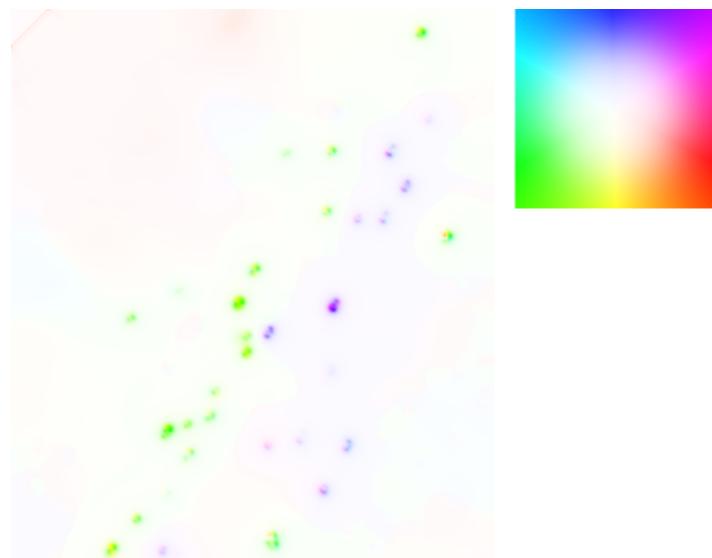


RAPPORT DE STAGE

Développement d'une méthode variationnel pour le
calcul du flot optique adapté au petits objets
mobiles dans des vidéos



Étudiant :
Théotim BARBIER

Maitre de stage :
Charles KERVANN

CENTRE INRIA DE L'UNIVERSITÉ DE RENNES
EQUIPE SAIRPICO
CAMPUS UNIVERSITAIRE DE BEAULIEU
AVENUE DU GÉNÉRAL LECLERC
35042 RENNES CEDEX

Remerciements

Je souhaite tout d'abord remercier le centre INRIA de l'université de Rennes, et plus particulièrement le directeur du centre Mr Pratick Gros, pour m'avoir accueilli au sein de cet établissement.

Je tiens ensuite à remercier mon tuteur au sein de l'INRIA, Mr Charles Kervrann, pour m'avoir fait confiance dans la mission qu'il m'a donné, m'avoir fait découvrir le domaine du traitement d'image et surtout pour m'avoir accompagné tout au long du stage.

Enfin, je tiens également à remercier l'équipe de recherche SAIRPICO de l'INRIA, au sein de laquelle j'ai effectué mon stage, qui m'ont gentiment accueilli et transmis leurs connaissances tout au long du stage.

Sommaire

Contents

1	Introduction	6
1.1	Présentation de l'INRIA	6
1.2	Présentation de l'équipe SAIRPICO	6
2	Mission	7
2.1	partie théorique	7
2.2	Méthode du stage	10
2.3	Implémentation	12
2.4	Problèmes rencontrés	15
2.5	Résultats expérimentaux	20
2.5.1	Comparaisons normes et initialisation	20
2.5.2	Analyse de l'influence du terme de parcimonie	22
2.5.3	Comparaisons des pré-traitements appliqués sur les images	22
2.5.4	Comparaison avec IPOL et SMOFlow	22
2.5.5	Expérience sur des image de microscopie	24
2.6	Perspectives	24
3	Conclusion	28
A	Linéarisation de l'équation 2.2	30
B	Problème convexe	30
C	Differentiation automatique pytorch	31
D	Colorisation	32
E	approximation des normes	32
F	Différences finies	32
G	Problèmes rencontrés	35
H	Problème avec le critère DFD	37
I	Exemple des pré-traitements	37

Résumé

Ce rapport présente mon expérience et ma mission en tant que stagiaire au sein du centre INRIA de l'université de Rennes. L'INRIA est un centre de recherche en informatique. Mon stage s'est déroulé au sein de l'équipe SAIRPICO, qui est spécialisée dans le traitement d'image de microscopie. L'équipe élaboré autant des méthodes traditionnelles de vision par ordinateur que des méthodes plus récentes basées sur la conception de réseaux de neurones.

Mon stage a porté sur le sujet du flot optique. Le flot optique vise à calculer le mouvement des éléments mobiles entre deux images. Il permet de repérer les mouvements apparents des objets sur l'image, comme par exemple le déplacement d'un véhicule. Dans le cadre de mon stage, le flot optique est principalement utilisé pour repérer le mouvement d'éléments microscopiques comme des protéines dans des cellules. Il permet ainsi d'identifier ces objets et d'estimer leurs trajectoires.

Néanmoins, les méthodes classiques de vision par ordinateur ne permettent pas de calculer le flot optique pour de très petits objets, de la taille quelques pixels. Ainsi, ma mission au sein de l'équipe a consisté à étudier une méthode variationnelle afin de calculer le flot optique entre deux images présentant de petits objets. La méthode que j'ai étudiée a pour particularité d'introduire un terme de régularisation et parcimonie (« sparse ») qui doit permettre de mieux localiser les frontières des objets mobiles tout en imposant un flot nul dans l'arrière plan. Mon stage a commencé par l'étude théorique des méthodes existantes du calcul du flot optique, notamment les méthodes variationnelles. Ensuite, j'ai abordé l'implémentation informatique du calcul en recourant à la bibliothèque pré-existante de « open CV » en python pour calculer le flot optique. Enfin, j'ai programmé moi-même l'implémentation en python de ma méthode du calcul du flot optique, qui consiste à minimiser une fonctionnelle d'énergie variationnelle à l'aide de la bibliothèque « pytorch ».

Ce rapport présente dans un premier temps le centre INRIA et l'équipe dans laquelle s'est déroulé mon stage. Ensuite, je reviens sur ma mission au sein de l'équipe en commençant par une introduction théorique du problème du calcul du flot optique. Je décris par la suite l'implementation d'une solution abordée durant le stage ainsi que les problèmes rencontrés. Enfin, je présente les résultats obtenus et une comparaison avec d'autres méthodes existantes.

Lexique

INRIA : Institut national de recherche en sciences et technologies du numérique

Flot optique/flux optique : Le flux optique est le mouvement apparent des objets, surfaces et contours d'une scène visuelle, causé par le mouvement relatif entre un observateur (l'œil ou une caméra) et la scène., Wikipedia [15]

Tracking (vision par ordinateur) : Le tracking en vision par ordinateur correspond au suivi d'un ou plusieurs objets sur une vidéo à l'aide d'un programme informatique.

Pytorch : Bibliothèque informatique python utilisée pour l'implémentation de réseaux de neurones.

Displaced Frame Differences (DFD): Le critère DFD entre deux images successives dont on connaît le flot optique consiste à reconstruire la deuxième image à l'aide de la première image et du champ de déplacement estimé. Ainsi, on peut effectuer la différence entre cette image reconstruite et l'image réelle.

1 Introduction

Pour mon stage de spécialité de quatrième année à l'INSA de Rouen, j'ai effectué 10 semaines à l'INRIA de l'université de Rennes au sein de l'équipe SAIRPICO.

1.1 Présentation de l'INRIA

L'INRIA désigne l'Institut national de recherche en sciences et technologies du numérique. Ce centre de recherche est un institut public spécialisé en informatique et mathématiques. Tout d'abord créé sous le nom de l'IRIA en 1979, on y étudie principalement l'informatique et l'automatique de l'époque. Actuellement, l'INRIA se focalise sur l'application de l'informatique et des technologies pour différents enjeux et domaines : l'énergie, la communication, les transports, la sécurité et de la protection de la vie privée, la santé, etc. Les 10 centres de recherche INRIA sont répartis dans toute la France et la plupart sont en collaboration avec une université de la ville. Aujourd'hui, l'INRIA réunit environ 3800 scientifiques réunis dans 220 équipes-projets, chacune spécialisée dans différentes applications. C'est un acteur global de la recherche informatique dans le monde.

L'INRIA de l'université de Rennes, dans laquelle j'ai effectué mon stage, a été créé en 1980. Le centre est spécialisé dans les domaines suivants : société numérique sûre, interactions humains-robots-mondes virtuels, biologie et santé numérique, écologie numérique. L'INRIA de Rennes réunit 30 équipes pour un total de 600 chercheurs.

1.2 Présentation de l'équipe SAIRPICO

L'équipe dans laquelle j'ai accompli mon stage est l'équipe SAIRPICO, sigle pour Imagerie Spatio-Temporelle, Intelligence Artificielle et Calcul Numérique pour la Biologie Cellulaire et Chemobiologie. Cette équipe est spécialisée dans le traitement d'image pour l'imagerie de microscopie, c'est à dire l'analyse des images observées à l'aide de microscopes photoniques



Figure 1.1: INRIA de Rennes

et électroniques. Un des problèmes majeurs rencontré dans ce domaine est que l'image microscopique permet d'observer des éléments de petits taille, à l'échelle nano-métrique, comme des cellules, des protéines, des tissus dont la taille et la netteté peuvent rendre le traitement de l'image difficile. L'équipe SAIRPICO est focalisée sur la recherche de méthodes et le développement d'algorithmes pour traiter ces images. L'équipe est principalement composée de chercheurs et de doctorants, localisés à Rennes et à Paris (Institut Curie).

2 Mission

Dans cette partie, je vais expliquer en quoi consistait ma mission durant mon stage. Tout d'abord, je présente l'aspect théorique du problème du flot optique.

2.1 partie théorique

Flot Optique

Definition : Le flot optique (ou flux optique) [15] correspond au calcul du mouvement visible entre deux images. Le mouvement observé sur une image peut provenir de différentes sources : éléments se déplaçant à l'intérieur du cadre, un mouvement global de l'espace par rapport à l'observateur, ou un mouvement de l'observateur lui-même. Ainsi, ce qu'on appelle le flot optique est la représentation mathématique de ce mouvement sur un plan 2D. Il est traditionnellement représenté par un champ de vecteurs sur une grille sous-échantionnée de pixels de l'image ou bien d'une représentation en couleur, plus « continue », qui, grâce à une lut de couleur, associe une couleur à une direction et une vitesse de

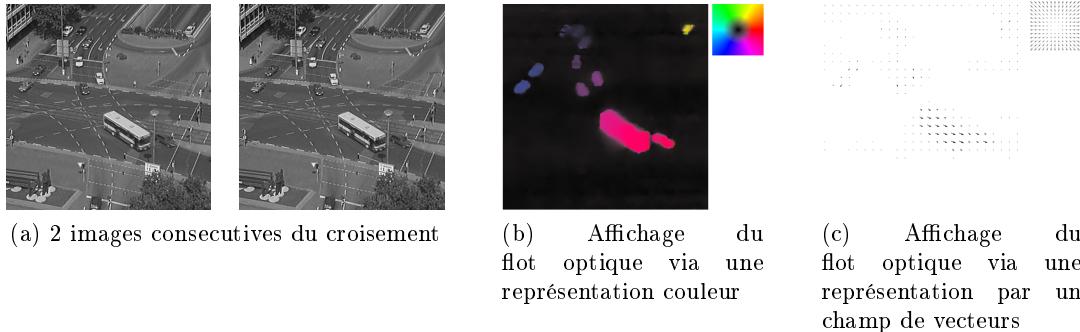


Figure 2.1: exemple de flot optique pour la vue de véhicules en mouvement sur un croisement routier

déplacement pour chaque position/pixel dans l'image (pas de sous-échantillonnage spatial dans l'affichage) (voir *figure 2.1*).

Le flot optique a énormément d'applications possibles en traitement d'image et en vision par ordinateur. Par exemple, en robotique, le flot optique permet de détecter les objets environnants et de calculer sa trajectoire et de planifier son déplacement afin d'éviter les collisions avec les objets dans la scène 3D. Il peut aussi servir dans la production d'effets audio visuels pour dans un contexte de cinématographie [13].

Dans notre cas, nous utilisons le flot optique sur des images de microscopie. Ces images comprennent des molécules dans des cellules. Le flot optique permet d'estimer le mouvement de ces particules pour leur dynamiques et le traffic intracellulaire. Notons que le flot optique peut aussi bien être calculé entre 2 images 2D ou entre deux volumes (3D).

Formulation mathématique

Pour estimer le flot optique, nous avons besoin de formuler quelques hypothèses. Soit séquence d'image 2D comme une fonction continue :

$$I : \Omega \times T \rightarrow \mathbb{R}$$

où $\Omega \subset \mathbb{R}^2$ représente les pixels de l'image et T désigne la durée de la séquence. Cette fonction représente l'intensité lumineuse d'un pixel (x, y) de l'image à un instant t . Le flot optique est défini comme une fonction :

$$w : \Omega \rightarrow \mathbb{R}^2$$

qui associe un pixel de l'image à un vecteur 2D dans l'espace représentant le mouvement associé au pixel.

L'hypothèse principale utilisée pour calculer le flot optique est l'invariance d'intensité lumineuse (brightness constancy) [3] définie par le fait que l'intensité lumineuse d'un objet en mouvement sur une image se conserve sur sa trajectoire. Cette contrainte peut être exprimée continuellement avec l'équation :

$$\frac{dI}{dt}(x(t), t) = 0 \quad (2.1)$$

où $x(t)$ correspond à la trajectoire de l'objet au fil du temps. Cette équation sous-tend que la variation d'intensité lumineuse d'un objet en déplacement est nulle. Cependant, expérimentalement, comme on ne dispose pas d'une fonction continue, on peut recourir à l'équation discrétisée associée :

$$I(x + w(x), t + 1) - I(x, t) = 0 \quad (2.2)$$

où w désigne le mouvement entre l'image au temps t et au temps $t + 1$. Ainsi, connaissant l'image I_1 et I_2 , l'équation (2.2) peut être résolue. Néanmoins, l'équation (2.2) pose certains problèmes dans un contexte d'optimisation du fait de la non linéarité (et de la non continuité) expérimentale de la fonction I . En général, on recourt à une version linéarisée de l'équation (2.1) (précisions en annexe A) ainsi définie qui suppose que le déplacement des objets est de faible amplitude:

$$\frac{\partial I}{\partial x_1}(x, t) \times w_1(x) + \frac{\partial I}{\partial x_2}(x, t) \times w_2(x) + \frac{\partial I}{\partial t}(x, t) = 0 \quad (2.3)$$

où $x = (x_1, x_2)$ représente la variable sur Ω et $w = (w_1, w_2)$.

Cette contrainte exprimée sous la forme d'une équation à résoudre est utilisée pour construire un terme d'attache-aux-données, noté ρ_{data} . On peut remarquer que les équations (2.2) et (2.3) ne possèdent pas une unique solution w du fait que notre vecteur a deux composantes. Pour que le problème soit bien posé, il nous faut rajouter une deuxième contrainte de régularisation spatiale qui peut être locale ou globale. On élabore un terme de régularisation ρ_{reg} . Le terme de régularisation le plus utilisé est le terme de variation totale (TV):

$$\rho_{reg}(x, w) = \nabla w_1^2(x) + \nabla w_2^2(x) \quad (2.4)$$

Finalement, pour résoudre le problème du flot optique, la principale approche utilisée est une méthode variationnelle qui revient à minimiser l'énergie :

$$E(w) = \int_{\Omega} \rho_{data}(x, I, w) + \lambda \rho_{reg}(x, w) dx \quad (2.5)$$

où λ est un paramètre permettant de contrôler l'importance de chaque terme. On peut également réécrire l'équation (2.5) de cette façon :

$$E(w) = \int_{\Omega} \lambda' \rho_{data}(x, I, w) + (1 - \lambda') \rho_{reg}(x, w) dx$$

où $\lambda' \in [0; 1]$ remplace λ pour former une combinaison linéaire[5]. Cette formulation permet de gérer plus précisément l'importance des deux termes de notre équation.

Etat de l'art des méthodes utilisées

Il existe de nombreuses méthodes pour calculer le flot optique à partir de 2 images successives [3]. La plupart de ces méthodes utilisent l'équation (2.5) comme point de départ.

La méthode la plus traditionnelle consiste à considérer simplement l'équation (2.5) comme un problème d'optimisation à résoudre. On peut alors choisir d'utiliser les équations d'Euler-Lagrange pour déterminer la solution numérique, comme dans [1]. D'autres auteurs ont également proposé une résolution multiobjective de l'équation variationnelle en séparant l'optimisation du terme d'attache-aux-données et de régularisation [12, 11]. Dans toutes ces approches variationnelles, différents termes d'attache-aux-données et de régularisation peuvent être utilisés. Le terme non-linéaire (2.2) plus fidèle à la réalité, mais le terme linéarisé (2.3) plus facile à utiliser dans un contexte d'optimisation. Pour le terme de régularisation, la plupart des auteurs utilise le terme TV[11] associé à la norme L_1 ou L_2 . Enfin, on peut encore appliquer un pré-traitement des images, comme un lissage local gaussien [4].

Plus récemment, les réseaux de neurones et le deep learning ont été explorés pour calculer le flot optique. Une fonction de coût similaire à l'équation (2.5) sert à entraîner le réseau. La plupart des réseaux sont des modèles supervisés. Tel que Flownet [2] qui utilise un réseau de neurones convolutif (CNN) sous la forme d'un encodeur/décodeur afin de prédire le flot optique. Un autre modèle est RAFT [14] qui se distingue par sa capacité à produire des estimations denses et détaillées du flot optique avec une grande précision, même pour des scènes complexes. Enfin, SMOFlow [6] est une approche basée sur un réseau de neurones dédié à l'estimation de mouvement de petits objets.

2.2 Méthode du stage

Le principal problème des méthodes existantes dans le contexte des images de microscopie photonique (fluorescence) est le déplacement de petits objets. Le but de ce stage est de proposer une méthode afin de prendre en compte les petits éléments d'images.

Le problème avec les méthodes déjà existantes se situe au niveau du terme de régularisation. En effet, la régularisation TV utilisée classiquement impose une certaine continuité du flot optique. Cette contrainte est utile pour imposer un mouvement continu sur tout l'ensemble d'un objet en mouvement. Cependant, cette contrainte est trop forte pour gérer les petits objets. En effet, la continuité du flot optique peut créer un effet de « bavure » (*figure 2.2*) autour des objets. Les petits objets apparaissent énormément grossis à cause de cet effet de « bavure ». Un autre problème est le fait qu'un petit objet très proche d'un autre n'est

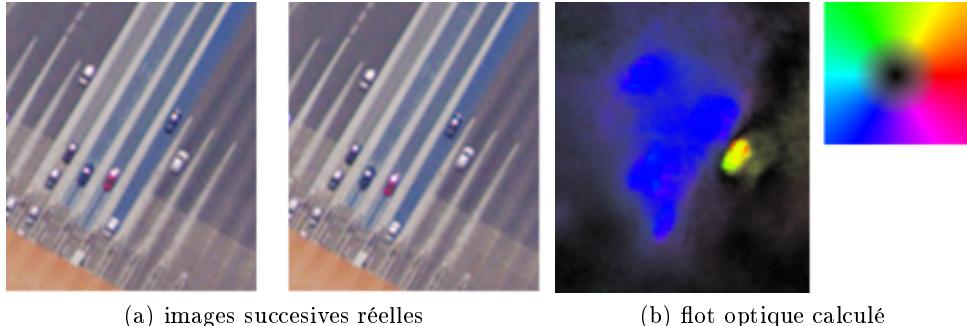


Figure 2.2: Calcul du flot optique à l'aide d'une méthode TV- L_1 [11]. On remarque l'effet de "bavure"(ou surlissage) du flot optique calculé figure (b) par rapport à l'image réelle (a).

pas très bien distingué par les méthodes et peut conduire à un même mouvement moyen calculé (*figure 2.2*) sur l'ensemble des 2 objets. Finalement, si le fond de l'image est fixe, le flot optique va être essentiellement nul, et par effet du régulariseur, les petits objets vont être confondus avec le fond et disparaître (*figure 2.3*) (effet de propagation du mouvement de l'arrière-plan).

Pour contourner ces difficultés, lors de mon stage, j'ai étudié les méthodes variationnelles et, en particulier, j'ai exploré l'idée d'ajouter un terme de parcimonie ("sparse"). Un terme parcimonieux est un terme qui impose au flot optique de tendre en tout point du domaine de l'image vers 0, c'est-à-dire :

$$\forall x \in \Omega, w(x) \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Cette contrainte a pour but de prendre en compte le fait que le fond de l'image soit stable et statique. Une autre idée dans l'utilisation de ce régulariseur est qu'il permettrait de calculer un flot optique plus discontinu, ce qui réduirait les effets de « bavures ». Cela peut paraître contradictoire avec le terme régulariseur TV, cependant c'est assez logique pour de petits objets dont le mouvement ne correspond qu'à quelques pixels de l'image.

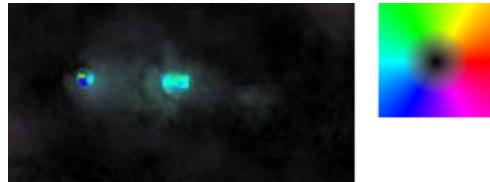
J'ai donc testé un terme « sparse ». Une méthode similaire a déjà été explorée dans un autre problème variationnel en restauration d'images [9]. Le but est donc de faire coexister les 2 termes dans un seul terme de régularisation. Pour cela, il existe différentes manières, mais je me suis concentré sur un terme nommé Sparse Variation (SV):

$$\rho_{reg}(x, w) = \sqrt{\alpha^2(\|\nabla w_1(x)\|_l^l + \|\nabla w_2(x)\|_l^l) + (1 - \alpha)^2 \underbrace{\|w(x)\|_k^k}_{\text{terme "sparse" }}} \quad (2.6)$$

où $\|\cdot\|_i$ correspond à la norme vectorielle L_i . Les normes classiquement utilisées sont les normes L_1 et L_2 , et nous verrons dans les résultats l'importance du choix des normes.



(a) images successives réelles



(b) flot optique calculé

Figure 2.3: Calcul du flot optique à l'aide d'une méthode TV- L_1 [11]. On remarque l'effet d'effacement du flot optique calculé figure (b) sur la troisième voiture rouge par rapport à l'image réelle (a).

Le paramètre $\alpha \in [0; 1]$ contrôle l'importance à accorder à chacun des deux régulariseurs. Ainsi, un poids α proche de 0 implique un flot optique discontinu et proche de 0, alors que lorsque le poids s'approche de 1, on retrouve le terme original TV et donc un champ de vecteur continu.

Ainsi, la nouvelle formule variationnelle du flot optique devient :

$$E(w) = \int_{\Omega} (1-\lambda) \sqrt{\alpha^2(\|\nabla w_1(x)\|_l^l + \|\nabla w_2(x)\|_l^l) + (1-\alpha)^2 \|w(x)\|_k^k} + \lambda \phi(\rho_{data}(x, I, w)) dx dy \quad (2.7)$$

avec ϕ une fonction utilisée pour l'optimisation (le plus souvent une norme $\|\cdot\|$).

Maintenant, pour calculer le flot optique w^* , il faut maintenant minimiser l'énergie (2.7) :

$$w^* = \arg \min_w E(w)$$

Pour cela, on dérive les équations d'Euler Lagrange [1] car notre problème est convexe (voir annexe). Dans notre cas, nous allons essayer de minimiser directement l'énergie $E(w)$ par un algorithme de descente de gradient stochastique.

2.3 Implémentation

Pour implémenter cette méthode, j'ai utilisé python et les bibliothèques openCV, pytorch ou encore numpy.

OpenCV OpenCV est une bibliothèque implementant de nombreuses méthodes pour des problèmes de vision par ordinateur. Elle a d'abord été implémentée en langage C et ensuite en Python. Cette bibliothèque est utile car elle propose différentes méthodes pour calculer le flot optique. Lors de mon stage, elle m'a donc servi comme point de départ pour initier l'implémentation de ma méthode et, plus précisément, pour visualiser en couleur le flot optique. OpenCV est également utile pour charger et pré-traiter les images s'il y a besoin.

pytorch Pytorch est une bibliothèque Python qui permet de construire et d'entraîner des réseaux de neurones. Ce qui nous intéresse est l'implémentation des méthodes d'optimisation utilisées pour entraîner ces réseaux [8]. En effet, l'entraînement d'un réseau consiste à minimiser une fonction coût par rapport à plusieurs paramètres. Ainsi, l'idée est d'utiliser ces méthodes pour minimiser notre énergie (2.7). Vous pouvez trouver plus de précision sur la méthode de différentiation de Pytorch en Annexe.

Implémentation de la méthode

Vous pouvez retrouver l'implémentation complète <https://github.com/theot-student/Sparse-Optical-Flow>. L'implémentation de la méthode se fait en trois étapes (*figure 2.4*):

1. Tout d'abord, on charge les images de la vidéo grâce à OpenCV. Si besoin, on les convertit en niveau de gris, ce qui est nécessaire pour notre optimisation.
2. Ensuite, pour chaque paire d'images de la vidéo, on procède au calcul du flot optique grâce à Pytorch.
3. Finalement, on convertit le champ de vecteur initialement en coordonnées cartésiennes dans des coordonnées polaires afin de visualiser le champ de vecteur colorisé (voir annexe).

Nous allons nous intéresser plus précisément à l'étape du calcul qui consiste à minimiser l'énergie (2.7). Dans un premier temps, nous avons besoin de composants pour implémenter le calcul de notre énergie (*algorithme 1*). Pour cela, j'ai créé 2 composants : une pour le calcul du terme d'attache-aux-données et une pour le calcul du terme de régularisation. Pour ces deux termes, j'ai utilisé les fonctions préexistantes de Pytorch pour les opérations de base. Cela permet d'utiliser facilement la différentiation automatique par la suite.

Notons que le flot optique $w(\cdot)$ défini comme une fonction continue est impossible à implémenter sous cette forme d'un point de vue numérique. Pour y remédier, il faut discréteriser cette fonction comme une fonction associant chaque point de l'image à un vecteur représentant le flot optique en ce point. En langage python, cette fonction w est représentée par un tensor (matrice en pytorch) de dimension $(l \times h \times 2)$ où l et h sont respectivement la longueur et

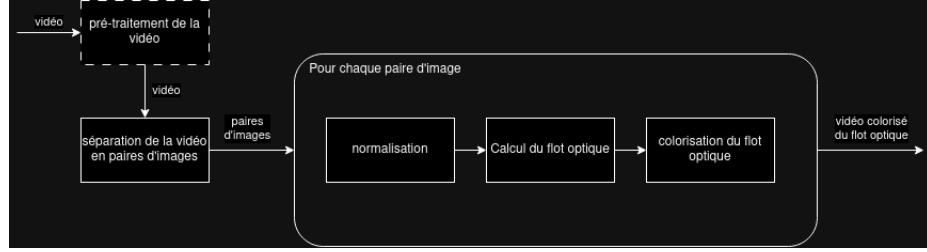


Figure 2.4: Pipeline de la méthode. La case du pré-traitement est en pointillé car c'est une étape qui n'est pas directement implémenté, mais qui reste nécessaire pour avoir de bons résultats.

Algorithme 1 implémentation python du calcul du terme de régularisation sparse (SV_loss) et d'attache-aux-données (data_term)

```

epsilon = 1e-6

def l1Norm(tensor):
    return torch.sqrt(torch.square(tensor) + epsilon)

def SV_loss(motion_field, weighting):
    dw1_x2 = torch.zeros_like(motion_field[:, :, :, :, 0])
    dw1_y2 = torch.zeros_like(motion_field[:, :, :, :, 0])
    dw2_x2 = torch.zeros_like(motion_field[:, :, :, :, 0])
    dw2_y2 = torch.zeros_like(motion_field[:, :, :, :, 0])

    dw1_x2[:, :, :-1, :] = motion_field[:, :, 1:, :, 0] - motion_field[:, :, :-1, :, 0]
    dw1_x2 = l1Norm(dw1_x2)
    dw1_y2[:, :, :-1] = motion_field[:, :, :, 1:, 0] - motion_field[:, :, :, :-1, 0]
    dw1_y2 = l1Norm(dw1_y2)
    dw2_x2[:, :, :-1, :] = motion_field[:, :, 1:, :, 1] - motion_field[:, :, :-1, :, 1]
    dw2_x2 = l1Norm(dw2_x2)
    dw2_y2[:, :, :-1] = motion_field[:, :, :, 1:, 1] - motion_field[:, :, :, :-1, 1]
    dw2_y2 = l1Norm(dw2_y2)

    sv = torch.sqrt((weighting * weighting * (dw1_x2 + dw1_y2 + dw2_x2 + dw2_y2)) \
        + ((1 - weighting) * (1 - weighting) * (l1Norm(motion_field[:, :, :, :, 0]) \
        + l1Norm(motion_field[:, :, :, :, 1]))) + epsilon)
    return sv

def data_term(motion_field, du_x, du_y, du_t):
    dterm = torch.sqrt(torch.square(du_x * motion_field[:, :, :, :, 0] \
        + du_y * motion_field[:, :, :, :, 1] + du_t) + epsilon)
    return dterm

```

la hauteur de l'image et la dernière dimension représente les coordonnées cartesiennes du vecteur.

Pour calculer le gradient de w et les dérivées partielles dont nous avons besoin pour calculer l'énergie (2.7) et le terme linéarisé (2.3), j'ai utilisé la méthode des différences finies. Les choix effectués pour les différences finies sont reportés en annexe, tout comme l'approximation utilisée pour les normes.

Après avoir calculé la fonction de coût, il reste à piloter l'optimisation Pytorch en utilisant la fonction « `optimizer.step()` » de Pytorch. Notons qu'un "scheduler" a également été implémenté afin de pouvoir changer le "learning rate", c'est-à-dire le coefficient de la descente de gradient, au fur et à mesure de l'optimisation.

Enfin, on répète ces 2 étapes de calcul de l'énergie et d'optimisation. La boucle s'arrête lorsqu'on observe 5 fois d'affilée le critère suivant : la différence entre 2 énergies successives est plus petite qu'une précision donnée en paramètre.

Dans mon implémentation, les principaux paramètres modifiables sont :

- le paramètre de régularisation λ qui permet de jouer sur l'équilibre entre le terme d'attache-aux-données et de régularisation.
- le paramètre α qui permet d'équilibrer le terme TV et le terme de parcimonie.
- la précision que l'on cherche à atteindre dans l'optimisation.

On peut également influer sur le "learning rate" de l'optimisation qui permet de modifier la vitesse de convergence de l'algorithme. Cependant, il faut faire attention en modifiant ce paramètre, car il peut empêcher d'atteindre la solution optimal s'il est mal choisi.

2.4 Problèmes rencontrés

Dans cette partie, je vais présenter quelques problèmes rencontrés lors de l'implémentation de la méthode. Je ne présente ici que les problèmes majeurs. Un résumé des problèmes mineurs est reporté en Annexe.

Initialisation du calcul du flot optique

L'un des principaux problèmes rencontrés est d'initialiser le flot optique. En effet, à l'inverse des problèmes inverses de débruitage d'image, le calcul du flot optique ne possède pas d'initialisation triviale. L'initialisation est importante car elle impacte la convergence vers la solution finale. Dans les tests que j'ai effectués, j'ai essayé différentes initialisations :

- initialisation de w à 0
- initialisation de w de façon aléatoire entre 0 et β , avec $\beta > 0$
- initialisation de w de façon aléatoire entre $-\beta$ et β .

En testant ces trois initialisations avec différentes valeurs pour β , j'ai remarqué que celle qui converge le plus rapidement et le plus sûrement vers une solution correcte est l'initialisation à 0. Ceci est logique dans notre cas, car, comme la majeure partie de l'image est statique, le flot optique espéré est principalement nul. Cependant, utiliser cette initialisation implique d'avoir une approximation robuste des différentes normes qui permet de calculer le gradient en 0. Il faut également noter que l'initialisation du flot optique change de manière proportionnelle le résultat obtenu sur tout le flot optique, ce qui est un problème détaillé dans la section 2.4.6.

Choix des normes

Pour le calcul des termes de régularisation et d'attache-aux-données, il faut choisir les normes à considérer. Les plus communes sont les normes L_1 et L_2 . Ces 2 normes sont implantables, cependant elles ont certains avantages et inconvénients.

Tout d'abord, la norme L_2 est facilement différentiable (même en 0). Ceci est un atout majeur dans notre optimisation, car notre champ de vecteurs est principalement nul et donc on doit seulement s'assurer que la dérivée de la norme en 0 existe. Cependant, le gros désavantage de la norme L_2 est qu'elle a tendance à surinterpréter les petites et grandes valeurs. Par exemple, si on a une valeur z de l'ordre de 10^{-3} , alors on a $\|z\|_2^2$ qui est de l'ordre de 10^{-6} . Théoriquement, cela ne pose pas de problème. Cependant, informatiquement, le calcul avec des valeurs très faibles n'est pas toujours très précis.

En ce qui concerne la norme L_1 , elle ne possède pas le désavantage de la norme L_2 , cependant elle n'est pas différentiable en 0. Ainsi, pour calculer la dérivée en 0, il faut utiliser une approximation de cette norme (voir Annexe).

Choix des termes d'attache-aux-données

Un problème important du flot optique par approche variationnelle est le choix du terme d'attache-aux-données à utiliser. Pour cela, on peut par exemple utiliser les différents termes présentés dans la section 2.1.2. Dans notre cas, nous nous intéressons aux termes non linéaires (2.2) et au terme linéarisé (2.3). Tout d'abord, regardons le terme linéarisé (2.3). Le calcul de ce terme requiert le calcul des dérivées partielles spatiale et temporelle de l'image. Pour cela, comme dit précédemment, on utilise la méthode des différences finies (voir Annexe) afin de calculer une approximation. De plus, la bibliothèque Pytorch permet

de calculer le gradient nécessaire pour l'algorithme de gradient descente. Plus précisément, la différentiation automatique de Pytorch permet de calculer chacune des dérivées partielles présentes dans le calcul du terme d'attache aux données et des différences finies, ce qui représente en fait la dérivée partielle de l'approximation de la dérivée partielle de l'image. Malgré toutes ces approximations, expérimentalement, le calcul fonctionne.

Pour le terme non linéaire (2.2), il faut dans un premier temps calculer le terme correspondant au déplacement du pixel sur la deuxième image : $I(x + w(x), t + 1) = I_2(x + w(x))$. Le problème est que notre solution w n'est pas exactement dans la discréétisation de pixel de l'image. Plus précisément, pour un pixel x donné, les composantes de $w(x) \in \mathbb{R}^2$ ne sont pas en coodonnées entières et donc $x + w(x)$ non plus et ne correspondent donc pas à un pixel existant de notre image. Une solution est d'utiliser une interpolation de l'image afin de calculer ce terme. Dans mon implémentation, j'ai utilisé une interpolation « au plus proche voisin » car elle demande un temps de calcul relativement faible. En ce qui concerne l'optimisation, Pytorch ne peut pas dériver automatiquement le terme non linéaire (car l'image n'est pas une fonction Pytorch). Il faut alors préciser une dérivée partielle explicite de ce terme pour que Pytorch puisse procéder à la différentiation. Ce qu'il faut noter, c'est que pour calculer cette dérivée, il faut réutiliser les différences finies utilisées dans le terme linéarisé. Par conséquent, on ne gagne pas forcément en précision. Vous trouverez plus d'explications sur ce terme et l'implémentation en Annexe.

Calcul des différences finies

Le choix des méthodes pour les différences finies a été important lors de ces recherches. Vous pouvez trouver plus de détails et des exemples pour les différentes méthodes en Annexe. En ce qui concerne l'implémentation, au final, j'ai utilisé un modèle centré pour le terme d'attache aux données et décentré pour le terme de régularisation.

Etude de l'intensité constante lumineuse / préprocessing

Un autre problème important du flot optique dans notre cas est le problème de l'intensité lumineuse qui dépend très fortement des conditions dans lesquelles les images sont acquises. En effet, entre deux images, on peut percevoir parfois une luminosité totalement différente d'une image à l'autre. Ce phénomène pose problème, car notre algorithme repose sur l'intensité lumineuse de 2 images successives pour le calcul du flot optique. Ainsi, il est important de normaliser les 2 images.

Par ailleurs, les images récupérées sont très souvent bruitées. En particulier, sur nos images de microscopie où le fond est statique , celui-ci est très sensible au bruit. Cela pose problème, car ce bruit peut conduire à un calcul erroné de mouvement par notre algorithme (*figure*

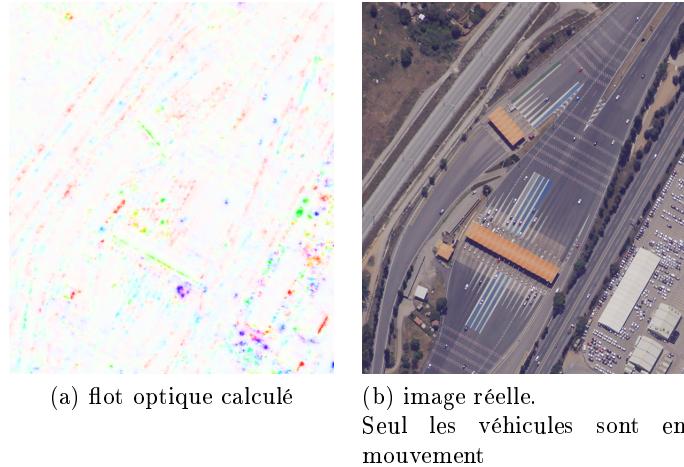


Figure 2.5: Exemple du problème du fond bruité dans le calcul du flot optique

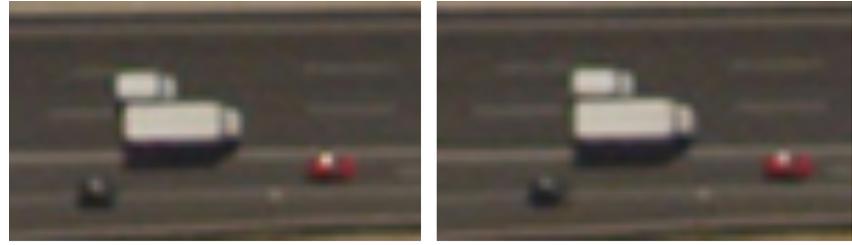
2.5). Il est donc important de procéder à un prétraitement des images afin de débruiter le fond.

Enfin, certains éléments ou objets des images sont totalement lisses au niveau de l'intensité lumineuse, c'est-à-dire que l'intensité lumineuse est constante sur tout l'objet (en particulier sur les images en niveau de gris). Cela peut alors poser problème dans notre méthode. En effet, comme sur la Figure 2.6, on peut voir que, pour un objet dont l'intensité lumineuse est constante et dont le mouvement est moindre, le flot optique calculé au centre de l'objet est nul. Il arrive aussi que le flot optique calculé sur le contour de l'objet pointe par erreur vers le centre de l'objet. Une explication plus théorique est donnée en Annexe.

Pour résoudre ces problèmes, il faut procéder à un pré-traitement qui permet de pallier à cette constance de l'intensité lumineuse. Pour cela, on peut appliquer tout d'abord une méthode de soustraction du fond. Cette méthode consiste à appliquer un filtre médian temporel sur la séquence d'images pour «d'effacer» les petits objets en mouvements. Ensuite, on peut soustraire chaque image de la séquence et l'image filtrée, ce qui fait ressortir les petits objets effacés par le filtre. Ensuite, il suffit de convertir l'image en niveaux de gris et d'appliquer un filtre Gaussien. Le filtre gaussien permet de transformer les petits objets en des taches «gaussiennes» sur l'image (*figure 2.7*) et d'atténuer partiellement la constance de l'intensité lumineuse.

Normalisation du flot optique

Lorsque l'on utilise le terme d'attaché-aux-données linéarisé (2.3), le calcul du flot optique dépend seulement des dérivées partielles des images. Ce calcul effectué par différences



(a) Deux images consécutives de véhicules se déplaçant. On remarque que les véhicules blanc ont une intensité lumineuse quasiment constante.



(b) calcul du flot optique de (a). On remarque que le flot optique calculé est nul au centre des véhicules

Figure 2.6: Exemple du problème du centre des objets



Figure 2.7: images consécutives d'un péage en vue satellite. Les images ont été pré-traitée comme expliqué dans la partie 2.4.6

finies dépend de l'intensité lumineuse des pixels des images. Ainsi, le flot optique dépend fortement de la normalisation des images effectuée avant le calcul (voir section 2.4.3). De ce fait, le flot optique calculé n'est pas le flot optique réel (par rapport à la mesure des pixels de l'image), mais un flot optique proportionnel à celui-ci, dépendant de la normalisation. On pourrait alors se demander s'il est possible de retrouver le flot optique réel à partir du flot optique calculé. Cependant, ce problème reste très ouvert et n'a pas été traité au cours de mon stage.

2.5 Résultats expérimentaux

Dans cette partie, je présente les résultats obtenus avec notre méthode. Pour toutes ces expérimentations, la lut de couleur associée au flot optique colorisé calculé par notre algorithme « sparse » est présentée dans la *figure 2.8* .

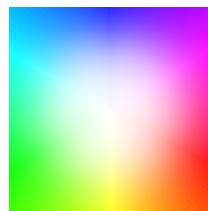


Figure 2.8: lut de couleur utilisé pour la colorisation du flot optique dans notre méthode parcimonieuse

Données utilisées Pour tester la méthode, j'ai utilisé différents types d'images simulant le déplacement de petits objets. Tout d'abord j'ai utilisé 2 courtes vidéos en couleurs montrant une vue satellite d'un réseau routier. Sur ces vidéos, les véhicules en déplacement simulent les petits objets que l'on peut retrouver sur nos images microscopiques. Dans un second temps, j'ai utilisé une image créée expérimentalement qui montre des extrémités de microtubules en mouvement. Ces vidéos sont générées artificiellement et présentent des dynamiques stochastiques d'extrémités de microtubules

2.5.1 Comparaisons normes et initialisation

Dans cette partie, je présente quelques comparaisons entre les différentes normes et initialisations possibles (voir sections 2.4.1 et 2.4.2). Ici, j'ai utilisé trois énergies (2.5) différentes : en combinant des normes L_1 ou L_2 pour les deux termes ou une norme L_1 (terme de régularisation) et une norme L_2 (terme d'attache-aux-données). Pour ce qui est de l'initialisation, j'en ai également testé 3 différentes : initialisation à 0, de manière aléatoire entre -0,1 et

initialisation \ norme	l_1	l_2	mix
0			
$[-0,1;0,1]$			
$[-1;1]$			

Table 1: Tableau de comparaison des différentes normes et initialisations. Le calcul a été effectué sur les images de la *Figure 2.7* avec les paramètres suivants : $\lambda = 0.7$, $\alpha = 0.9$, précision à 10^{-10} .

0,1 et de manière aléatoire entre -1 et 1. Les résultats sont présentés dans le *tableau 1*. Nous pouvons remarquer que les méthodes utilisant la norme L_2 fonctionnent pour une initialisation plus différente de 0 (i.e., champ nul), ce qui n'est pas le cas si on considère seulement la norme L_1 . Cependant, nous pouvons observer également que, par rapport à la norme L_1 , les deux autres méthodes sont moins précises, car certains véhicules sont confondus avec le fond de la scène. Notons que les trois méthodes fonctionnent correctement pour une initialisation à 0. Cependant, dès que l'initialisation n'est plus nulle, nous observons des problèmes pour le calcul avec un flot optique ne correspondant absolument pas à la réalité. Les temps d'exécution des différentes méthodes sont présentés dans le *tableau 2*. Le temps de calcul est assez faible pour toutes les méthodes quand l'initialisation est à 0. Cependant, on remarque que le calcul est plus rapide pour une initialisation entre -0,1 et 0,1 dès que la norme L_2 entre en jeu. Ceci est très visible, notamment pour la méthode combinant les normes L_1 et L_2 dont le temps de calcul est 2 fois moins long que pour la méthode L_1 avec l'initialisation à 0. Cependant, comme vu précédemment, les résultats sont moins précis avec cette méthode. Pour toutes les expériences suivantes, j'ai donc décidé d'utiliser la norme L_1 avec l'initialisation à 0.

initialisation \ norme	l_1	l_2	mix
0	14,89	28,38	14,40
$[-0.1, 0.1]$	40,00	26,76	7,036
$[-1, 1]$	37,45	38,40	37,50

Table 2: Comparaison des temps de calcul (en secondes) des différentes méthodes avec les même paramètres que le *tableau 1*

2.5.2 Analyse de l'influence du terme de parcimonie

Dans un premier temps, j'ai effectué quelques expériences pour comprendre l'impact du terme de parcimonie sur le calcul du flot optique. Pour cela, comme illustré sur la *figure 2.9*, j'ai effectué le calcul du flot optique sur une image satellite pour différentes valeurs de α . On remarque alors que lorsque α est faible, on retrouve l'effet de « bavure » provoqué par le terme TV. À l'inverse, lorsque le terme de parcimonie est important, le mouvement des objets est de moins en moins perceptible jusqu'à disparaître complètement. Ce phénomène met ainsi en évidence l'effet attendu par l'utilisation du terme de parcimonie qui permet de réduire l'effet de « bavure ». On peut alors tenter de déterminer des paramètres « parfaits » qui permettent la meilleure approximation du flot optique.

2.5.3 Comparaisons des pré-traitements appliqués sur les images

Pour cette expérience, on va comparer le calcul du flot optique par notre méthode sur les images de base avec les images prétraitées par la méthode décrite dans la section 2.4.6 . On présente les résultats sur la figure 2.10 (image satellite d'un péage). D'autres images sont présentées en Annexe. On remarque que le flot optique calculé pour les images sans prétraitement ne produit pas de bons résultats pour les petits objets en comparaison des images prétraitées. Nous pouvons cependant remarquer que le temps de calcul est 2 fois moins long pour l'image qui n'a pas subie de prétraitement.

2.5.4 Comparaison avec IPOL et SMOFlow

Ensuite, j'ai comparé ma méthode variationnelle avec d'autres méthodes comme la méthode variationnelle TV- L_1 qui associe la régularisation TV avec la norme L_1 ou SMOFlow, une méthode utilisant un réseau de neurones et une régularisation sparse. On peut observer une comparaison des différentes méthodes sur l'image sur la figure 2.11. On observe alors que notre méthode « sparse » performe mieux sur les images satellites de réseaux routiers que la méthode TV- L_1 notamment au niveau des véhicules très proches. La méthode variationnelle parcimonieuse et SMOFlow ont des performances assez similaires.

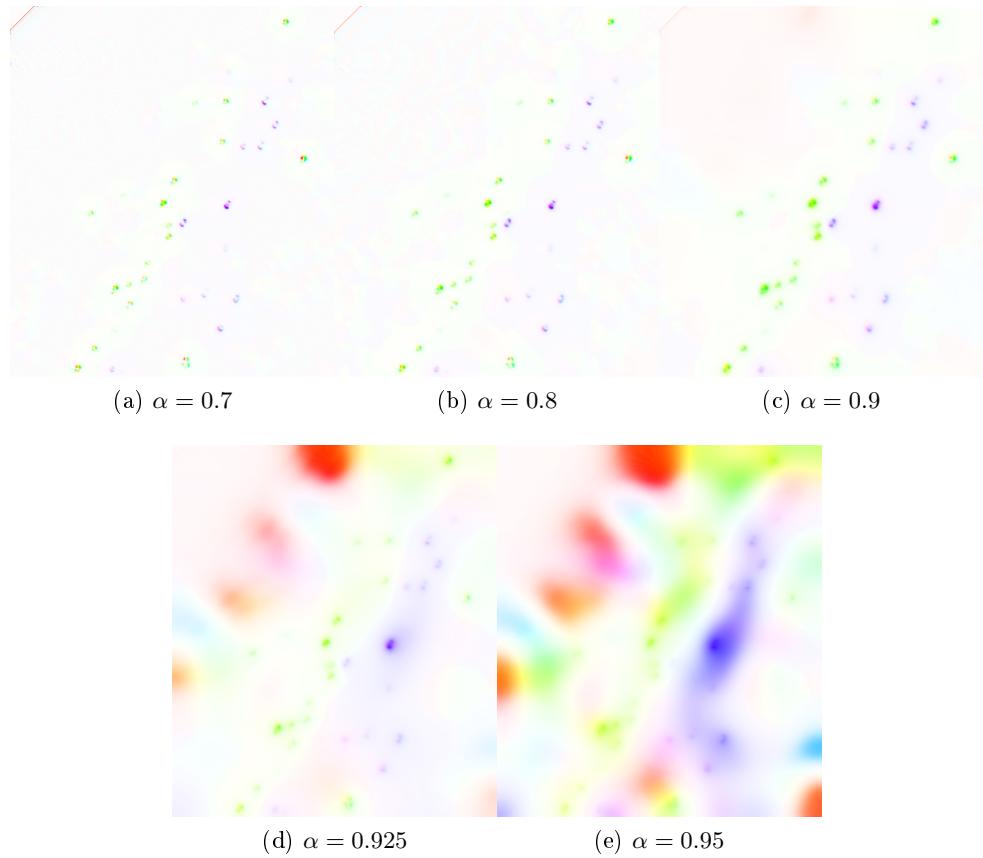
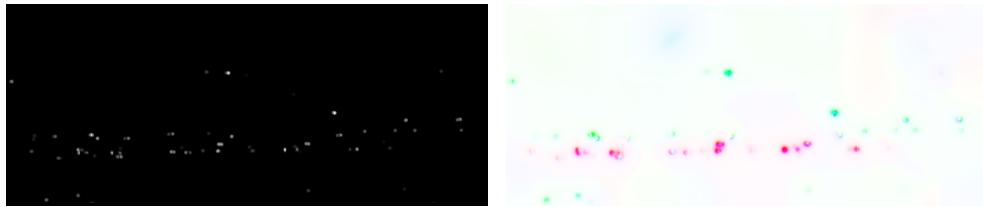


Figure 2.9: Comparaison des flot optiques calculés en fonction du paramètre α contrôlant le niveau de parcimonie. On utilise les images de la *figure 2.7* afin de calculer le flot optique. Les autres paramètres sont fixes : $\lambda = 0.7$ et la précision est de 10^{-10} . La lut de couleur est celle présentée sur la en *figure 2.8*



(a) Avec pré-traitement. Temps de calcul : 14,87 secondes



(b) sans pré-traitement. Temps de calcul : 7,35 secondes

Figure 2.10: Comparaison du calcul du flot optique avec et sans pré-traitement. Les paramètres sont : $\lambda = 0.7$, $\alpha = 0.9$, précision à 10^{-10} . A gauche : la première image sur laquelle le flot optique est calculé. A droite : le flot optique calculé.

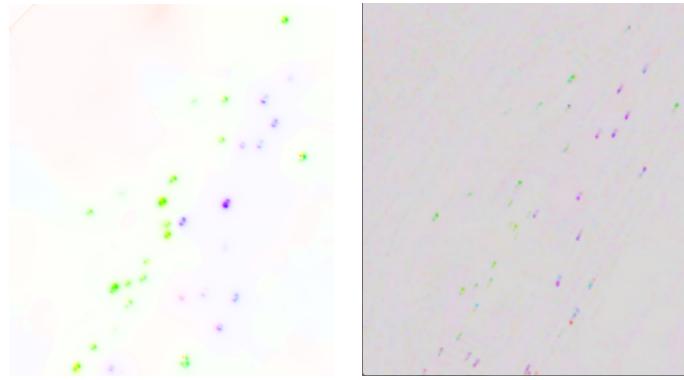
2.5.5 Expérience sur des image de microscopie

Pour cette dernière expérience, j'ai testé notre algorithme sur une vidéo présentant des extrémités de microtubules animés de mouvements stochastiques. Contrairement aux véhicules, ces petites entités mobiles, assimilables à des spots Gaussiens anisotropes, sont agités de mouvements erratiques sans direction privilégiée, et peuvent disparaître dans le fond. Le calcul du flot optique devient alors beaucoup plus difficile. La Figure 2.12 présente le flot optique calculé sur ces images. On remarque que le flot optique arrive à repérer la plupart des extrémités de microtubules. Cependant, nous pouvons également observer des effets de « bavures » qui font que plusieurs extrémités de microtubules sont fusionnées au sein d'une même région. De plus, le flot optique calculé n'est pas toujours orienté dans la bonne direction.

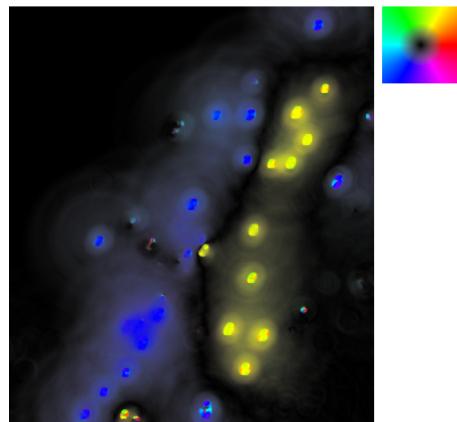
2.6 Perspectives

Amélioration possible des résultats

Plusieurs actions pourraient être menées pour compléter les résultats. Tout d'abord, nous pourrions utiliser la DFD afin de valider nos résultats. Un autre moyen serait d'annoter manuellement le flot optique sur chacune des images afin de comparer cette vérité au flot calculé par notre algorithme.

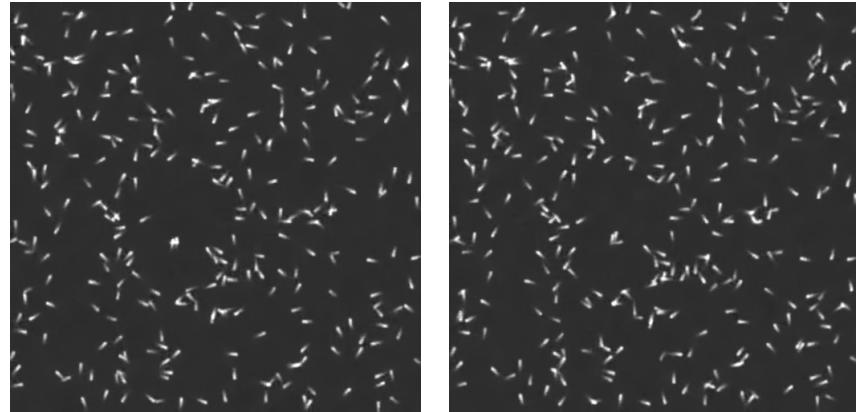


(a) Méthode sparse.
Paramètres : $\lambda = 0.7$,
 $\alpha = 0.9$ et précision à 10^{-10}

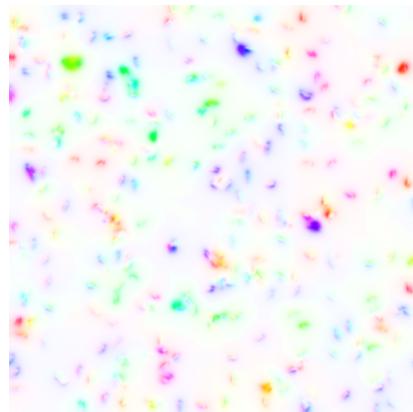


(c) Méthode IPOL TV- L_1 . Cette méthode utilise une lut de couleur spéciale (en haut à gauche de l'image)

Figure 2.11: Comparaison des 3 énergies (2.5) sur les images de péage (*figure 2.7*)



(a) images traitées



(b) flot optique calculé

Figure 2.12: calcul du flot optique sur des images d'extrémités de microtubule avec les paramètres suivants : $\lambda = 0.7$, $\alpha = 0.9$, précision à 10^{-10} .

Méthode DFD

Une idée qui a également été abordée lors de mon stage est d'utiliser le DFD comme terme d'attache-aux-données. Pour rappel, le Displaced Frame Differences (DFD) entre deux images successives dont on connaît le flot optique consiste à reconstruire la deuxième image à l'aide de la première image et du flot optique. Ainsi, on peut effectuer la différence entre cette image reconstruite et l'image réelle. Cette différence est censée être nulle si le flot optique est parfait. Une méthode pourrait donc être d'utiliser cette idée comme terme pour notre méthode variationnelle. Pour ce stage, j'ai simplement essayé de poser le problème de façon théorique (voir annexe), mais je n'ai pas étudié l'implémentation possible de cette méthode qui représente un problème en soi.

Choix des paramètres

Une des difficultés de nos méthodes réside dans le choix des paramètres de régularisation et de parcimonie. En effet, le flot optique calculé dépend fortement de ces paramètres et, pour un flot optique s'approchant du flot optique réel, il faut bien choisir ces paramètres. Le problème étant que ces paramètres dépendent de la vidéo utilisée. Lors de mon stage, j'ai passé beaucoup de temps à chercher de bons paramètres pour calculer le flot optique. Ainsi, il serait intéressant d'avoir une méthode permettant de trouver les paramètres optimaux. Pour cela, j'ai étudié le principe du min-max dans la recherche des paramètres optimaux, comme dans [5]. Ce principe marche habituellement pour des problèmes inverses faiblement mal posés. Malheureusement, le problème initial du flot optique est un problème mal posé et ne permet pas d'utiliser le principe du min-max.

3 Conclusion

Lors de mon stage, j'ai donc étudié et implémenté une nouvelle méthode variationnelle pour le calcul du flot optique pour de petits objets. Cette méthode est caractérisée par l'utilisation d'un nouveau terme de parcimonie permettant de repérer plus facilement les petits objets. Nous avons vu que cette méthode permettait d'obtenir de bons résultats pour des images expérimentales. De plus, cette méthode profite de l'efficacité de l'optimisation sous Pytorch. Ce stage m'a permis de m'initier à l'étude des problèmes variationnels et aux problèmes de traitement d'images. De plus, il m'a permis d'approfondir mes compétences en programmation et en l'utilisation des bibliothèques Python, notamment la bibliothèque Pytorch, très utile à l'heure actuelle. Pour finir, une suite possible de mes recherches serait de tester l'utilisation du terme de parcimonie sur d'autres méthodes d'optimisation plus robustes ou bien de trouver une méthode de recherche des paramètres optimaux.

References

- [1] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 25–36, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [2] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.
- [3] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.
- [4] Denis Fortun, Noémie Debroux, and Charles Kervrann. Spatially-variant kernel for optical flow under low signal-to-noise ratios: application to microscopy. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 42–48, 2017.
- [5] M.A. Gennert and A.L. Yuille. Determining the optimal weights in multiple objective function optimization. In *[1988 Proceedings] Second International Conference on Computer Vision*, pages 87–89, 1988.
- [6] Sarra Khairi, Etienne Meunier, Renaud Fraisse, and Patrick Bouthemy. Efficient local correlation volume for unsupervised optical flow estimation on small moving objects in large satellite images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 440–448, June 2024.

- [7] Sarra Khairi, Etienne Meunier, Renaud Fraisse, and Patrick Bouthemy. Efficient local correlation volume for unsupervised optical flow estimation on small moving objects in large satellite images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 440–448, 2024.
- [8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017. Accessed: 2024-08-08.
- [9] Sylvain Prigent, Hoai-Nam Nguyen, Ludovic Leconte, Cesar Augusto Valades-Cruz, Bassam Hajj, Jean Salamero, and Charles Kervrann. Spitfir (e): a supermaneuverable algorithm for fast denoising and deconvolution of 3d fluorescence microscopy images and videos. *Scientific Reports*, 13(1):1489, 2023.
- [10] PyTorch. Pytorch optimizers, 2024. Accessed: 2024-08-08.
- [11] Javier Sanchez Perez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3:137–150, 2013.
- [12] Frank Steinbrücker, Thomas Pock, and Daniel Cremers. Large displacement optical flow computation without warping. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1609–1614. IEEE, 2009.
- [13] Sebastian Sylwan. The application of vision algorithms to visual effects production. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Computer Vision – ACCV 2010*, pages 189–199, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [14] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 402–419. Springer, 2020.
- [15] Wikipedia. Flux optique — wikipedia, l’encyclopedie libre, 2020. [En ligne; Page disponible le 20-mars-2020].

A Linéarisation de l'équation 2.2

Soit $x \in \Omega$ un pixel de l'image. Pour linéariser l'équation du flot optique, il faut passer par l'équation non linéaire (2.2) :

$$I(x + w(x), t + 1) - I(x, t) = 0.$$

En posant $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ et $w(x) = \begin{pmatrix} w_1(x) \\ w_2(x) \end{pmatrix}$, on peut alors réécrire :

$$I(x + w(x), t + 1) = I(x_1 + w_1(x), x_2 + w_2(x), t + 1).$$

En considérant des déplacements faibles, on peut alors utiliser une série de Taylor :

$$I(x_1 + w_1(x), x_2 + w_2(x), t + 1) = I(x_1, x_2, t) + \frac{\partial I}{\partial x_1} w_1(x) + \frac{\partial I}{\partial x_2} w_2(x) + \frac{\partial I}{\partial t} + \underbrace{\dots}_{\text{terme d'ordre supérieurs}}.$$

Alors, en tronquant cette série, on obtient une version linéarisée de l'équation 2.2 :

$$\frac{\partial I}{\partial x_1} w_1(x) + \frac{\partial I}{\partial x_2} w_2(x) + \frac{\partial I}{\partial t} = 0.$$

B Problème convexe

On part de l'équation (2.3) :

$$\frac{\partial I}{\partial x_1} w_1(x) + \frac{\partial I}{\partial x_2} w_2(x) + \frac{\partial I}{\partial t} = 0.$$

On note : $\frac{\partial I}{\partial x_1} = I_{x_1}$, $\frac{\partial I}{\partial x_2} = I_{x_2}$ et $\frac{\partial I}{\partial t} = I_t$.

Supposons dans notre cas que l'on utilise la norme $\|\cdot\|_2^2$, ce qui correspond ici à mettre au carré notre terme d'attache aux données au pixel $x \in \Omega$:

$$\begin{aligned} \rho_{data}(w_1, w_2) &= (I_{x_1} w_1 + I_{x_2} w_2 + I_t)^2 \\ &= I_{x_1}^2 w_1^2 + 2I_{x_1} I_{x_2} w_1 w_2 + I_{x_2}^2 w_2^2 + I_t^2 + 2I_{x_1} I_t w_1 + 2I_{x_2} I_t w_2. \end{aligned}$$

On peut alors calculer la hessienne :

$$\begin{aligned} H &= \begin{pmatrix} \frac{\partial^2 \rho_{data}}{\partial w_1^2} & \frac{\partial^2 \rho_{data}}{\partial w_1 w_2} \\ \frac{\partial^2 \rho_{data}}{\partial w_2 w_1} & \frac{\partial^2 \rho_{data}}{\partial w_2^2} \end{pmatrix} \\ &= \begin{pmatrix} 2I_{x_1}^2 & 2I_{x_1} I_{x_2} \\ 2I_{x_1} I_{x_2} & 2I_{x_2}^2 \end{pmatrix}. \end{aligned}$$

On peut alors montrer que cette matrice est semi-définie positive, ce qui montre la convexité de notre fonction. On peut également montrer la convexité pour d'autres normes et pour les termes de régularisation.

C Différentiation automatique pytorch

Pytorch est une bibliothèque capable d'effectuer l'optimisation automatique d'une énergie. Plus théoriquement, pour minimiser une fonction linéaire, Pytorch utilise une méthode appelée différentiation automatique. Cela consiste à utiliser la règle de la chaîne afin de calculer le gradient de notre fonction Coût par rapport à chacun des paramètres. Tout d'abord, il faut préciser à Pytorch pour quelles variables nous voulons calculer le gradient. Pour cela, Pytorch enregistre chacune des opérations effectuées pour calculer l'énergie à optimiser. Il va pouvoir construire le graphe de calcul, qui répertorie l'ordre des opérations et les variables utilisées pour ces calculs (voir *figure C.1*). Ainsi, Pytorch peut remonter le graphe afin de calculer chacune des dérivées partielles grâce à la règle de la chaîne. Notons, que l'on doit utiliser des fonctions qui sont directement implémentées dans Pytorch pour que la différentiation soit automatique. Sinon, on doit lui préciser un calcul explicite de la dérivée partielle pour qu'il puisse la calculer.

Grâce à cela, on peut alors utiliser une méthode de descente de gradient pour minimiser notre fonction. Vous pouvez trouver plus d'informations sur la différentiation automatique de Pytorch ici [8].

D Colorisation

Pour coloriser notre flot optique, on utilise la norme de couleur HSV ou Teinte Saturation Valeur. Cette norme permet de coloriser facilement un champ de vecteur en repaire polaire. Pour cela, il faut tout d'abord passer notre champ de vecteur en coordonnées polaires. Ensuite, on utilise les paramètres de teinte et de saturation pour stocker respectivement l'angle et le rayon de nos vecteurs. Il faut penser, à normaliser les angles et les rayons entre 0° et 180° pour la teinte et entre 0 et 255 pour la saturation. De plus, pour observer une plus nettement la différence entre les mouvements et ne pas garder les mouvements trop petits qui pourraient correspondre à du bruit, j'ai utilisé la fonction sigmoïde suivante sur les rayons :

$$f(x) = \frac{255}{1 + e^{0.03(-x+127.5)}}$$

Enfin, il ne reste qu'à enregistrer notre champ de vecteurs comme une image en prenant soin de convertir notre norme HSV en RGB avec la bibliothèque OpenCV.

E approximation des normes

En ce qui concerne les normes, j'ai utilisé une approximation différentiable des normes L_1 et L_2 :

$$\begin{aligned}\|x\|_1 &= \sum_i |x_i| \simeq \sum_i \sqrt{x^2 + \epsilon^2} \\ \|x\|_2 &= \sqrt{\sum_i x_i^2} \simeq \sqrt{(\sum_i x_i^2) + \epsilon^2}\end{aligned}$$

avec ϵ représentant l'epsilon machine, c'est-à-dire le plus petit nombre positif différent de 0 qui peut s'écrire informatiquement. Cette approximation est différentiable, car comme $\epsilon > 0$, la plus petite valeur possible pour la norme est $\sqrt{\epsilon^2} = \epsilon$ et donc la dérivée en 0 est bien définie.

F Différences finies

Pour calculer les dérivées partielles et les gradients présents dans notre énergie, j'ai utilisé la méthode des différences finies. Cependant, il existe différents choix possibles pour le calcul

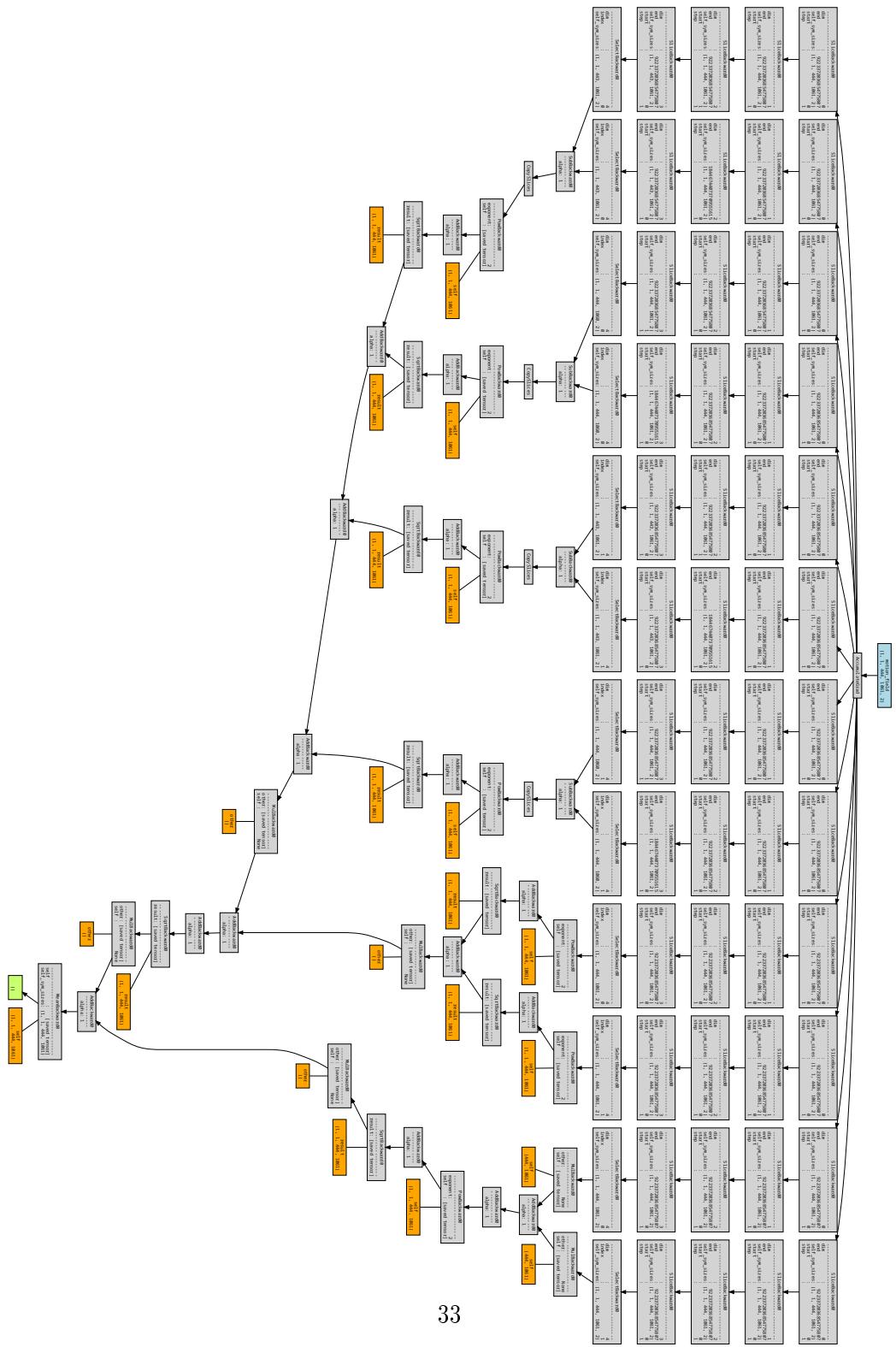


Figure C.1: Graphe d'opérations du calcul de l'énergie (2.7)

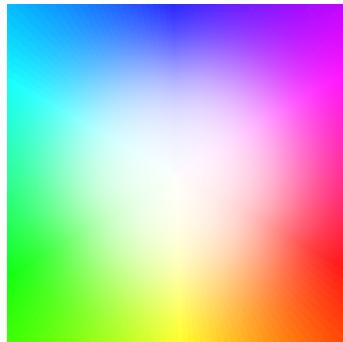


Figure D.1: lute de couleur. On observe que la zone blanche au centre est assez grande pour éviter d'afficher les très petits déplacements induits par le bruit.



Figure F.1: exemple de flot optique calculé avec les dérivées centrées (la couleur est différente de la lute de couleur habituelle mais ce n'est pas important)

des différences finies. Les plus connus sont les modèles centrés : $f'(x) = \frac{f(x+h)-f(x-h)}{2h}$ et décentré : $f'(x) = \frac{f(x+h)-f(x)}{h}$. Théoriquement, le terme centré est censé être plus précis. Les deux modèles se comportent légèrement différemment expérimentalement. Pour le terme d'attache-aux-données ρ_{data} , le changement de modèle n'influence pas trop le résultat, et on peut donc utiliser le modèle centré. Pour ce qui est du terme de régularisation, et notamment de la régularisation TV, il y a quelques différences entre les deux modèles. On peut voir sur les *figures F.1 et F.2* une différence au niveau de la continuité de la couleur. En effet, avec les dérivées centrées, on observe une sorte de quadrillage sur le flot optique. Ceci pourrait être dû au fait que le schéma centré prend en compte seulement les pixels adjacents au pixel actuel. Ainsi, les dérivées calculées dépendent des points adjacent, ce qui correspond à un quadrillage de l'image. Pour les dérivées décentrées, le flot optique est beaucoup plus « smooth ». Nous allons donc utiliser les dérivées décentrées dans nos calculs afin d'éviter le phénomène de quadrillage. Le seul problème des dérivées décentrées réside dans le choix du sens dans lequel on les calcule. Ce choix, pourrait théoriquement avoir une influence sur le calcul du flot optique en favorisant le calcul de certains mouvements. Cependant, expérimentalement, cela ne s'est pas vérifié.



Figure F.2: exemple de flot optique calculé avec les dérivées décentrées

G Problèmes rencontrés

Algorithme d'optimisationFlot optique/flux optique : Le flux optique est le mouvement

Pour l'optimisation avec Pytorch, il existe de nombreuses méthodes [10]. Les 2 plus connues sont les algorithmes SGD et ADAM qui sont des algorithmes de descentes de gradients stochastiques. Lors de ce stage, j'ai testé les 2 méthodes dans notre problème du flot optique. Dans les faits, l'algorithme SGD a une convergence plus « douce » que ADAM. Dans notre cas, après expérience, l'algorithme ADAM semble converger plus rapidement et plus précisément que l'algorithme SGD.

Problème avec la constance de l'intensité lumineuse

Pour expliquer ce problème, prenons un exemple avec le terme non linéaire (2.2) : soit $x \in \Omega$ un pixel d'un objet sur l'image 1. Soit x^* le pixel correspondant au mouvement réel de l'objet sur l'image 2. On peut constater que les pixels ont bien la même intensité. Cependant, le pixel \bar{x} de l'image 2 positionné en x a également la même intensité, car c'est encore un point de l'objet. Alors, les 2 paires (x, x^*) et (x, \bar{x}) , qui correspondent respectivement au mouvement réel de l'objet et à un mouvement nul, vérifient bien l'équation (2.2). Cependant, l'algorithme de calcul du flot optique préférera un mouvement nul du fait de l'initialisation et du terme de régularisation parcimonieuse. Ainsi, le flot optique calculé est correct sur les bords de l'objet, mais nul en son centre. Habituellement, le terme de régularisation TV permet de limiter ce problème, car il force une continuité dans le mouvement tout au long de l'objet. Cependant, dans notre cas, le terme de parcimonie contrebalance le terme TV. Notons que le problème existe également avec le terme linéarisé (2.3), car l'approximation des dérivées partielles avec les différences finies est nulle sur les objets constants.

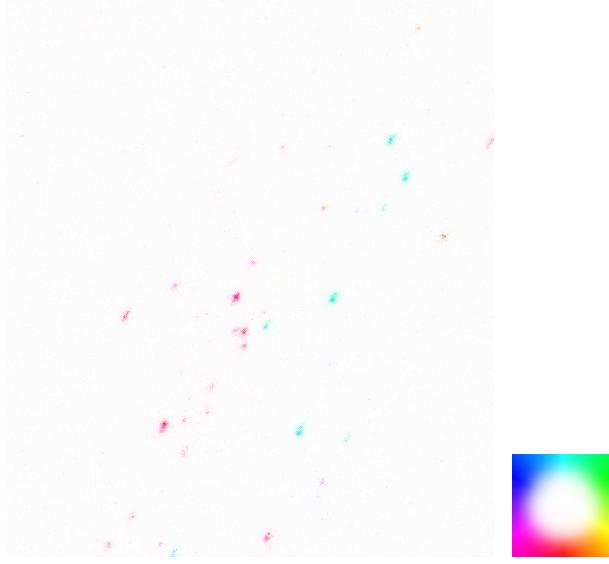


Figure G.1: exemple de résultat avec le terme non linéaire

Etude du terme non linéaire

Comme dit précédemment, pour utiliser le terme non linéaire :

$$I_2(x + w(x)) - I_1(x) = 0$$

il faut utiliser une interpolation de la deuxième image. De plus, il faut alors calculer une dérivée partielle explicite de ce terme. Pour cela, on doit calculer $\nabla_w(I_2(x + w(x)) - I_1(x))$. On obtient alors :

$$\nabla_w I_2(x + w(x)) = \begin{pmatrix} \frac{\partial I_2}{\partial x_1}(x + w(x)) \\ \frac{\partial I_2}{\partial x_2}(x + w(x)) \end{pmatrix}.$$

Il faut calculer à nouveau les différences finies afin de calculer les dérivées partielles de l'image. Par ailleurs, il faut ajouter une interpolation, car $x + w(x)$ n'est toujours pas un pixel entier de l'image. Ces deux éléments font que le calcul du terme non linéaire n'est pas spécialement meilleur. En ce qui concerne les résultats obtenus grâce au terme non linéaire, ils sont plutôt bons, mais ne surpassent pas les résultats avec le terme linéarisé (voir Figure G.1).

Sous-échantillonage et pyramide d'images

Un des problèmes du flot optique est qu'il est difficile à calculer pour de grands déplacements. Ainsi, une autre stratégie pour calculer le flot optique est de construire une pyramide

d'images correspondant à un sous-échantillonnage spatial ("downsampling") des images initiales. Ainsi, la résolution de la première image traitée est beaucoup plus faible que l'image initiale. On commence alors par calculer le flot optique sur l'image la plus réduite. Ensuite, on récupère le flot optique calculé qui va nous servir d'initialisation du flot optique pour la deuxième image. Et ainsi de suite, jusqu'à ce qu'on retourne à l'image initiale. Cette technique de réduire la résolution permet de transformer les grands déplacements de plusieurs pixels en petits déplacements sur les images de moins grandes résolutions.

Lors de mon stage, j'ai essayé d'adapter cette stratégie pour notre méthode parcimonieuse. La méthode en elle même marchait, cependant elle n'est pas adaptée aux images que j'utilisais. En effet, lorsque l'on baisse la résolution, les petits objets des images ont immédiatement disparu sur la deuxième échelle de la pyramide. Ainsi, les premières itérations de la méthode sur les images réduites donnaient de mauvais résultats et très comparables à ceux obtenus avec une initialisation aléatoire du flot optique pour la suite.

H Problème avec le critère DFD

Lors de mon stage, j'ai essayé de poser le problème variationnel du flot optique en utilisant le DFD comme terme d'attache aux données. Voici ce que j'ai écrit :

Disposant de l'image initiale I_0 et du flot optique w , on souhaite reconstruire \hat{I}_1 une approximation de l'image réelle I_1 . Pour cela, on considère un point $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ de I_1 . On pose l'ensemble :

$$A_x(w) = \{a, b : a + w_1(a, b) = x_1, b + w_2(a, b) = x_2\}$$

qui représente l'ensemble des antécédents possibles d'un pixel de l'image I_1 par rapport au flot optique w .

Alors, on peut calculer \hat{I}_1 en x comme :

$$\hat{I}_1(x) = \frac{1}{\#A_x(w)} \sum_{a,b \in A_x(w)} I_0(a, b)$$

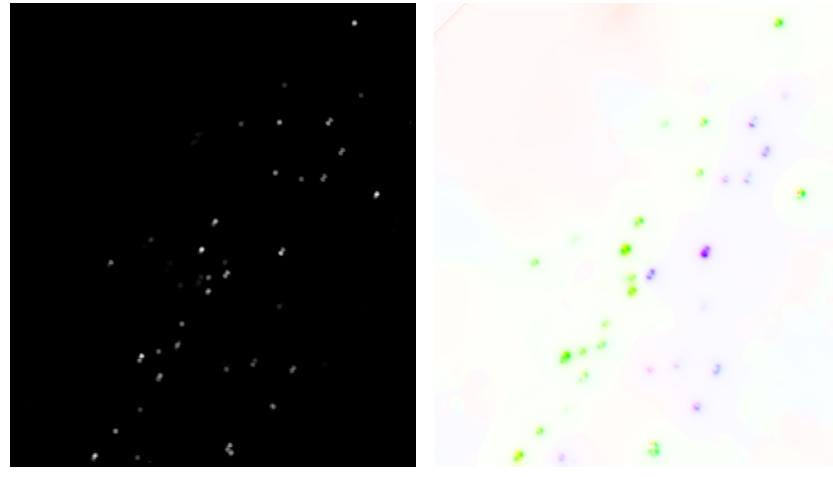
où $\#A_x(w)$ correspond au nombre d'éléments dans $A_x(w)$. Ce terme calcule l'intensité lumineuse au pixel x comme la moyenne de l'intensité lumineuse des pixels arrivant en x .

On a alors le nouveau terme d'attache aux données :

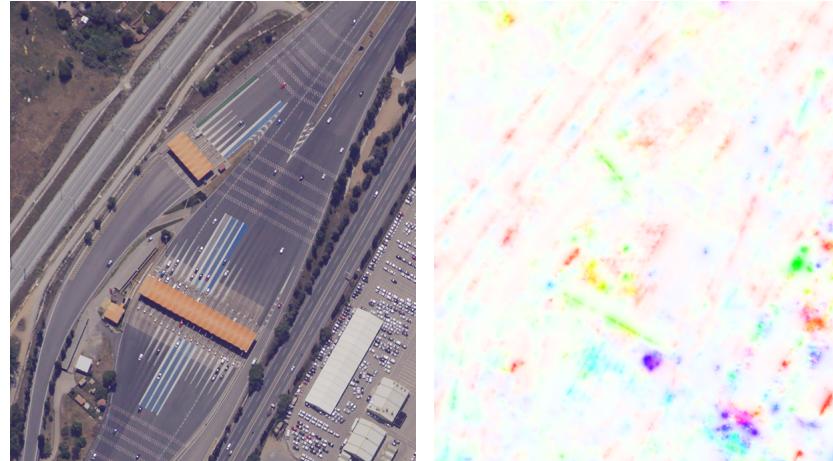
$$\hat{I}_1(x) - I_1(x) = 0.$$

I Exemple des pré-traitements

Vous trouverez ici d'autres exemples de comparaisons entre les images avec et sans prétraitement.



(a) Avec pré-traitement



(b) sans pré-traitement

Figure I.1: Comparaison du calcul du flot optique avec et sans traitement (soustraction de fond et lissage Gaussien) pour un péage routier. Les paramètres sont les suivants : $\lambda = 0.7$, $\alpha = 0.9$, précision à 10^{-10} . A gauche : la première image sur laquelle le flot optique est calculé. A droite : le flot optique calculé.

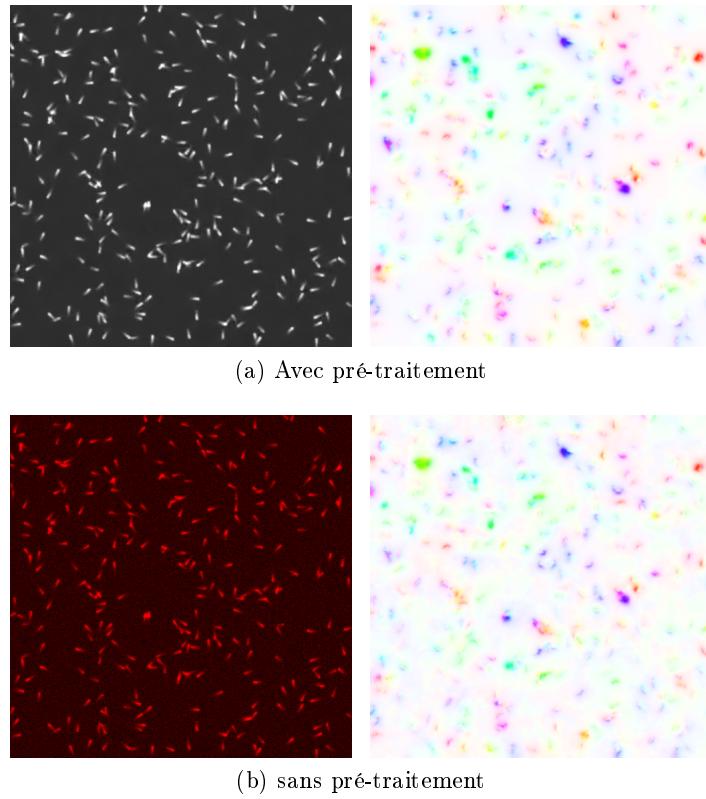


Figure I.2: Comparaison du calcul du flot optique avec et sans traitement (débruitage) pour des extrémités des microtubules en mouvement. Les paramètres sont les suivants : $\lambda = 0.7$, $\alpha = 0.9$, précision à 10^{-10} . A gauche : la première image sur laquelle le flot optique est calculé. A droite : le flot optique calculé.