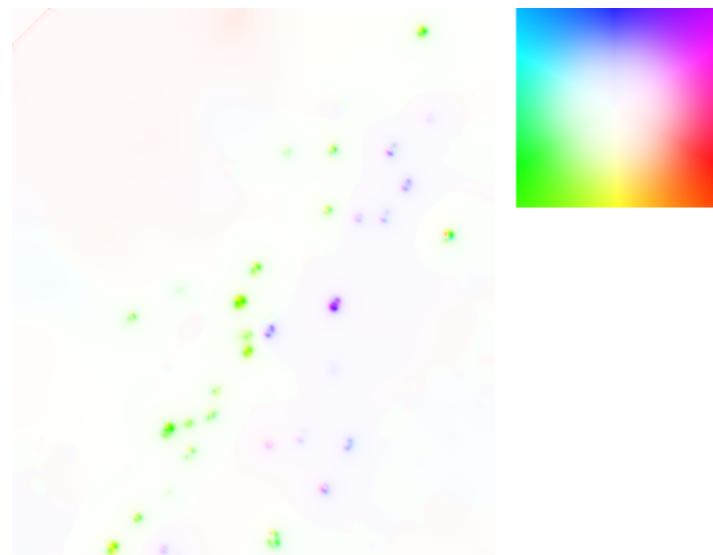


## RAPPORT DE STAGE

### Développement d'une méthode « sparse » pour le calcul du flot optique



**Étudiant :**  
Théotim BARBIER

**Maitre de stage :**  
Charles KERVRANN

CENTRE INRIA 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

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Présentation de l'INRIA . . . . .	6
1.2	Présentation de l'équipe SAIRPICO . . . . .	6
<b>2</b>	<b>Mission</b>	<b>7</b>
2.1	partie théorique . . . . .	7
2.2	Méthode du stage . . . . .	10
2.3	Implémentation . . . . .	13
2.4	Problèmes rencontrés . . . . .	15
2.5	Résultats expérimentaux . . . . .	19
2.5.1	comparaisons normes et initialisation . . . . .	20
2.5.2	Visualisation de l'impact du terme « sparse » . . . . .	22
2.5.3	comparaisons pré traitement images . . . . .	22
2.5.4	comparaison avec IPOL et SMOFlow . . . . .	23
2.5.5	Expérience sur des micro organismes . . . . .	23
2.6	Idées de continuation . . . . .	25
<b>3</b>	<b>Conclusion</b>	<b>26</b>
<b>A</b>	<b>Linéarisation de l'équation 2.2</b>	<b>28</b>
<b>B</b>	<b>problème convexe</b>	<b>28</b>
<b>C</b>	<b>differentiation automatique pytorch</b>	<b>29</b>
<b>D</b>	<b>opencv colorisation</b>	<b>30</b>
<b>E</b>	<b>approximation des normes</b>	<b>30</b>
<b>F</b>	<b>différences finies</b>	<b>30</b>
<b>G</b>	<b>problèmes rencontrés</b>	<b>33</b>
<b>H</b>	<b>problème DFD</b>	<b>35</b>
<b>I</b>	<b>exemple pré traitement</b>	<b>36</b>

## 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, autant par les méthodes traditionnel de vision par ordinateur que par des méthodes plus récentes comme l'utilisation de réseaux de neurones.

Plus particulièrement, mon stage a porté sur le sujet du flot optique. Le flot optique correspond à calculer le mouvement des éléments entre 2 images. Plus précisément, le flot optique permet de repérer les mouvements apparents des objets sur l'image, comme par exemple le déplacement d'une voiture entre 2 images. Dans le cadre de mon stage, le flot optique est principalement utilisé pour repérer le mouvement d'éléments microscopiques comme des cellules ou des particules. Cela permet ainsi d'identifier ces objets, leur mouvement et également de faire du « tracking » de ceux ci.

Cependant, les méthodes classiques de vision par ordinateur ne permettent pas de calculer le flot optique pour de très petits objets, de la taille d'une cellule par exemple. Ainsi, ma mission au sein de l'équipe a consisté en l'étude d'une méthode variationnel afin de calculer le flot optique entre 2 images comportant de petits objets. La méthode sur laquelle j'ai travaillé a pour particularité d'introduire un terme de régularisation « sparse » qui pourrait permettre au flot optique de prendre en compte les petits objets de l'image. Mon stage a commencé par l'étude théorique des méthodes déjà existantes du calcul du flot optique et en particulier des méthodes variationnelles. Ensuite, j'ai abordé l'application informatique du calcul en utilisant la bibliothèque pré-existante de « open CV » en python pour calculer le flot optique. Enfin, j'ai programmé par moi même l'implémentation en python de mon calcul du flot optique, qui consiste à optimiser directement l'équation variationnel à l'aide de la bibliothèque « pytorch ».

Ce rapport présente dans un premier temps le centre INRIA et l'équipe dans laquelle c'est déroulé mon stage. Ensuite, je reviendrais sur ma mission au sein de l'équipe en commençant par une introduction théorique du problème du flot optique. Je développerais par la suite l'implémentation d'une solution abordée durant le stage ainsi que les problèmes rencontrés. Enfin, je finirais par présenter 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 dans la 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 DFD de 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 optiques. 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

FIGURE 1.1 – INRIA de Rennes



et Chemobiologie. Cette équipe est spécialisée dans le traitement d'image pour l'imagerie microscopique. Pour être plus clair, l'imagerie microscopique désigne les images observées à l'aide de microscopes électroniques, dans les domaines de chimie cellulaire ou organique, par exemple. Ainsi, pour apprécier des résultats à partir de ces images, il faut tout d'abord les traiter. L'un des problèmes rencontrés dans ce domaine est que l'image microscopique permet d'observer des éléments microscopiques comme des cellules, des protéines, des tissus dont la taille et la netteté peuvent rendre le traitement de l'image difficile. C'est ici que l'équipe intervient. En effet, l'équipe SAIRPICO est focalisée sur la recherche d'outils et de méthodes pour traiter ces images. L'équipe est principalement composée de chercheurs et de doctorants.

## 2 Mission

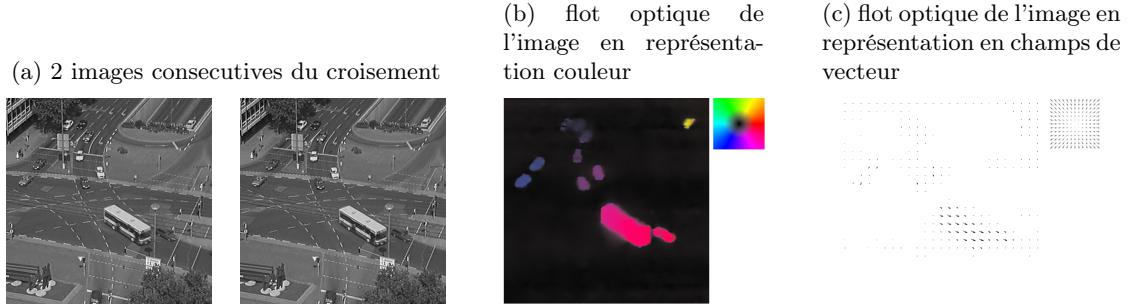
Dans cette partie, je vais expliquer en quoi consistait ma mission lors de mon stage. Tout d'abord, je vais présenter l'aspect théorique du problème du flot optique.

### 2.1 partie théorique

#### Flot Optique

**Definition :** Le flot optique (ou flux optique) [15] entre 2 images correspond au calcul du mouvement visible entre ces images. Le mouvement sur une image peut provenir de différentes sources : d'éléments se déplaçant à l'intérieur du cadre, d'un mouvement global de l'espace par rapport à l'observateur ou d'un mouvement de l'observateur lui-même. Ainsi, ce qu'on appelle le flot optique est la représentation mathématique de ce mouvement. Il

FIGURE 2.1 – exemple de flot optique pour la vue de véhicules en mouvement sur un croisement routier



est traditionnellement représenté par un champ de vecteurs indiquant le mouvement d'un nombre fini 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 déplacement (voir *figure 2.1*).

Le flot optique a énormément d'applications possibles dans le traitement d'image et la vision par ordinateur. Par exemple, dans la robotique, le flot optique peut servir dans la détection d'objets environnants et le tracking de ces objets pour des problèmes de navigation et d'évitement des objets. Il peut aussi servir dans la production d'effets audio visuels pour des films [13].

Dans notre cas, nous utilisons le flot optique sur des images microscopiques. Ces images comprennent des molécules, des cellules ou encore des organismes microscopiques. Ainsi, le flot optique permet de retrouver le mouvement de ces particules pour par exemple effectuer du tracking de celles-ci afin de les étudier. En effet, le mouvement d'un organisme peut en dire long sur son comportement et son utilité. Notons que le flot optique peut aussi bien être calculé sur des images en 2 dimensions qu'en 3 dimensions.

### calculs théoriques

Pour estimer le flot optique, nous avons besoin d'une hypothèse. Pour cela, commençons par définir une 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  $T$  représente l'aspect temporel de la séquence. Cette fonction représente l'intensité lumineuse d'un pixel  $(x, y)$  de l'image à un instant  $t$ . Notons également notre flot optique comme une fonction :

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

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

L'hypothèse principale utilisée pour le flot optique est l'invariance d'intensité lumineuse (brightness constancy) [3] défini par le fait qu'un objet en mouvement sur une image est censé conserver la même intensité lumineuse tout au long du déplacement. 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 traduit l'hypothèse que la variation d'intensité lumineuse d'un objet en déplacement est nulle. Cependant, expérimentalement, on ne dispose pas d'une fonction continue, et l'on peut donc utiliser l'équation discrétisée associée :

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

où  $w$  dénote donc 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 chercher à être résolue. Néanmoins, l'équation (2.2) pose certains problèmes en termes d'optimisation du fait de la non linéarité (et de la non continuité) expérimentale de la fonction  $I$ . On peut alors penser à linéariser l'équation (2.1) (précisions en annexe A) de la sorte :

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

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

Le terme engendré par ces équations est appelé terme d'attache aux données, noté  $\rho_{data}$ . On peut remarquer que ces équations ne possèdent pas une unique solution  $w$  du fait que notre vecteur possède 2 dimensions ( $w \in \mathbb{R}^2$ ). Le problème du flot optique est donc un problème mal posé. Ainsi, pour que le problème soit bien posé, il nous faut rajouter une deuxième contrainte de régularisation qui peut être locale ou globale. On note ce terme de régularisation  $\rho_{reg}$ . Le terme de régularisation le plus utilisé est le terme de variation totale (TV) :

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

Ainsi, en minimisant ce terme, on constraint le champs de vecteur à être invariant à un pixel donné  $x$ .

Finalement, pour résoudre le problème du flot optique, la principale méthode 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  $\lambda$  est un paramètre permettant de réguler l'impact de chaque terme. On peut également réécrire l'équation (2.5) de cette façon :

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

avec  $\lambda \in [0; 1]$

### état de l'art des méthodes utilisées

Il existe de nombreuses méthodes pour calculer le flot optique de 2 images successives [3]. La pluparts de ces méthodes utilisent l'équation (2.5) comme base.

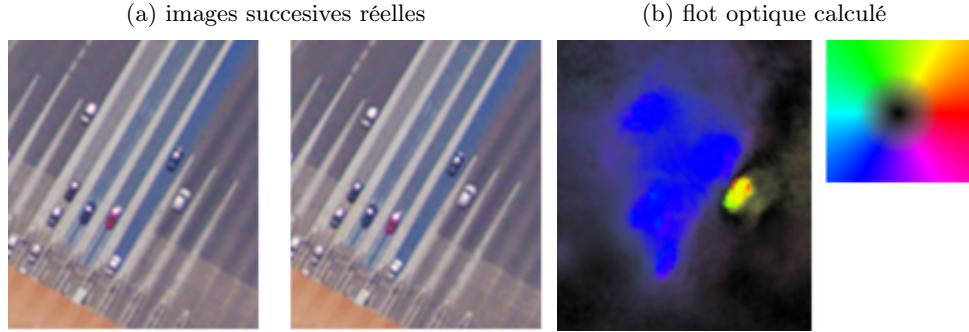
Dans un premier temps, la méthode plus traditionnelle consiste à considérer simplement l'équation (2.5) comme un problème variationnel. On peut alors choisir d'utiliser les équations d'Euler-Lagrange pour l'optimisation numérique, comme dans [1]. D'autres ont également proposé une résolution multiobjective 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'attaches aux données et de régularisations peuvent être utilisés. Tout d'abord, certains utilisent soit le terme non-linéaire (2.2) qui consiste en une approche plus proche de la réalité, et d'autres utilisent le terme linéarisé (2.3) plus facile à optimiser. Pour le terme de régularisation, la plupart utilise le terme TV[11] associé à la norme l1 ou l2. Enfin, on peut encore utiliser d'autres méthodes comportant un pré-traitement des images, comme un lissage local gaussien [4].

Plus récemment, l'utilisation de réseaux de neurones et du deep learning a été explorée pour calculer le flot optique. Pour cela, ils utilisent une fonction de coût similaire à l'équation (2.5) pour entraîner le réseau. La plupart des réseaux sont des modèles supervisés. On a par exemple 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 de machine learning utilisé est RAFT [14] qui se distingue par sa capacité à produire des estimations denses et détaillées du flot optique avec une grande précision, même dans des scènes complexes. Enfin, il existe SMOFlow [6] qui est une approche en réseau de neurones pour de petits objets.

## 2.2 Méthode du stage

Le principal problème des méthodes déjà existantes dans notre cas des images microscopiques est que le déplacement des petits objets est très mal mesuré. Or, dans nos images, les particules en mouvement peuvent être très petites, ce qui ne permet pas de calculer le flot optique par des méthodes traditionnelles. Ainsi, le but de ce stage est de trouver une méthode afin de prendre en compte les petits éléments d'images.

FIGURE 2.2 – Calcul du flot optique à l'aide d'une méthode TV- $l_1$  [11]. On remarque l'effet de bavure du flot optique calculé figure (b) par rapport à l'image réel (a).



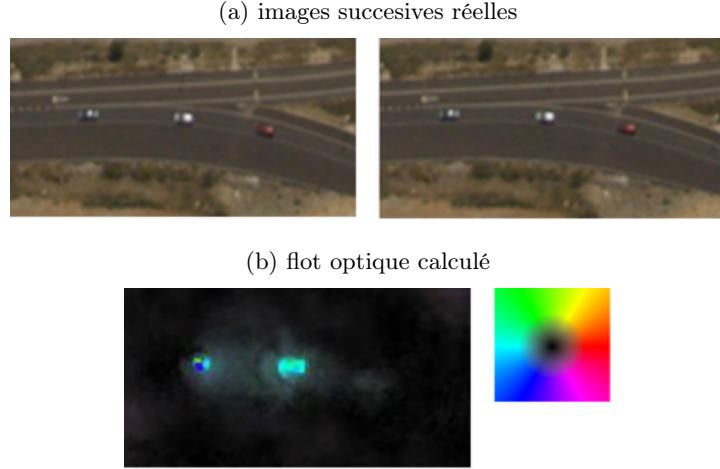
Le problème avec les méthodes déjà existantes se situe au niveau du terme de régularisation. En effet, la régularisation TV utilisée classiquement impose une certaine invariance et continuité du flot optique. Cette contrainte est utile pour imposer un mouvement continu sur tout l'ensemble d'un objet en mouvement. Cependant, cette contrainte est un problème pour les petits objets. En effet, la continuité du flot optique peut créer un effet de « bavure » (*figure 2.2*) autour des objets. Ainsi, les petits objets peuvent être 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 pas très bien différencié par les méthodes et peut conduire à un même mouvement (*figure 2.2*) sur l'ensemble des 2 objets qui est différent de la réalité. Finalement, le dernier problème est le fait que si le fond de l'image est fixe, le flot optique va être principalement nul, et par effet du régulariseur, les petits objets vont être confondus avec le fond et le mouvement va être effacé (*figure 2.3*) (sorte d'effet de propagation du mouvement de l'arrière-plan).

Pour contrer ces problèmes, lors de mon stage, j'ai étudié les méthodes traditionnelles d'approche variationnelle et, en particulier, j'ai exploré l'idée d'ajouter un terme « sparse » dans ces méthodes. Un terme « sparse » (creux en français) est un terme qui impose au flot optique de tendre 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 ne bouge pas. Ainsi, tout le fond de l'image possède un flot optique nul. Ce terme « sparse » aide donc le calcul du flot optique à respecter l'immobilité du fond. Une autre idée dans l'utilisation de ce régulariseur est qu'il permettrait de calculer un flot optique plus discontinu, ce qui empêcherait les effets de « bavures ». Cela peut paraître contradictoire

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 réel (a).



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.

Plus théoriquement, j'ai donc testé d'ajouter un terme « sparse » au terme préexistant TV. Une méthode similaire a déjà été explorée dans un autre problème variationnel avec [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. Le paramètre  $\alpha \in [0; 1]$  représente le poids entre les 2 régularisateurs. Ainsi, théoriquement, un poids  $\alpha$  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  $\phi$  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, habituellement, on utilise les équations d'Euler Lagrange [1]. Dans notre cas, nous allons essayer de minimiser directement l'énergie  $E(w)$  par un algorithme de descente de gradient stochastique, car notre problème est convexe (voir annexe).

### 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 dans du langage C et ensuite en Python. Cette bibliothèque est utile pour avoir une base, car elle propose différentes méthodes pour calculer le flot optique. Lors de mon stage, elle m'a donc servi de base pour commencer l'implémentation de ma méthode et, plus précisément, pour la visualisation en couleur du flot optique. OpenCV est également utile pour charger et pré-traiter les images s'il y a besoin.

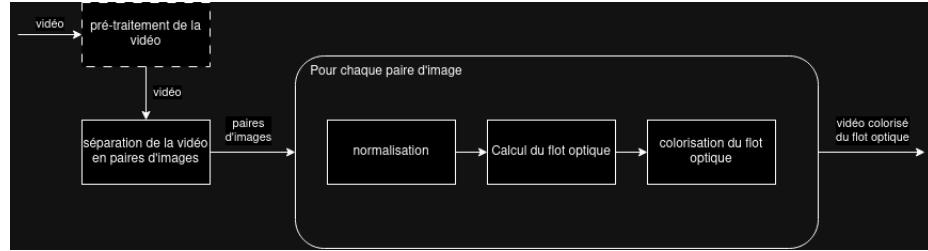
**pytorch** Pytorch est une bibliothèque Python qui permet de construire et d'entraîner des réseaux de neurones. Cependant, ce qui nous intéresse dans Pytorch n'est pas la possibilité de pouvoir facilement créer un réseau de neurones. Ce qui nous intéresse est l'implémentation des méthodes d'optimisation utiles pour entraîner ces réseaux [8]. En effet, l'entraînement d'un réseau consiste en la minimisation d'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 [ici](#). 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, qui est nécessaire pour notre optimisation.

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.



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 l'optimisation de notre énergie (2.7). Dans un premier temps, nous avons besoin de méthodes pour implémenter le calcul de notre énergie, qui représente la fonction coût à minimiser (*algorithme 1*). Pour cela, j'ai créé 2 méthodes : une pour le calcul du terme d'attache aux données et une pour le calcul du terme de régularisation. Pour les deux termes, j'ai utilisé les fonctions préexistantes de Pytorch pour les opérations de base, car cela permet d'utiliser facilement la différentiation automatique par la suite.

Malheureusement, l'approximation du flot optique  $w(\cdot)$  comme une fonction continue est impossible informatiquement. Pour pallier cela, 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 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 la hauteur de l'image et la dernière dimension représente les coordonnées cartésiennes 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. Vous trouverez en Annexe les choix effectués pour les différences finies. Vous trouverez également en annexe l'approximation utilisée pour les normes.

Après avoir calculé la fonction de coût, il reste à faire avancer l'optimisation Pytorch en utilisant la fonction « `optimizer.step()` » de Pytorch. Notez 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

---

**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

```

---

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.

**paramètres** Dans mon implémentation, les principaux paramètres modifiables sont :

- le paramètre de régularisation  $\lambda$  qui permet de jouer sur l'équilibre entre le terme d'attache aux données et de régularisation.
- le paramètre  $\alpha$  de « sparsity » qui permet d'équilibrer entre le terme TV et le terme « sparse »
- la précision que l'on cherche dans l'optimisation

On peut également influer sur le learning rate de l'optimisation qui permet de modifier la vitesse de convergence de la solution. Cependant, il faut faire attention en modifiant ce paramètre, car il peut également amener la solution à ne pas converger.

## 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 importants. Un résumé des autres problèmes est présenté en Annexe.

## initialisation flot optique

L'un des principaux problèmes rencontrés est la question de comment 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. De plus, l'initialisation est importante, car elle peut changer la convergence de la minimisation. 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  $\beta$ , avec  $\beta > 0$
- intialisation de  $w$  de façon aléatoire entre  $-\beta$  et  $\beta$

En testant ces trois initialisations avec différentes valeurs pour  $\beta$ , 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 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 dont je parle en section 2.4.6.

## normes

Pour le calcul des termes de régularisation et d'attache aux données, il faut choisir les normes à utiliser. Les plus communes sont les normes  $l_1$  et  $l_2$ . Ces 2 normes sont implémentables, cependant elles ont certains avantages et inconvénients.

Tout d'abord, la norme  $l_2$  est facilement différentiable (même en 0) si elle est élevée au carré. Ceci est un atout majeur dans notre optimisation, car notre champ de vecteurs est principalement nul et donc on doit 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 a surinterpréter les petites et grandes valeurs. Par exemple, si on a une valeur  $x$  de l'ordre de  $10^{-3}$ , alors on a  $\|x\|_2^2$  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 d'en calculer une approximation. De plus, la bibliothèque Pytorch arrive à 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. Cela peut sembler beaucoup d'approximation, cependant, 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 forcément 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 » 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, ainsi on ne gagne pas forcément en précision. Vous trouverez plus d'explications sur ce terme et l'implémentation en annexe.

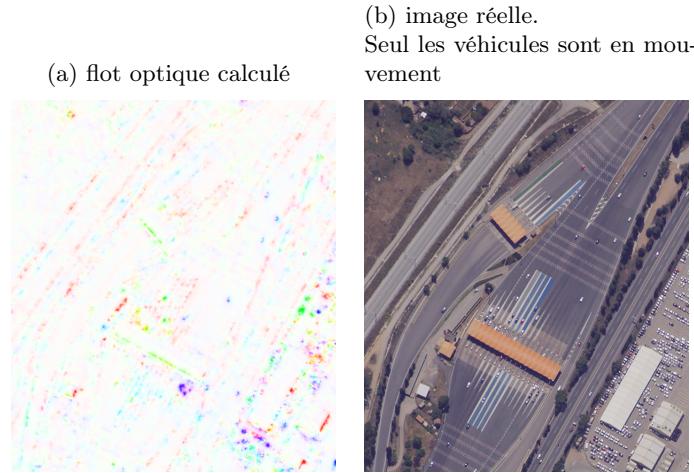
## différences finies

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.

## intensité lumineuse/préprocessing

Un autre problème important du flot optique dans notre cas est le problème de l'intensité lumineuse. Premièrement, l'intensité lumineuse dépend très fortement des conditions dans lesquelles les images sont prises. En effet, sur une même, on peut avoir une luminosité totalement différente d'une image à l'autre. Ce phénomène pose problème, car notre algorithme se base sur l'intensité lumineuse de 2 images successives pour le calcul du flot optique.

FIGURE 2.5 – Exemple du problème du fond bruité dans le calcul du flot optique



Ainsi, il est important de faire une normalisation des 2 images afin que les valeurs calculées correspondent à la réalité.

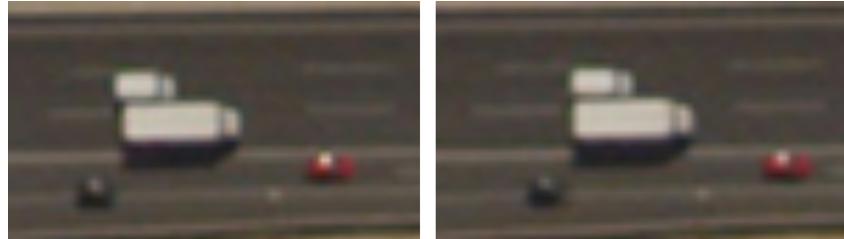
Secondement, les images récupérées sont très souvent bruitées. En particulier, sur nos images où le fond est immobile, celui-ci est très sensible au bruit. Cela pose problème, car ce bruit peut conduire à un calcul de mouvement pour notre algorithme (*figure 2.5*). Il est alors important de procéder 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. En plus, de cela, il arrive parfois que le flot optique calculé sur le contour de l'objet parte vers le centre de l'objet et non vers le mouvement réel de l'objet. Une explication plus théorique est donnée en Annexe.

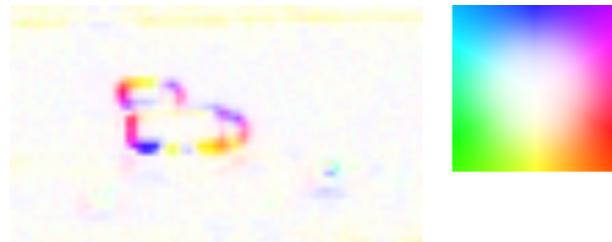
Pour résoudre ce problème, une idée est de procéder à un pré-traitement qui permet d'enlever cette constance de l'intensité lumineuse. Pour cela, on peut appliquer tout d'abord une méthode de soustraction. Cette méthode consiste à appliquer un filtre médian sur l'image qui permet « d'effacer » les petits objets en mouvements sur l'image. Ensuite, on peut soustraire l'image de base et l'image filtrée, ce qui fait ressortir les petits objets effacés par le filtre. Cette méthode permet d'effacer le fond immobile de l'image, ce qui résout un de nos problèmes. Ensuite, il suffit de passer l'image en nuances de gris et d'appliquer un filtre gaussien. Le filtre gaussien permet de rendre les petits objets comme des « gaussiennes » sur l'image (*figure 2.7*) et réduit la constance de l'intensité lumineuse, ce qui résout ce dernier

FIGURE 2.6 – Exemple du problème du centre des objets

(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



problème.

### normalisation du flot optique

Lorsque l'on utilise le terme d'attache aux données linéarisé 2.3, le calcul du flot optique dépend seulement des dérivées partielles des images. Plus exactement, ce calcul effectué avec les différences finies dépend de l'intensité lumineuse des pixels des images. Ainsi, le flot optique calculé 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 compliqué et n'a pas été résolu au cours de mon stage.

## 2.5 Résultats expérimentaux

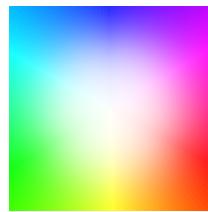
Dans cette partie, je vais présenter les résultats obtenus avec notre méthode « sparse ». Pour toutes ces expérimentations, la lut de couleur associée au flot optique colorisé calculé par

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



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 « sparse »



**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 idéales expérimentalement, car le mouvement réel des objets sur celles-ci peut facilement être identifié. Ainsi, on peut aisément comparer le flot optique calculé par la méthode avec le mouvement réel des éléments. Cependant, cela reste une validation qualitative et assez subjective. Il faudrait implémenter une validation automatique comme le « Displaced Frame Differences » (DFD).

### 2.5.1 comparaisons normes et initialisation

Dans cette partie, nous allons regarder quelques comparaisons entre les différentes normes et initialisations possibles (voir sections 2.4.1 et 2.4.2). Ici, j'ai utilisé trois méthodes différentes : soit toutes les normes en norme  $l_1$ , soit toutes en norme  $l_2$ , soit un mix où le

TABLE 1 – tableau de comparaison des différentes normes et initialisation. Le calcul a été effectué sur les images de la *figure 2.7*. Les paramètres sont :  $\lambda = 0.7$ ,  $\alpha = 0.9$ , précision à  $10^{-10}$ .

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

terme de régularisation est en norme  $l_1$  et le terme d'attache aux données est en norme  $l_2$ . Pour ce qui est de l'initialisation, j'en ai également testé 3 différentes : initialisation à 0, de manière aléatoire entre -0,1 et 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, ce qui n'est pas le cas si l'on utilise 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 effacés. Pour ce qui est de l'initialisation, 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. En plus de cela, 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.

TABLE 2 – Comparaison des temps de calcul (en secondes) des différentes méthodes sur les même paramètres que le *tableau 1*

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

FIGURE 2.9 – Comparaison des flot optiques calculé en fonction du paramètre  $\alpha$  de « sparsity ». On utilise les images de la *figure 2.7* afin de calculé le flot optique. Les autres paramètres sont fixes :  $\lambda = 0.7$  et la précision est de  $10^{-10}$ . La lut de couleur est celle présenté en *figure 2.8*

(a)  $\alpha = 0.7$     (b)  $\alpha = 0.8$     (c)  $\alpha = 0.9$     (d)  $\alpha = 0.925$     (e)  $\alpha = 0.95$



### 2.5.2 Visualisation de l’impact du terme « sparse »

Dans un premier temps, j’ai effectué quelques expériences pour visualiser l’impact qu’apporte le poids du terme sparse sur le calcul du flot optique. Pour cela, comme illustré sur la *figure 2.9*, j’ai effectué le calcul du flot optique d’une image satellite pour différentes valeurs du poids de la régularisation « sparse ». On remarque alors que lorsque le terme de régularisation est faible, on retrouve l’effet de « bavure » provoqué par le terme TV. À l’inverse, lorsque le terme « sparse » est important, le mouvement des objets est de moins en moins visible jusqu’à disparaître complètement. Ce phénomène montre ainsi l’effet attendu par l’utilisation du terme « sparse » qui permet de réduire l’effet de « bavure ». On peut alors essayer de trouver des paramètres « parfaits » qui permettent la meilleure approximation du flot optique.

### 2.5.3 comparaisons pré traitement 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 en 2.4.6 . On observe les résultats dans la figure 2.10 sur l’image satellite d’un péage. D’autres images sont présentées en Annexe. On remarque que le flot optique calculé pour les images sans prétraitement

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.

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



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



n’arrive pas à trouver 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 de prétraitement.

#### 2.5.4 comparaison avec IPOL et SMOFlow

Ensuite, j’ai comparé la méthode « sparse » que j’utilise 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 dans 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. Les méthodes « sparse » et SMOFlow ont des performances assez similaires.

#### 2.5.5 Expérience sur des micro organismes

Pour cette dernière expérience, j’ai testé notre algorithme sur une vidéo de microtubules en mouvement. Contrairement aux véhicules, ces petits organismes vont dans tous les sens possibles 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

FIGURE 2.11 – Comparaison des 3 méthodes sur les images de péage (*figure 2.7*)

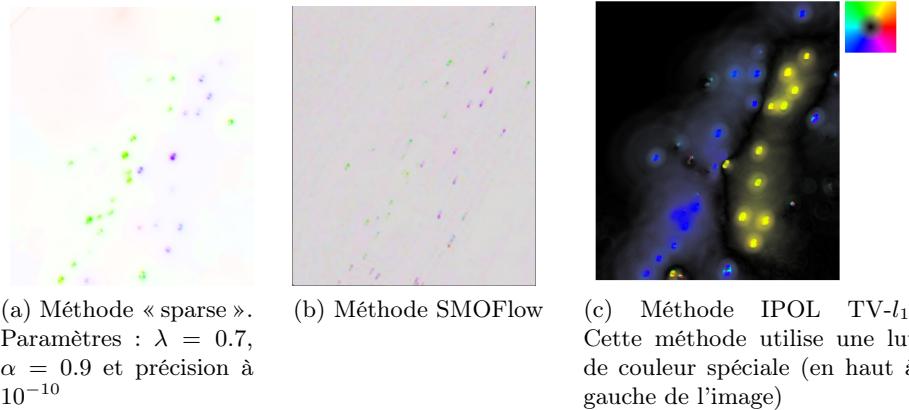
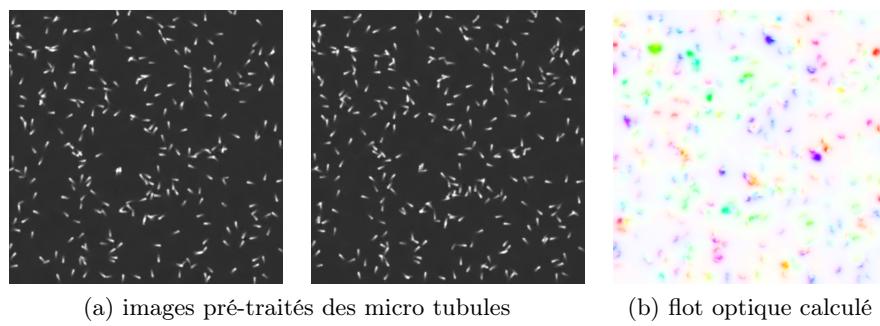


FIGURE 2.12 – calcul du flot optique sur des images de micro tubules. Les paramètres sont :  $\lambda = 0.7$ ,  $\alpha = 0.9$ , précision à  $10^{-10}$ .



que le flot optique arrive à repérer la plupart des microtubules. Cependant, nous pouvons également observer des effets de « bavures » qui font que plusieurs microtubules sont réunis en un seul mouvement. De plus, le flot optique calculé n'est pas toujours dans la bonne direction.

## 2.6 Idées de continuation

### complément possible sur les résultats

Plusieurs axes pourraient être observés pour compléter les résultats. Tout d'abord, nous pourrions utiliser une validation DFD mentionné précédemment 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.

### 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) de 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 à part entière.

### choix paramètres

Une des difficultés de nos méthodes réside dans le choix des paramètres de régularisation et de « sparsity ». En effet, le flot optique calculé dépend fortement de ces paramètres et, pour avoir 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]. Cependant, ce principe marche habituellement pour des problèmes bien 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 « sparse » 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 traitements 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 sparse sur d'autres méthodes d'optimisations plus robustes ou bien de trouver une méthode de recherche des paramètres optimaux.

## Références

- [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, Noémi 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’encyclopédie 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 :

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

Alors, 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 differentiation 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. Ensuite, 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 opencv 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 nos angles et rayons entre  $0^\circ$  et  $180^\circ$  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  $\epsilon$  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

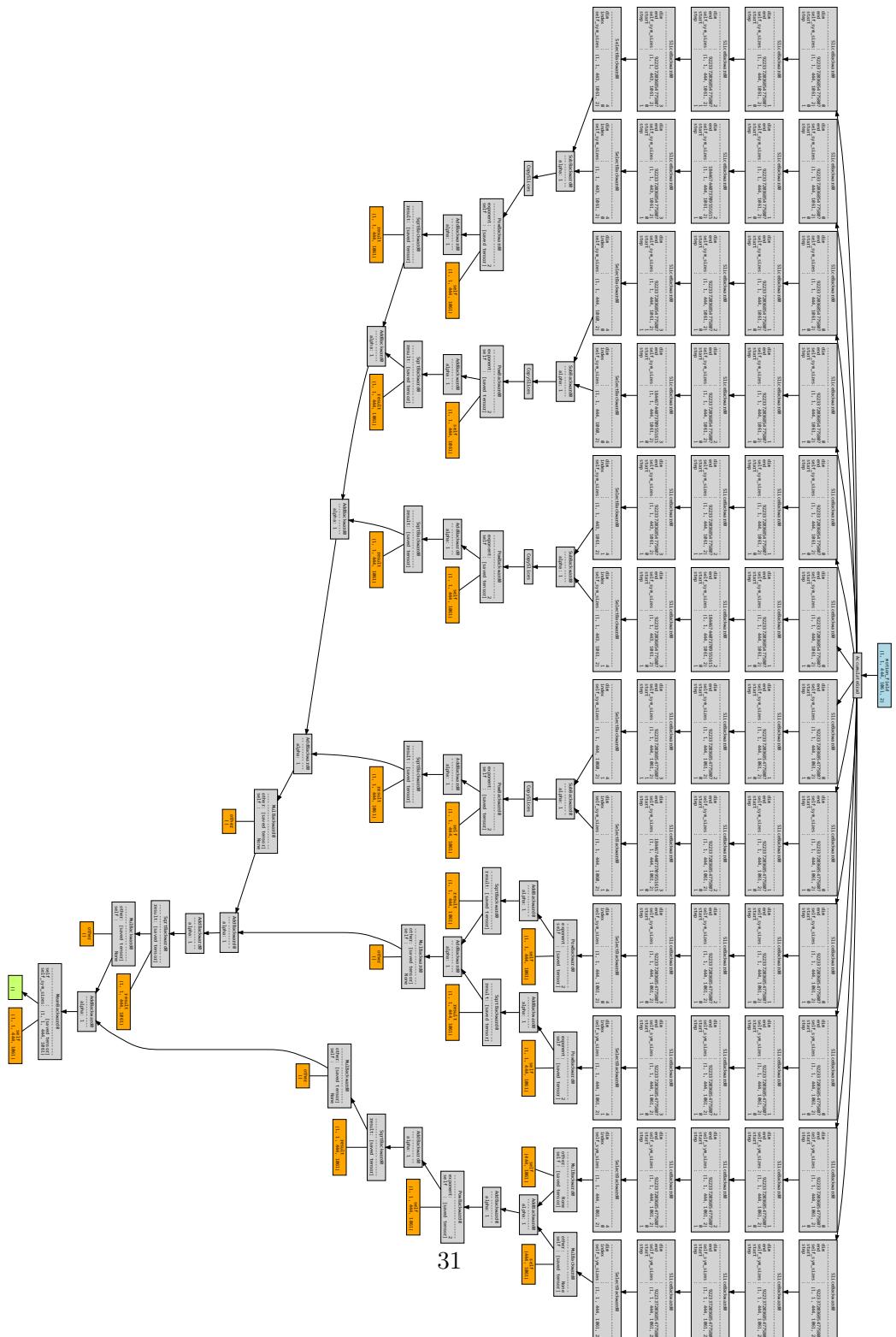


FIGURE D.1 – lute de couleur. On observe que la zone blache au centre est assez grande pour éviter de montrer les très petits déplacements causés par le bruit.

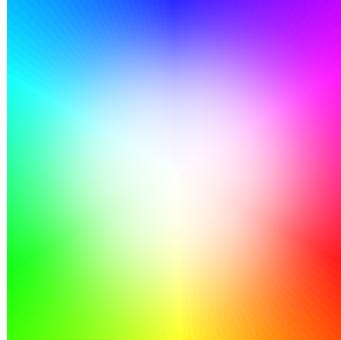
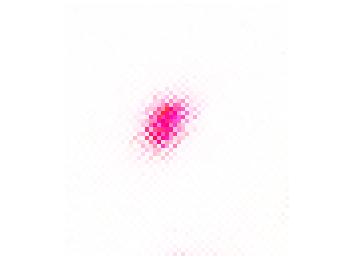
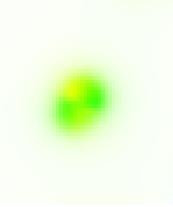


FIGURE F.1 – exemple de flot optique calculé avec les dérivées centrées (la couleur est différente de la lute de couleur habituelle mais ce n'est pas important)



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'attaché aux données  $\rho_{data}$ , 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 2 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 pour nos calculs afin d'éviter le phénomène de quadrillage. Le seul problème des dérivées décentrées réside dans le choix du sens dans lequel on calcule. Ce sens, 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

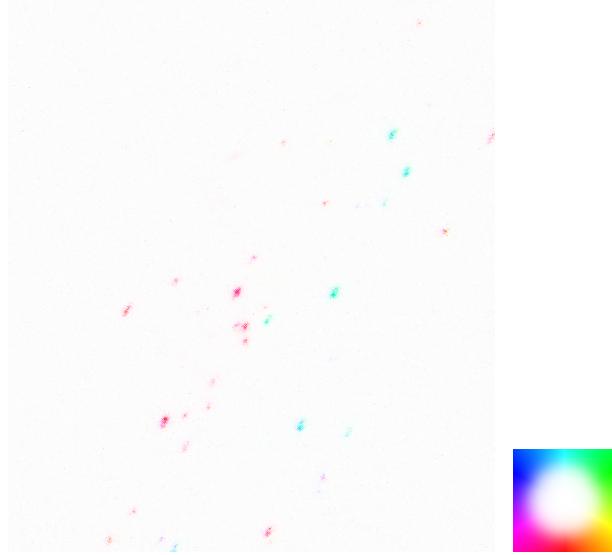
### algorithme d'optimisation

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 constance 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, respectent 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 « sparse ». 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 « sparse » contrebalance le terme TV, donc il nous faut autre chose. 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 ces objets constants.

FIGURE G.1 – exemple de résultat avec le terme non linéaire



### utilisation du terme non linéaire

Comme dit dans le rapport, pour utiliser le terme non linéaire :

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

il faut utiliser une interpolationpolation de notre 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 alors calculer à nouveau les différences finies afin de calculer les dérivées partielles de l'image. En plus de cela, 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 dépassent pas les résultats avec le terme linéarisé (voir figure).

### downsampling

Un des problèmes du flot optique est qu'il est difficile à calculer pour de grands déplacements. Ainsi, une autre méthode pour calculer le flot optique est de partir d'une image et

d'utiliser une pyramide d'image correspondant à un sous-échantillonnage (downsampling) des images de base. Ainsi, la résolution de la première image traitée est beaucoup plus faible que l'image de base. 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 de base. 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'implémenter cette méthode pour notre méthode « sparse ». 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 sont complètement effacés et donc cela empêche de calculer le flot optique. Ainsi, les premières itérations de la méthode sur les images réduites donnaient de mauvais résultats et revenaient à utiliser une initialisation aléatoire du flot optique pour la suite.

## H problème 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 pré traitement

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

FIGURE I.1 – Comparaison du calcul du flot optique avec et sans pré-traitement pour un péage routier. 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.

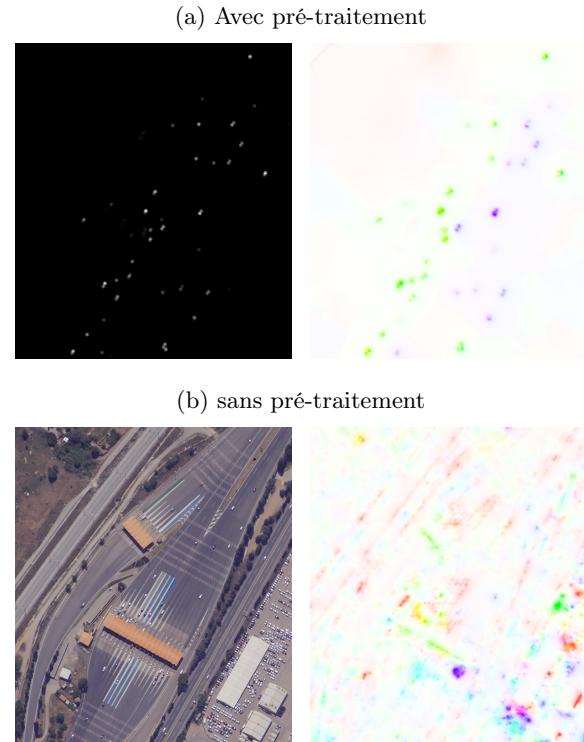


FIGURE I.2 – Comparaison du calcul du flot optique avec et sans pré-traitement pour des micro-tubules en mouvement. Les paramètres sont :  $\lambda = 0.7$ ,  $\alpha = 0.9$ , précision à  $10^{-10}$ . A gauche : la première image sur laquelle le flot optique est calculé. A droite : le flot optique calculé.

