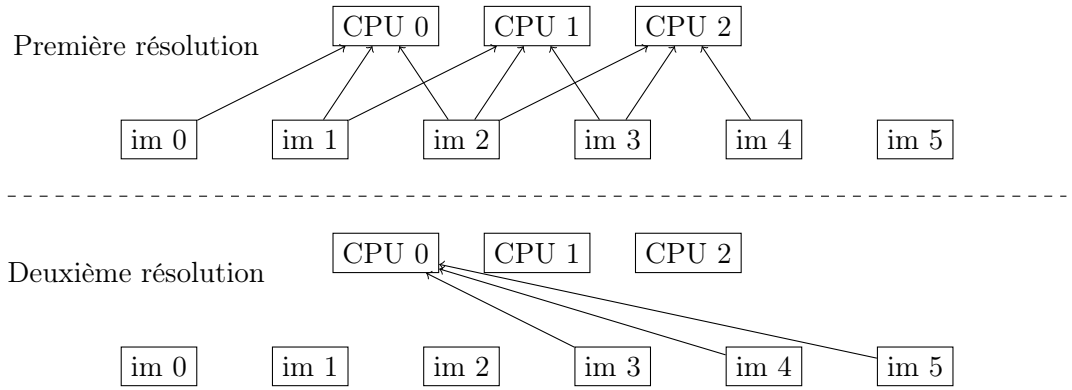


FIGURE 7.5 – Répartition des tâches pour la restauration de quatre images prisent dans une séquence avec trois processeurs.

L'accélération théorique est déterminée de la même manière que dans le cas de la parallélisation d'une séquence. Elle est cette fois-ci donnée par

$$\text{Speedup}_{th} = \text{(nbImages-2)} \cdot \left\lceil \frac{\text{nbImages} - 2}{\text{nbCPU}} \right\rceil^{-1}.$$

7.2 Conclusion

Dans ce dernier chapitre, nous avons présenté un exemple d'application de l'estimation du flot optique. En utilisant un algorithme proposé par H. Grossauer permettant de déterminer les zones dégradées d'une image d'un film à l'aide du flot optique, nous avons pu définir un masque de dégradation. L'implémentation des équations de Ginzburg-Landau a ensuite permis, à partir de ce masque, de reconstruire les parties altérées des différentes images. Sur un exemple concret, nous avons ainsi pu reconstruire les zones manquantes sur une image d'une séquence vidéo. Un algorithme de parallélisation de séquence a ensuite été combiné à cette approche afin de réaliser le travail en simultané pour plusieurs images.

Conclusion et perspectives

Dans ce manuscrit, nous avons étudié différentes méthodes de résolution pour l'estimation du flot optique. Nous avons dans un premier temps développé des modèles numériques permettant de déterminer une solution au problème en petits déplacements. Une étude concernant des implémentations parallèles a ensuite été menée. En effet, dans le cas du problème de l'estimation du flot optique, sa résolution peut être très coûteuse en temps de calcul d'autant plus si la résolution des images est importantes. Dans une seconde partie, les méthodes utilisées dans les premiers chapitres ont été appliquées dans le cas plus complexe des grands déplacements.

Dans le chapitre 1, nous avons présenté un état de l'art des principales méthodes variationnelles utilisées actuellement pour la résolution du problème du flot optique. En particulier, nous avons décrit la méthode locale de Lucas et Kanade [80], la méthode globale de Horn et Schunck [69] ainsi que la combinaison de ces deux méthodes proposée par Bruhn et Weickert [31]. Nous avons ensuite présenté une stratégie de contrôle local de la régularisation proposée par Belhachmi et Hecht [15] permettant une amélioration de la solution au niveau des bords des objets de l'image. Suite à cela, nous avons présenté le problème du flot optique dans le cas d'une luminosité variable et nous avons exposé deux modèles existant pour la prise en compte de cette nouvelle problématique. Enfin, nous avons conclu le chapitre par une courte présentation du problème du flot optique en grands déplacements.

Dans le chapitre 2, nous avons introduit les principales méthodes de parallélisation utilisées dans ce travail : la méthode de décomposition de domaine et la méthode moins connue de parallélisation en temps, le pararéel.

Le coeur de notre travail débute au chapitre 3 dans lequel nous avons mis en place des modèles d'éléments finis pour la résolution du flot optique en petits déplacements. Après avoir implémenté la méthode classique considérant la constance de la luminosité, nous nous sommes intéressés à un modèle prenant en compte la variation de l'intensité lumineuse. L'adaptation locale du paramètre de régularisation a ensuite été ajoutée aux programmes. Nous avons alors proposé une stratégie de contrôle adaptatif concernant le paramètre supplémentaire ajouté dans le cadre du traitement de la luminosité, en

suivant le même esprit que la méthode adaptative initiale. Cette stratégie a été couplée à l'adaptation du maillage, possible grâce à l'utilisation des éléments finis et du logiciel FreeFem++. Une étude analytique utilisant la Γ-convergence nous a permis de montrer l'existence d'une solution au problème. A l'aide des évaluations numériques, nous avons pu constater dans un premier temps que la modification du modèle initial entraîne une réelle amélioration du résultat dans le cas où la séquence présente une variation de la lumière. Nous avons ensuite observé l'effet du contrôle local de la régularisation. Comme souhaité, ce dernier a permis d'obtenir une nette amélioration de la solution sur les arêtes de l'image. Concernant l'adaptation du maillage, nous avons pu constater que le nombre de degrés de liberté était fortement réduit dès la première itération ce qui a permis d'obtenir une première amélioration du temps de calcul. Dans ce même chapitre, nous avons ensuite étudié l'effet d'un algorithme de rapprochement itératif couplé à une méthode de flous gaussiens appelée coarse to fine. Cette méthode a permis d'obtenir une bonne amélioration de la précision du résultat. Cependant, bien que le processus itératif soit entièrement intégré aux itérations du contrôle local de la régularisation, cette méthode entraîne un coût bien supérieur en termes de temps de calcul en raison d'un nombre plus important de convolutions à effectuer. Une grande partie de ce manuscrit portant sur l'optimisation du temps de calcul, cette méthode n'a pas été retenue dans la suite des travaux. Enfin, le chapitre s'est terminé par l'estimation du flot optique dans le cas d'un test présentant une variation lumineuse non régulière, c'est à dire uniquement localisée dans certaines zones de l'image. L'implémentation du contrôle local proposé pour la variable supplémentaire a ainsi donné une bonne estimation du champ de vecteurs dans les zones où la luminosité a changé, sans créer une perte de qualité de la solution dans les autres zones.

Dans le chapitre 4, nous avons implémenté les méthodes parallèles pour les modèles de résolution du problème du flot optique en petits déplacements présentées dans le chapitre 3. Nous avons commencé par présenter en détails les différentes étapes de l'algorithme pararéel. Pour cela, nous avons transformé le problème statique en un problème d'évolution en temps. Nous avons pu constater que les gains de temps de la méthode sont intéressants pour un grand nombre de pas grossiers, c'est-à-dire un grand nombre de processeurs. De plus, en étudiant l'algorithme proposé par Aubanel [7] dont l'ordre des opérations est modifié, le temps total a pu être encore réduit. Nous avons ensuite décrit une méthode permettant la parallélisation de l'estimation du flot optique pour toute une série d'images telle qu'une séquence vidéo. Bien que très intuitive, cette implémentation a l'avantage de traîter le flot optique sur plusieurs images de façon simultanée et indépendante sans nécessiter de communication entre les processeurs durant le calcul. On pourra se tourner vers cette méthode si le travail à effectuer concerne une longue séquence d'images ou encore la coupler à d'autre méthodes parallèles. Afin de gagner plus de temps que la méthode pararéelle, nous avons implémenté une méthode de décomposition de domaine. Après avoir décrit la méthode utilisée afin d'obtenir le meilleur découpage possible de l'image, nous avons relevé les temps de calcul pour plusieurs découpages et comparé les résultats avec ceux de la méthode pararéelle. La décomposition de domaine implémentée

a en effet permis un gain de temps de calcul plus important. En confiant la tâche de l'estimation de la solution sur un sous-domaine à plusieurs processeurs au lieu d'un seul et en utilisant le solveur MUMPS, nous avons pu ajouter encore un niveau de parallélisme. Cet algorithme s'est révélé être le plus efficace en termes de temps de calcul. Enfin, afin de garder les avantages de la méthode pararéelle, plus particulièrement la souplesse qu'elle propose via ses deux solveurs, tout en étant efficaces en temps, nous avons choisi de coupler cette méthode à la décomposition de domaine. En combinant ces deux méthodes, nous avons pu réaliser de meilleures performances que le pararéel seul.

Dans les chapitres 5 et 6, nous avons appliqué aux grands déplacements les différentes stratégies présentées dans les chapitres 3 et 4. Afin de traiter la non-linéarité du problème, nous avons dans un premier temps considéré une décomposition semi-implicite et utilisé la méthode de point fixe. La prise en compte de la luminosité a permis de mieux traiter les zones d'occlusions, plus importantes dans le cas des grands déplacements. L'adaptation de la régularisation ainsi que du maillage ont permis, comme dans les chapitres précédents, d'affiner les arêtes des objets des images et de rapidement réduire le nombre de degrés de liberté. Sur les résultats obtenus avec cet algorithme, l'estimation du mouvement observé sur les séquences de grands déplacements reste globalement bien orientée malgré des résultats moins précis dans les zones fortement texturées. Dans le chapitre 6, les différentes méthodes parallèles ont été reprises pour ce nouveau problème. La décomposition de domaine est restée la méthode la plus efficace en termes de gain de temps de calcul. Cependant, nous avons vu que la possibilité offerte par le pararéel de disposer d'un solveur fin et d'un solveur grossier permettait d'envisager d'autres stratégies afin de lever la non-linéarité. En effet, plusieurs méthodes ont été proposées grâce à la combinaison de ces deux solveurs. Dans un premier temps, nous avons implémenté la décomposition semi-implicite présentée pour l'algorithme séquentiel pour les deux solveurs. Nous avons ensuite gardé cette décomposition pour le solveur grossier et proposé deux différentes décompositions pour le solveur fin, la discrétisation explicite du terme non-linéaire et l'appel de l'algorithme des petits déplacements. Nous avons alors pu nous servir de la souplesse qu'apporte l'algorithme pararéel afin de simplifier la méthode. Enfin, la combinaison du pararéel et de la décomposition de domaine a permis de garder les avantages proposés par le pararéel tout en réduisant les temps de calcul de cette méthode.

Dans le dernier chapitre de la thèse, nous avons donné un exemple d'application concrète en utilisant l'estimation du flot optique dans le cas d'un problème de restauration de films. En suivant la stratégie de H. Grossauer [59] et à l'aide de nos modèles, nous avons pu déterminer les zones dégradées d'une image d'une séquence à partir de l'image suivante et de l'image précédente. En utilisant le masque de dégradation ainsi obtenu, nous avons ensuite reconstitué les parties endommagées grâce à l'implémentation des équations de Ginzburg-Landau. Pour terminer ce travail, nous avons choisi la méthode de parallélisation de séquences d'images permettant de réaliser plusieurs estimations de flots optiques de façon simultanée.

Parmi les perspectives envisagées, nous pouvons citer l'implémentation de nos algo-

rithmes utilisant la méthode des éléments finis dans le cas de la parallélisation sur carte graphique. En effet, jusqu'à présent, les programmes utilisant cette parallélisation utilisent la méthode des différences finies. Or, cette méthode ne permet pas d'utiliser des maillages non-structurés comme ce que nous avons fait grâce à l'adaptation de maillage. L'implémentation de la parallélisation des méthodes d'éléments finis sur carte graphique permettrait d'effectuer une adaptation du maillage et proposerait certainement des gains de temps encore plus importants.

Dans le chapitre concernant les séquences en petits déplacements, nous avons vu que le principe de rapprochement itératif apporte une nette amélioration de l'estimation du mouvement. Ce point pourrait être le sujet d'une nouvelle étude dans la continuité des travaux de cette thèse.

Nous avons utilisé des méthodes permettant de traiter des séquences dont la luminosité est variable. Une étude plus complète comparant les résultats obtenus avec ceux de l'approche proposée par Bruhn et Weickert pourrait être effectuée en complément de ce travail.

D'autre perspectives sont encore envisageables telles qu'une amélioration de la méthode du couplage du pararéel avec la décomposition de domaine dans le but de pouvoir traiter des images 3D. Cela pourrait avoir un intérêt pour des applications médicales ou encore pour l'analyse du trafic réel. Une autre perspective possible permettant d'améliorer la précision de l'estimation du mouvement serait d'étudier une approche proposant un couplage entre les méthodes d'estimation du flot optique et les méthodes de segmentation d'image permettant de déterminer les contours des objets. Pour finir, la prise en compte des incertitudes constituerait une amélioration au niveau des zones d'occlusions.

Annexe A

Résultats de convergence

Théorème A.0.1. *Soit $\boldsymbol{U}_{\boldsymbol{\Lambda}}$ la solution du problème (3.7). Supposons que $\boldsymbol{U}_{\boldsymbol{\Lambda}} \in \left(\bigcup_{\ell=1}^{L} \mathcal{H}^2(\Omega_\ell)\right) \bigcap \mathcal{H}^1(\Omega)$. Il existe une constante C qui ne dépend ni de h ni de $\boldsymbol{\Lambda}$ telle que*

$$\| \boldsymbol{U}_{\boldsymbol{\Lambda}} - \boldsymbol{U}_{\boldsymbol{\Lambda},h} \|_{\rho,\boldsymbol{\Lambda}} \leqslant C \sum_{\ell=1}^{L} \sqrt{\boldsymbol{\Lambda}_\ell} h_\ell \, \| \boldsymbol{U}_{\boldsymbol{\Lambda}} \|_{\mathcal{H}^1(\Omega_\ell)}$$

avec $\boldsymbol{\Lambda}_\ell = (\alpha_\ell, \alpha_\ell, \lambda)^t$.

Démonstration. A l'aide du lemme de Cea [35], nous avons une première majoration de l'erreur donnée par

$$\|\mathbf{U}_{\boldsymbol{\Lambda}} - \mathbf{U}_{\boldsymbol{\Lambda},h}\|_{\rho,\boldsymbol{\Lambda}} \leqslant C \left(\inf_{\mathbf{V}_h \in \mathcal{V}_h} \|\mathbf{U}_{\boldsymbol{\Lambda}} - \mathbf{V}_h\|_{\boldsymbol{\Lambda}} + \sup_{\mathbf{W}_h \in \mathcal{V}_h} \frac{(a_{\boldsymbol{\Lambda}} - a_{\boldsymbol{\Lambda},h})(\mathbf{V}_h, \mathbf{W}_h)}{\|\mathbf{W}_h\|_{\rho,\boldsymbol{\Lambda}}} \right)$$

On démontre la propriété suivante.

Proposition A.0.1. *Pour tout réel s tel que $1 \leqslant s \leqslant 2$, il existe une constante C indépendante de h et $\boldsymbol{\Lambda}$ telle que $\forall \boldsymbol{V} \in \left(\bigcup_{\ell=1}^{L} \mathcal{H}^s(\Omega_\ell)\right) \bigcap \mathcal{H}^1(\Omega)$ on ait*

$$\left\| \boldsymbol{V}_{\boldsymbol{\Lambda}} - \Pi_h^{\boldsymbol{\Lambda}} \boldsymbol{V} \right\|_{\rho,\boldsymbol{\Lambda}} \leqslant C \sum_{\ell=1}^{L} \sqrt{\boldsymbol{\Lambda}_\ell} h_\ell^{(s-1)} \, \| \nabla \boldsymbol{V}_{\boldsymbol{\Lambda}} \|_{\mathcal{H}^{(s-1)}(\Omega_\ell)}$$

où $\Pi_h^{\mathbf{\Lambda}} \boldsymbol{V}$ *désigne la projection orthogonale de* $\mathcal{H}^1(\Omega)$ *sur* $\mathcal{V}_h$ *pour la norme* $\|.\|_{\rho,\mathbf{\Lambda}}$.

Démonstration. Dans le cas où $s = 1$, le résultat vient de la définition de $\Pi_h^{\mathbf{\Lambda}}$. Si $s = 2$, en notant $\mathbf{I}_h$ l'opérateur de l'interpolation de Lagrange sur le maillage $\mathcal{T}_h$ on a

$$\left\|\mathbf{V}_{\mathbf{\Lambda}} - \Pi_h^{\mathbf{\Lambda}}\mathbf{V}\right\|_{\rho,\mathbf{\Lambda}} \leqslant \|\mathbf{V} - \mathbf{I}_h\mathbf{V}\|_{\rho,\mathbf{\Lambda}}$$

Le théorème 16.1 de [35], $\forall 1 \leqslant \ell \leqslant L$

$$|\ \mathbf{V}_{\mathbf{\Lambda}} - \mathbf{I}_h\mathbf{V}\ |_{H^1(\Omega_\ell)} \leqslant Ch_\ell \|\nabla\mathbf{V}\|_{\mathcal{H}^1(\Omega_\ell)}$$

donne le résultat. □

Afin d'estimer le second terme de l'erreur, on écrit

$$\begin{aligned}
(a_{\mathbf{\Lambda}} - a_{\mathbf{\Lambda},h})(\mathbf{V}_h, \mathbf{W}_h) &= \sum_{K\in\mathcal{T}_h} |\ \int_K \mathbf{W}_h^t(\mathbf{A}_\rho - \mathbf{A}_{\rho,h})\mathbf{V}_h d\mathbf{x}\ | \\
&\leqslant \sum_{K\in\mathcal{T}_h} \|\mathbf{A}_\rho - \mathbf{A}_{\rho,h}\|_{L^\infty(K)} \|\mathbf{V}_h\|_{L^2(K)} \|\mathbf{W}_h\|_{L^2(K)} \\
&\leqslant \sum_{K\in\mathcal{T}_h} \zeta_{\min(\mathbf{A}_\rho)}^{-\frac{1}{2}} \|\mathbf{A}_\rho - \mathbf{A}_{\rho,h}\|_{L^\infty(K)} \left\|\mathbf{A}_\rho^{\frac{1}{2}}\mathbf{V}_h\right\|_{L^2(K)} \left\|\mathbf{A}_\rho^{\frac{1}{2}}\mathbf{W}_h\right\|_{L^2(K)}
\end{aligned}$$

où $\zeta_{\min(\mathbf{A}_\rho)}$ désigne la valeur propre minimale de la matrice $\mathbf{A}_\rho$. Nous utilisons ensuite le théorème 7.1 de [44] : $\forall v \in W^{1,\infty}(K)$,

$$\inf_{q\in P_0} \|v - q\|_{L^\infty(K)} \leqslant Ch_K \|v\|_{W^{1,\infty}(K)}$$

pour obtenir l'inégalité

$$|\ (a_{\mathbf{\Lambda}} - a_{\mathbf{\Lambda},h})(\mathbf{V}_h, \mathbf{W}_h)\ | \leqslant C\zeta_{\min(\mathbf{A}_\rho)}^{-\frac{1}{2}} \|\mathbf{A}_\rho\|_{W^{1,\infty}(\Omega)} h \|\mathbf{V}_h\|_{\rho,\mathbf{\Lambda}} \|\mathbf{W}_h\|_{\rho,\mathbf{\Lambda}}.$$

La combinaison de ces estimations termine la preuve du théorème (A.0.1).

□

Annexe B

Outils pour la Γ-convergence

Nous rappelons ici les définitions et résultats utiles pour la Γ-convergence, pour plus de détails voir [3, 26].

Définition B.0.1. *Soit Ω un ensemble borné de $\mathbb{R}^n$ et u une fonction mesurable définie sur Ω à valeur dans $\mathbb{R}^m$. On définit $S = \mathbb{R}^m \cup \{\infty\}$ et $x \in \Omega$ fixé. Un élément $z \in S$ est appelé approximation limite de u en x si pour tout voisinage $\mathcal{V}$ de z*

$$\lim_{\rho \to \infty} \frac{1}{\rho^n} \mid \{y \in \Omega, \mid y - x \mid < \rho, u(y) \notin \mathcal{V}\} \mid = 0.$$

Si $z \in \mathbb{R}^m$, x est un point de Lebesgue de u. On note S_u le complémentaire de l'ensemble des points de Lebesgue de u.

Définition B.0.2. *On dit qu'une fonction $u \in L^1(\Omega, \mathbb{R}^m)$ est à variations bornées dans Ω si sa dérivée au sens des distributions Du, vérifie pour toute fonction $\varphi \in \mathcal{C}_c^1(\Omega, \mathbb{R}^{n,m})$*

$$\sum_{\alpha=1}^{m} \int_\Omega u^\alpha \operatorname{div}(\varphi^\alpha) dx = \sum_{\alpha=1}^{m} \sum_{i=1}^{n} \int_\Omega \varphi_i^\alpha dDu_i u^\alpha.$$

L'ensemble des fonctions à variations bornées est noté $BV(\Omega, \mathbb{R}^m)$.

Proposition B.0.2. *Si $u \in BV(\Omega, \mathbb{R}^m)$, alors sa dérivée au sens des distributions peut se décomposer de la façon suivante*

$$Du = D^a u + D^s u$$

où $D^a u$ et $D^s u$ sont respectivement la partie absolument continue et la partie singulière par rapport à la mesure de Lebesgue.

Définition B.0.3. *Une fonction u appartenant à $BV(\Omega, \mathbb{R}^m)$ est une fonction à variations bornées spéciale, $u \in SBV(\Omega, \mathbb{R}^m)$, si la restriction $D^s u|_{\Omega \setminus S_u} = 0$. Elle est appelée fonction à variations bornées spéciale généralisée, $u \in GSBV(\Omega, \mathbb{R}^m)$, si $g(u) \in SBV(\Omega, \mathbb{R}^m)$ pour toute fonction $g \in \mathcal{C}^1(\mathbb{R}^m)$ telle que ∇g est à support compact.*

Afin de définir la Γ-convergence, nous définissons les limites suivantes.

Définition B.0.4. *Soient (X, d) un espace métrique et F_n une suite de fonctions définies sur X à valeurs dans $\mathbb{R} \cup \{\pm\infty\}$. Pour tout $u \in X$, on définit les Γ-limites de F par*

$$F'(u) = \Gamma - \liminf_{k\to\infty} F_k(u) = \inf_{u_k \to u} \liminf_{k\to\infty} F_k(u_k),$$

et

$$F''(u) = \Gamma - \limsup_{k\to\infty} F_k(u) = \inf_{u_k \to u} \limsup_{k\to\infty} F_k(u_k),$$

On dit que F_k Γ-converge vers F si $F = F' = F''$.

Nous pouvons à présent énoncer le théorème de la Γ-convergence.

Théorème B.0.2. *Supposons que $(F_k)_k$ Γ-converge vers F et que pour tout k, u_k soit un minimiseur de F_k sur X. Alors, si une sous-séquence de $(u_k)_k$ converge vers $u \in X$, u est un minimiseur de F et $(F(u_k))_k$ converge vers $F(u)$.*

Remarque B.0.1. *Si $G : X \to [-\infty, +\infty]$ est continue et $(F(u_k))_k$ converge vers F, alors $(G + F_k)_k$ Γ-converge vers $G + F$.*

Bibliographie

[1] A. A. Alatan and L. Onural. Gibbs random field model based 3-d motion estimation by weakened rigidity. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 2, pages 790–794 vol.2, Nov 1994.

[2] L.M. Álvarez León, J. Esclarin Monreal, M. Lefébure, and J. Sánchez Pérez. A PDE model for computing the optical flow. page 1349–1356, 1999.

[3] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of bounded variation and free discontinuity problems*, volume 254. Clarendon Press Oxford, 2000.

[4] L. Ambrosio, M. Miranda Jr., and D. Pallara. Special functions of bounded variation in doubling metric measure spaces. *Calculus of variations : topics from the mathematical heritage of E. De Giorgi*, 14 :1–45, 2004.

[5] L. Ambrosio and V.M. Tortorelli. Approximation of functional depending on jumps by elliptic functional via t-convergence. *Communications on Pure and Applied Mathematics*, 43(8) :999–1036, 1990.

[6] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2 :283–310, 1989.

[7] E. Aubanel. Scheduling of tasks in the parareal algorithm. *Parallel Computing*, 37(3) :172–182, 2011.

[8] G. Aubert, R. Deriche, and P. Kornprobst. Computing optical flow via variational techniques. *SIAM Journal on Applied Mathematics*, 60 :156–182, 1999.

[9] G. Aubert, R. Deriche, and P. Kornprobst. Computing optical flow via variational techniques. *SIAM Journal on Applied Mathematics*, 60(1) :156–182, 1999.

[10] G. Aubert and P. Kornprobst. A mathematical study of the relaxed optical flow problem in the space BV(ω). *SIAM Journal on Mathematical Analysis*, 30 :1282–1308, 1999.

[11] G. Bal and Y. Maday. A "parareal" time discretization for non-linear PDE's with application to the pricing of an american put. In *Recent Developments in Domain Decomposition Methods*, volume 23 of *Lecture Notes in Computational Science and Engineering*, pages 189–202. Springer Berlin Heidelberg, 2002.

[12] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12 :43–77, 1994.

[13] A. Batmalle, P.E. Bécue, and V. Le Gallic. Accélération des méthodes de parallélisation en temps par approches de type quasi-Newton. Technical report, Ecole Nationnale Supérieure de Cachan, France, 2011.

[14] M. Bedez, Z. Belhachmi, O. Haeberlé, R. Greget, S. Moussaoui, J.-M. Bouteiller, and S. Bischoff. A fully parallel in time and space algorithm for simulating the electrical activity of a neural tissue. *Journal of Neuroscience Methods*, 257 :17–25, 2016.

[15] Z. Belhachmi and F. Hecht. Control of the effects of regularization on variational optic flow computations. *Journal of Mathematical Imaging and Vision*, 40 :1–19, 2011.

[16] Z. Belhachmi and F. Hecht. An adaptive approach for segmentation and TV denoising in the optic flow estimation. 2014. Working paper or preprint.

[17] Z. Belhachmi, M. Kallel, M. Moaker, and A. Theljani. Weighted harmonic and Ginzburg-Landau equations in image inpainting. 2015. Working paper or preprint.

[18] C. Bernard. Fast optic flow computation with discrete wavelets. Technical report, Centre de Mathématiques Appliquées, École Polytechnique, 1997.

[19] C.P. Bernard. Discrete wavelet analysis for fast optic flow computation. *Applied and Computational Harmonic Analysis*, 11(1) :32–63, 2001.

[20] M.J. Black. *Robust incremental optical flow.* PhD thesis, Yale university, 1992.

[21] M.J. Black and P. Anandan. Robust dynamic motion estimation over time. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings CVPR'91*, pages 296–302, 1991.

[22] M.J. Black and P. Anandan. The robust estimation of multiple motions : Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63 :75–104, 1996.

[23] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, 1987.

[24] L. Blanc-Féraud, M. Barlaud, and T. Gaidon. Motion estimation involving discontinuities in multiresolution scheme. *Optical engineering*, 32 :1475–1482, 1993.

[25] B. Bourdin and A. Chambolle. Implementation of an adaptive finite-element approximation of the mumford-shah functional. *Numerische Mathematik*, 85 :609–646, 2000.

[26] A. Braides. *Gamma-convergence for Beginners*, volume 22. Clarendon Press, 2002.

[27] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision - ECCV 2004*, volume 3024, pages 25–36. Springer Berlin Heidelberg, 2004.

[28] T. Brox and J. Malik. Large displacement optical flow : Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33 :500–513, 2011.

[29] A. Bruhn. *Variational optic flow computation : Accurate modelling and efficient numerics*. PhD thesis, Department of Mathematics and Computer Science, Saarland University, Saarbrucken, Diss, 2006.

[30] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Real-time optic flow computation with variational methods. In *Computer Analysis of Images and Patterns*, volume 2756, pages 222–229. Springer Berlin Heidelberg, 2003.

[31] A. Bruhn, J. Weickert, and C. Schnorr. Lucas/Kanade meets Horn/Schunck : Combining local and global optic flow methods. *International Journal of Computer Vision*, 61 :211–231, 2005.

[32] A. Chambolle and G. Dal Maso. Discrete approximation of the mumford-shah functional in dimension two. *ESAIM : Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 33 :651–672, 1999.

[33] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2450, 2013.

[34] J. Chergui, I. Dupays, D. Girou, P.-F. Lavallée, D. Lecas, and P. Wautelet. Message Passing Interface (MPI). *Cours de l'Institut du Développement et des Ressources en Informatique Scientifique*, 2011.

[35] P.G. Ciarlet. Basic error estimates for elliptic problems. In *Finite Element Methods (Part 1)*, volume 2 of *Handbook of Numerical Analysis*, pages 17–351. Elsevier, 1991.

[36] I. Cohen. Nonlinear variational method for optical flow computation. In *Proceedings of the Scandinavian conference on image analysis*, volume 1, pages 523–530, 1993.

[37] G. Dal Maso. *An introduction to* Γ*-convergence*, volume 8. Springer Science & Business Media, 2012.

[38] E. De Giorgi and L. Ambrosio. chapter New Functionals in Calculus of Variations, pages 49–59. Springer US, 1989.

[39] E. De Giorgi and T. Franzoni. Su un tipo di convergenza variazionale. *Atti Accad. Naz. Lincei Cl. Sci. Fis. Mat. Natur. Rend.*, 68 :842—-850, 1975.

[40] R. Deriche, P. Kornprobst, and G. Aubert. Optical-flow estimation while preserving its discontinuities : A variational approach. In *Asian Conference on Computer Vision*, pages 69–80. Springer, 1995.

[41] V. Devlaminck and J.-P. Dubus. Estimation of compressible or incompressible deformable motions for density images. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 125–128. IEEE, 1996.

[42] V. Dolean, P. Jolivet, and F. Nataf. *An Introduction to Domain Decomposition Methods : algorithms, theory and parallel implementation*, volume 144. France, 2015. Lecture.

[43] V. Dolean, S. Lanteri, and F. Nataf. Convergence analysis of additive Schwarz for the Euler equations. *Applied Numerical Mathematics*, 49 :153–186, 2004.

[44] T. Dupont and R. Scott. Polynomial approximation of functions in Sobolev spaces. *Mathematics of Computation*, 34 :441–463, 1980.

[45] L.C. Evans. Partial differential equations. *Graduate Studies in Mathematics 2nd edn.*, 19, 2010.

[46] R. Fezzani. *Approche parallèle pour l'estimation du flot optique par méthode variationnelle*. PhD thesis, Université Pierre et Marie Curie-Paris VI, Paris, France, 2011.

[47] D.J. Fleet and A.D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5 :77–104, 1990.

[48] J.C. Gee, L. Le Briquer, C. Barillot, D.R. Haynor, and R.K. Bajcsy. Bayesian approach to the brain image matching problem. In *Medical Imaging 1995*, pages 145–156. International Society for Optics and Photonics, 1995.

[49] M.A. Gennert and S. Negahdaripour. Relaxing the brightness constancy assumption in computing optical flow. *Technical Report*, 1987.

[50] S. Ghosal and P. Vanek. A fast scalable algorithm for discontinuous optical flow estimation. *IEEE Transactions on pattern analysis and machine intelligence*, 18(2) :181–194, 1996.

[51] J.J. Gibson. The senses considered as perceptual systems. 1966.

[52] D. Gilliocq-Hirtz and Z. Belhachmi. Coupling parareal and adaptive control in optical flow estimation with application in movie's restoration. In *International Conference on Computer Vision and Image Analysis Applications (ICCVIA, 2015)*, pages 1–6. IEEE, Jan 2015.

[53] D. Gilliocq-Hirtz and Z. Belhachmi. A massively parallel multi-level approach to a domain decomposition method for the optical flow estimation with varying illumination. *SMAI Journal of Computational Mathematics, submitted*, 2016.

[54] D. Gilliocq-Hirtz and Z. Belhachmi. Parallel methods for the optical flow in large displacements. *In preparation*, 2016.

[55] V.L. Ginzburg. On the theory of superconductivity. *Il Nuovo Cimento (1955-1965)*, 2(6) :1234–1250, 1955.

[56] V.L. Ginzburg and L.D. Landau. *On Superconductivity and Superfluidity : A Scientific Autobiography*, chapter On the Theory of Superconductivity, pages 113–137. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[57] D.A. Gordon. Static and dynamic visual fields in human space perception. *J. Opt. Soc. Am.*, 55(10) :1296–1303, 1965.

[58] C. Grava. *Compensation de mouvement par réseaux neuronaux cellulaires : application en imagerie médicale*. PhD thesis, Institut National des Sciences Appliquées de Lyon, France, 2003.

[59] H. Grossauer. Inpainting of movies using optical flow. In *Mathematical Models for Registration and Applications to Medical Imaging*, pages 151–162. Springer, 2006.

[60] H. Grossauer and O. Scherzer. Using the complex Ginzburg-Landau equation for digital inpainting in 2D and 3D. In *Scale-Space*, pages 225–236. Springer, 2003.

[61] F. Guichard and L. Rudin. Accurate estimation of discontinuous optical flow by minimizing divergence related functionals. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 497–500. IEEE, 1996.

[62] S.N. Gupta and J.L. Prince. On div-curl regularization for motion estimation in 3-d volumetric imaging. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 929–932 vol.1, Sep 1996.

[63] R.M. Haralick and J.S. Lee. The facet approach to optic flow. Technical report, Virginia Polytechnic Inst and State Univ Blacksburg Dept Of Computer Science, 1983.

[64] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey vision conference*, volume 15, pages 147–151. Citeseer, 1988.

[65] F. Hecht. New development in FreeFem++. *Journal of Numerical Mathematics*, 20 :251–266, 2012.

[66] D.J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1 :279–302, 1988.

[67] F. Heitz and P. Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12) :1217–1232, 1993.

[68] B. Horn. *Robot vision*. MIT press, 1986.

[69] B.K. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17 :185–203, 1981.

[70] J. J. Koenderink. Optic flow. *Vision research*, 26(1) :161–179, 1986.

[71] J. J. Koenderink and A. J. Van Doorn. Affine structure from motion. *JOSA A*, 8(2) :377–385, 1991.

[72] J. J. Koenderink and A.J. Van Doorn. How an ambulant observer can construct a model of the environment from the geometrical structure of the visual inflow. *Kybernetik*, 77 :224–227, 1977.

[73] J.J. Koenderink and A. J. Van Doorn. Dynamic shape. *Biological cybernetics*, 53(6) :383–396, 1986.

[74] J.J. Koenderink and A.J. Van Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24(1) :51–59, 1976.

[75] T. Kohlberger, C. Schnörr, A. Bruhn, and J. Weickert. Domain decomposition for variational optical flow computation. *Image Processing, IEEE Transactions on*, 14 :1125–1137, 2005.

[76] A. Kumar, A.R. Tannenbaum, and G.J. Balas. Optical flow : a curve evolution approach. *Image Processing, IEEE Transactions on*, 5 :598–610, 1996.

[77] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps "pararéel". *Comptes Rendus de l'Académie des Sciences, Série I, Mathematics*, 332 :661–668, 2001.

[78] P.-L. Lions. On the Schwarz alternating method. III : a variant for nonoverlapping subdomains. In *Third international symposium on domain decomposition methods for partial differential equations*, volume 6, pages 202–223. SIAM Philadelphia, PA, 1990.

[79] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B : Biological Sciences*, 208(1173) :385–397, 1980.

[80] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, pages 674–679. Morgan Kaufmann Publishers Inc., 1981.

[81] V. Manet. La méthode des éléments finis : vulgarisation des aspects mathématiques, illustration des capacités de la méthode. DEA. Eléments finis pour l'ingénieur, ViM2, Lyon, France, 2014.

[82] J. Marzat. Estimation temps réel du flot optique. *Rapport de Stage, INRIA, ENSEM*, 2008.

[83] J. Marzat, Y. Dumortier, and A. Ducrot. Real-time dense and accurate parallel optical flow using CUDA. In *17th International Conference in Central Europe on Computer Graphics, WSCG '2009, Visualization and Computer Vision in cooperation with EUROGRAPHICS, University of West Bohemia Plzen, Czech Republic*, pages 105–112. Václav Skala-UNION Agency, 2009.

[84] F. Massanes, M. Cadennes, and J.G. Brankov. Compute-unified device architecture implementation of a block-matching algorithm for multiple graphical processing unit cards. *Journal of Electronic Imaging*, 20 :033004, 2011.

[85] E. Mémin and P. Pérez. A multigrid approach for hierarchical motion estimation. In *Sixth International Conference on Computer Vision*, pages 933–938. IEEE, 1998.

[86] E. Mémin, P. Pérez, and D. Machecourt. *Dense estimation and object-oriented segmentation of the optical flow with robust techniques.* PhD thesis, INRIA, 1996.

[87] Y. Mileva, A. Bruhn, and J. Weickert. Illumination-robust variational optical flow with photometric invariants. In F. Hamprecht, C. Schnorr, and B. Jahne, editors, *Pattern Recognition*, pages 152–162. Springer Berlin Heidelberg, 2007.

[88] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *Technical report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation.* 1980.

[89] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5) :577–685, 1989.

[90] H.-H. Nagel. On the estimation of optical flow : Relations between different approaches and some new results. *Artificial Intelligence*, 33(3) :299–324, 1987.

[91] H.-H. Nagel. On the estimation of optical flow : Relations between different approaches and some new results. *Artificial intelligence*, 33(3) :299–324, 1987.

[92] H.-H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8 :565–593, 1986.

[93] Y. Nakaya and H. Harashima. Motion compensation based on spatial transformations. *IEEE Transactions on Circuits and Systems for Video Technology*, 4 :339–356, 366–7, 1994.

[94] S. Negahdaripour and C.-H. Yu. A generalized brightness change model for computing optical flow. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 2–11. IEEE, 1993.

[95] P. Nesi. Variational approach to optical flow estimation managing discontinuities. *Image and Vision Computing*, 11(7) :419–439, 1993.

[96] J.-M. Odobez. *Estimation, détection et segmentation du mouvement dans une séquence d'images : une approche robuste et markovienne*. PhD thesis, Université de Rennes 1, France, 1994.

[97] M. Orkisz and P. Clarysse. Estimation du flot optique en présence de discontinuités : une revue. *Traitement du Signal*, 13(5) :489–513, 1996.

[98] M. Otte and H.-H. Nagel. Optical flow estimation : Advances and comparisons. In *Computer Vision — ECCV '94*, volume 800, pages 49–60. Springer Berlin Heidelberg, 1994.

[99] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67 :141–158, 2006.

[100] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67 :141–158, 2006.

[101] M. Proesmans, L. Van Gool, E. Pauwels, and A. Oosterlinck. Determination of optical flow and its discontinuities using non-linear diffusion. In *European Conference on Computer Vision*, pages 294–304. Springer, 1994.

[102] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D : Nonlinear Phenomena*, 60 :259–268, 1992.

[103] P. Ruhnau, T. Kohlberger, C. Schnörr, and H. Nobach. Variational optical flow estimation for particle image velocimetry. *Experiments in Fluids*, 38 :21–32, 2005.

[104] A. Samake. *Large scale nonconforming domain decomposition methods*. PhD thesis, Université de Grenoble, France, 2014.

[105] C. Sbordone. Su alcune applicazioni di un tipo di convergenza variazionale. *Annali della Scuola Normale Superiore di Pisa - Classe di Scienze*, 2(4) :617–638, 1975.

[106] C. Schnörr. Determining optical flow for irregular domains by minimizing quadratic functionals of a certain class. *International Journal of Computer Vision*, 6(1) :25–38, 1991.

[107] V.E. Seferidis and M. Ghanbari. General approach to block-matching motion estimation. *Optical Engineering*, 32 :1464–1474, 1993.

[108] E.P. Simoncelli and E.H. Adelson. Computing optical flow distributions using spatio-temporal filters. *Technical Report*, 1991.

[109] A. Spinei. *Estimation du mouvement par triades de filtres de Gabor. Application au mouvement de transparence.* PhD thesis, Institut national polytechnique de Grenoble, Grenoble, France, 1998.

[110] C. Stiller and J. Konrad. Estimating motion in image sequences. *Signal Processing Magazine, IEEE*, 16 :70–91, 1999.

[111] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *Computer Vision – ECCV 2010*, volume 6311, pages 438–451. Springer Berlin Heidelberg, 2010.

[112] A. Theljani. *Partial Differential Equations Methods and Regularization Techniques for Image Inpainting.* PhD thesis, University of Tunis El Manar, LAMSIN Laboratory, Tunisia, 2015.

[113] A.N. Tikhonov and V.Y. Arseni. Solutions of ill-posed problems. *Winston and Sons, Washington, DC*, 1977.

[114] M. Tistarelli. Computation of coherent optical flow by using multiple constraints. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 263–268. IEEE, 1995.

[115] R. Verfürth. A posteriori error estimation and adaptive mesh-refinement techniques. *Journal of Computational and Applied Mathematics*, 50 :67–83, 1994.

[116] A. Verri, F. Girosi, and V. Torre. Differential techniques for optical flow. *JOSA A*, 7(5) :912–922, 1990.

[117] A. Verri and T. Poggio. Motion field and optical flow : Qualitative properties. *IEEE Transactions on pattern analysis and machine intelligence*, 11(5) :490–498, 1989.

[118] Y. Wang, O. Lee, and A. Vetro. Use of two-dimensional deformable mesh structures for video coding. II. the analysis problem and a region-based coder employing an active mesh representation. *IEEE Transactions on Circuits and Systems for Video Technology*, 6 :647–659, 1996.

[119] J. Weickert, A. Bruhn, N. Papenberg, and T. Brox. Variational optic flow computation : From continuous models to algorithms. In *International Workshop on Computer Vision and Image Analysis (ed. L. Alvarez), IWCVIA'03, Las Palmas de Gran Canaria*, volume 3, pages 1–6, 2003.

[120] J. Weickert and C. Schnörr. Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of Mathematical Imaging and Vision*, 14 :245–255, 2001.

[121] Y. Weiss and E.H. Adelson. Motion estimation and segmentation using a recurrent mixture of experts architecture. In *Neural Networks for Signal Processing V. Proceedings of the 1995 IEEE Workshop*, pages 293–302, 1995.

[122] Y. Weiss, E.P. Simoncelli, and E.H. Adelson. Motion illusions as optimal percepts. *Nature neuroscience*, 5 :598–604, 2002.

[123] R. P. Wildes, A.-M. Lanzillotto, M.J. Amabile, and T.-S. Leu. Physically based fluid flow recovery from image sequences. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 969–975. IEEE, 1997.

[124] D. Willick and Y.-H. Yang. Experimental evaluation of motion constraint equations. *CVGIP : Image Understanding*, 54(2) :206–214, 1991.

[125] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L 1 optical flow. pages 214–223, 2007.

[126] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel. Complementary optic flow. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 5681, pages 207–220. Springer Berlin Heidelberg, 2009.

Techniques variationnelles et calcul parallèle en imagerie. Estimation du flot optique avec luminosité variable en petits et larges déplacements.

Diane Gilliocq-Hirtz

Résumé

Le travail présenté dans cette thèse porte sur l'estimation du flot optique par méthodes variationnelles en petits et en grands déplacements. Nous proposons un modèle basé sur la combinaison locale-globale à laquelle nous ajoutons la prise en compte des variations de la luminosité. La particularité de ce manuscrit réside dans l'utilisation de la méthode des éléments finis pour la résolution des équations. En effet, cette méthode se fait pour le moment très rare dans le domaine du flot optique. Grâce à ce choix de résolution, nous proposons d'implémenter un contrôle local de la régularisation ainsi qu'une adaptation de maillage permettant d'affiner la solution au niveau des arêtes de l'image. Afin de réduire les temps de calcul, nous parallélisons les programmes. La première méthode implémentée est la méthode parallèle en temps appelée pararéel. En couplant un solveur grossier et un solveur fin, cet algorithme permet d'accélérer les calculs. Pour pouvoir obtenir un gain de temps encore plus important et également traiter les séquences en haute définition, nous utilisons ensuite une méthode de décomposition de domaine. Combinée au solveur massivement parallèle MUMPS, cette méthode permet un gain de temps de calcul significatif. Enfin, nous proposons de coupler la méthode de décomposition de domaine et le pararéel afin de profiter des avantages de chacune. Dans une seconde partie, nous appliquons tous ces modèles dans le cas de l'estimation du flot optique en grands déplacements. Nous proposons de nous servir du pararéel afin de traiter la non-linéarité de ce problème. Nous terminons par un exemple concret d'application du flot optique en restauration de films.

Mots clés : Flot optique, méthode variationnelle, méthode des éléments finis, variation de luminosité, contrôle adaptatif, calcul parallèle, pararéel, décomposition de domaine.

Abstract

The work presented in this thesis focuses on the estimation of the optical flow through variational methods in small and large displacements. We propose a model based on the combined local-global strategy to which we add the consideration of brightness intensity variations. The particularity of this manuscript is the use of the finite element method to solve the equations. Indeed, for now, this method is really rare in the field of the optical flow. Thanks to this choice of resolution, we implement an adaptive control of the regularization and a mesh adaptation to refine the solution on the edges of the image. To reduce computation times, we parallelize the programs. The first method implemented is a parallel in time method called parareal. By combining a coarse and a fine solver, this algorithm speeds up the computations. To save even more time and to also be able to handle high resolution sequences, we then use a domain decomposition method. Combined with the massively parallel solver MUMPS, this method allows a significant reduction of computation times. Finally, we propose to couple the domain decomposition method and the parareal to have the benefits of both methods. In the second part, we apply all these models to the case of the optical flow estimation in large displacements. We use the parareal method to cope with the non-linearity of the problem. We end by a concrete example of application of the optical flow in film restoration.

Keywords : Optical flow, variational method, finite element method, varying illumination, adaptive control, parallel computation, parareal, domain decomposition.