

Projet Recherche

-

Apprentissage non-supervisé pour l'estimation de champs de déformation à la surface de matériaux soumis à des contraintes mécaniques

Oscar Egretteau

Promotion 2024



Année 2025-2026

Sous la supervision de Frédéric Sur

Table des matières

1 Motivations	2
2 Flot Optique	2
2.1 Introduction	2
2.1.1 Hypothèse pour le Flot Optique	2
2.1.2 Établissement de l'équation du Flot Optique	2
2.1.3 Problème de l'ouverture	2
2.2 Estimation du flot optique	2
2.2.1 Méthode de Horn-Schunck	2
2.2.2 Méthode de Lucas-Kanade	3
2.3 Méthodes Numériques	3
2.3.1 OpenCV	3
2.3.2 Méthode TV-L1	3
2.3.3 Implémentation et premiers résultats	4
3 Estimation de Flot Optique par réseaux de neurones	7
3.1 Les Réseaux de Neurones	7
3.1.1 Principe	7
3.1.2 Les ConvNet	8
3.1.3 Les U-Net	8
3.2 Estimation de Flot Optique par apprentissage supervisé	8
3.2.1 L'architecture FlowNet	8
3.2.2 Modèle simplifié	9
3.2.3 Limites de l'apprentissage supervisé	9
3.2.4 Résultats	10
3.3 Estimation de Flot Optique par apprentissage non-supervisé	11
3.3.1 Motivations	11
3.3.2 Approche technique	11
4 Annexes	11
Références	11

1 Motivations

écrire des choses

2 Flot Optique

2.1 Introduction

2.1.1 Hypothèse pour le Flot Optique

L'hypothèse la plus importante qui sert dans la détermination du flot optique est celle de l'illumination constante. Elle suppose que le triplet $(X_1(t), X_2(t), X_3(t))$, représenté par le couple $(x_1(t), x_2(t))$ dans le plan de la caméra a une luminosité qui ne dépend pas du temps : $I(t, x_1(t), x_2(t)) = I_0, \forall t \geq 0$.

2.1.2 Établissement de l'équation du Flot Optique

On suppose donc ici que l'illumination d'un point au temps t reste constante. On a alors :

$$\forall t \geq 0, I(t, x_1(t), x_2(t)) = I_0$$

On dérive donc de chaque côté par rapport à t :

$$\forall t \geq 0, \frac{d}{dt} I(t, x_1(t), x_2(t)) = 0$$

D'après la formule de dérivation des fonctions composées, on obtient :

$$\begin{aligned} \frac{d}{dt} (I(t, x_1(t), x_2(t))) &= \frac{\partial}{\partial t} I(t, x_1(t), x_2(t)) + \frac{dx_1}{dt} \frac{\partial}{\partial x_1} I(t, x_1(t), x_2(t)) + \frac{dx_2}{dt} \frac{\partial}{\partial x_2} I(t, x_1(t), x_2(t)) \\ &= \frac{\partial}{\partial t} I(t, x_1(t), x_2(t)) + \nabla I \cdot v \end{aligned}$$

$$\text{avec } v := \left(\frac{dx_2}{dt}, \frac{dx_1}{dt} \right)$$

Finalement, on obtient donc l'équation du flot optique, pour laquelle on cherche v :

$$\frac{\partial I}{\partial t} + \nabla I \cdot v = 0$$

2.1.3 Problème de l'ouverture

L'équation du flot optique,

$$\frac{\partial I}{\partial t} + \nabla I \cdot v = 0$$

est une équation à deux inconnues, $\frac{dx_1}{dt}$ et $\frac{dx_2}{dt}$, et dont les solutions reposent donc sur une droite. Le problème admet donc une infinité de solutions : il s'agit d'un problème inverse mal posé.

2.2 Estimation du flot optique

2.2.1 Méthode de Horn-Schunck

[4] La première méthode qui permet l'estimation du flot optique consiste à ajouter une contrainte supplémentaire, celle visant à avoir un champ assez lisse. En notant $u := \frac{dx_1}{dt}$ et $v := \frac{dx_2}{dt}$, on a que $\nabla u = \left(\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2} \right)$. Le premier terme représente la variation de u dans la direction x et le deuxième la variation dans la direction y . Horn et Schunck introduisent donc une fonctionnelle à minimiser :

$$\iint \left(\frac{\partial I}{\partial t} + \nabla I \cdot (u, v) \right)^2 dx dy$$

Qui revient à trouver un couple (u, v) qui résoud l'équation du flot optique, car le minimum de celle-ci est 0, atteint pour les couples qui vérifient l'équation.

On ajoute de plus une fonctionnelle traduisant la contrainte de régularité, que l'on va également chercher à minimiser.

$$\lambda \iint \|\Delta(u, v)\|^2 dx dy$$

où λ est un paramètre strictement positif

Ainsi, la méthode Horn-Schunck consiste en la minimisation de la fonctionnelle qui correspond à la somme des deux précédentes, c'est-à-dire

$$E(u, v) := \iint \left(\frac{\partial I}{\partial t} + \nabla I \cdot (u, v) \right)^2 dx dy + \lambda \iint \|\Delta(u, v)\|^2 dx dy$$

E est positive en tant que somme de carrés et par positivité de l'intégrale.

Un des désavantages de cette méthode est qu'elle ne peut être utilisée que pour mesurer des déplacements importants, car la régularité imposée implique que l'on ne peut pas mesurer les petites variations locales car se font régulariser.

2.2.2 Méthode de Lucas-Kanade

[5] Deuxièmement, on peut estimer le flot optique via une deuxième méthode, celle de Lucas-Kanade. Elle suppose quant à elle que le déplacement d'un point de l'image entre deux instants consécutifs est petit et approximativement constant dans un voisinage du point p . On note (u, v) le vecteur des vitesses locales autour de ce point p . En supposant qu'il y a n pixels dans cette fenêtre et que l'on les nomme (q_1, \dots, q_n) , u et v doivent vérifier le système suivant :

$$\begin{cases} \frac{\partial I}{\partial x}(q_1)u + \frac{\partial I}{\partial y}(q_1)v = -\frac{\partial I}{\partial t}(q_1) \\ \frac{\partial I}{\partial x}(q_2)u + \frac{\partial I}{\partial y}(q_2)v = -\frac{\partial I}{\partial t}(q_2) \\ \vdots \\ \frac{\partial I}{\partial x}(q_n)u + \frac{\partial I}{\partial y}(q_n)v = -\frac{\partial I}{\partial t}(q_n) \end{cases}$$

En ayant injecté dans l'équation du flot optique. On peut réécrire ce système sous forme matricielle :

$$\begin{pmatrix} \frac{\partial I}{\partial x}(q_1) & \frac{\partial I}{\partial y}(q_1) \\ \frac{\partial I}{\partial x}(q_2) & \frac{\partial I}{\partial y}(q_2) \\ \vdots & \vdots \\ \frac{\partial I}{\partial x}(q_n) & \frac{\partial I}{\partial y}(q_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} \frac{\partial I}{\partial t}(q_1) \\ \frac{\partial I}{\partial t}(q_2) \\ \vdots \\ \frac{\partial I}{\partial t}(q_n) \end{pmatrix}$$

Soit de la forme $Ax = b$. De plus x est solution s'il minimise $\|Ax - b\|_2 = \min \{Ay - b, y \in \mathbb{R}^2\}$ et on retrouve là un problème de moindre carrés.

2.3 Méthodes Numériques

2.3.1 OpenCV

OpenCV (pour Open Source Computer Vision Library) est une bibliothèque Python spécialisée dans le traitement d'images en temps réel. Ses utilisations sont très diverses, et vont de la lecture et du traitement d'une image jusqu'au calcul du flot optique entre 2 frames, ceci étant ce qui nous intéresse particulièrement.

2.3.2 Méthode TV-L1

[8] La méthode TV-L1 est une méthode qui se base sur la minimisation de la fonctionnelle suivante :

$$E(u) = \int_{\Omega} (|\nabla u_1| + |\nabla u_2| + \lambda |\rho(u)|)$$

Où :

- $u = (u_1, u_2)$ est le champ de flux.
- $|\nabla u_1| + |\nabla u_2|$ est le terme de régularisation.
- ρ est la forme linéarisée de $I_1(x + u) - I_0(x)$ où I_0 est la luminosité de la première image, et I_1 celle de la deuxième.
- λ est le facteur de régularisation : plus λ est grand, plus le résultat sera lisse. Au contraire, pour λ petit, le résultat sera plus proche de la réalité, mais aussi possiblement plus bruité.

Enfin, avec son implémentation dans OpenCV, l'algorithme TV-L1 prend également en compte un deuxième paramètre, θ . Celui-ci contrôle la vitesse de convergence et la stabilité du schéma numérique. Si θ est trop petit, la convergence peut-être trop lente et moins stable. Par contre, pour θ plus grand, la convergence sera plus rapide et moins instable. Les résultats peuvent toutefois devenir incohérents.

Les valeurs typiques utilisées sont $\lambda = 0.1$, $\theta = 0.3$. Pour calculer le flot optique entre deux frames, on utilise donc le code suivant :

```

1 def optical_flow(frame1, frame2):
2     gray1 = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
3     gray2 = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY) #on importe les images
4     optical_flow = cv2.optflow.DualTVL1OpticalFlow_create()
5     optical_flow.setLambda(0.1) # On choisit la valeur de lambda
6     optical_flow.setTheta(0.3) # On choisit la valeur de theta
7     flow = optical_flow.calc(gray1, gray2, None)
8     return flow

```

2.3.3 Implémentation et premiers résultats

L'affichage du flot optique superposé sur la première frame se fait en Python de la manière suivante, via toujours la bibliothèque OpenCV.

```

1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 def optical_flow(frame1, frame2):
6     gray1 = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
7     gray2 = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)
8     optical_flow = cv2.optflow.DualTVL1OpticalFlow_create()
9     optical_flow.setLambda(0.1)
10    optical_flow.setTheta(0.3)
11    flow = optical_flow.calc(gray1, gray2, None)
12    return flow
13
14 def visu_optical_flow(flow, start_frame, min_magnitude=1.0):
15     vis = start_frame.copy()
16     H, W = flow.shape[:2]
17     #mag correspond à l'intensité du mouvement
18     #ang correspond à la direction du mouvement
19     mag, ang = cv2.cartToPolar(flow[..., 0], flow[..., 1])
20     mean_mag = np.mean(mag)
21     scale = max(1, 10 / (mean_mag + 1e-6)) #echelle de visualisation en fonction
22     #de la moyenne de l'intensité du mouvement
23     step = 25 # espacement des flèches
24     for y in range(0, H, step):
25         for x in range(0, W, step):
26             dx, dy = flow[y, x]
27             if np.linalg.norm(np.array([dx, dy])) > min_magnitude: # ignorer le
28                 bruit faible
29                 end_point = (int(x + scale * dx), int(y + scale * dy))
30                 cv2.arrowedLine(vis, (x, y), end_point, (0, 255, 0), 1, tipLength
31                 =0.3)
32
33     vis_rgb = cv2.cvtColor(vis, cv2.COLOR_BGR2RGB) #opencv utilise BGR mais
34     #Matplotlib utilise RGB
35     plt.figure(figsize=(7, 6))
36     plt.imshow(vis_rgb)
37     plt.title("Optical Flow")
38     plt.axis("off")
39     plt.show()
40
41 if __name__ == "__main__":
42     frame1 = cv2.imread("1.png")
43     frame2 = cv2.imread("2.png")
44     flow = optical_flow(frame1, frame2)
45     visu_optical_flow(flow, frame1, min_magnitude=1.0)

```

On applique ceci par exemple sur la vidéo d'une route, entre les deux images suivantes [1].



Ainsi le flot optique devrait être de la forme :

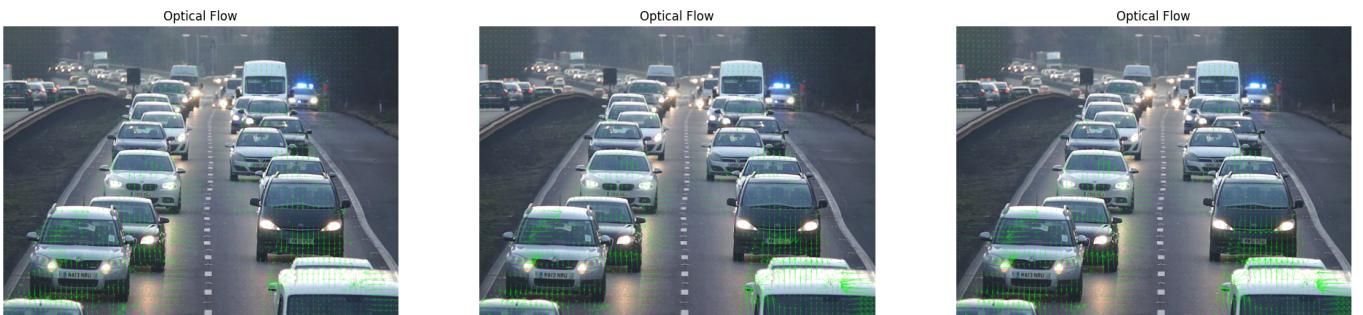


L'exécution du programme nous renvoie toutefois

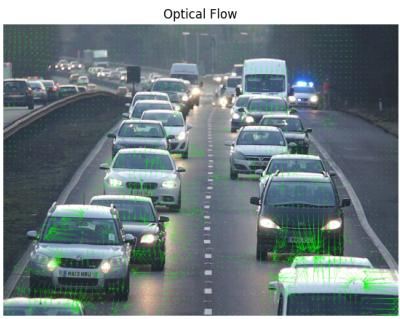


Ceci est assez cohérent. En effet, on voit certaines flèches qui ne vont pas dans le sens attendu, ce qui est dû à l'hypothèse faite sur la constance de l'illumination, qui n'est pas vérifiée dans la réalité. De plus, les véhicules du fond ne forment pas de flèche car ne semblent pas bouger, en raison de l'effet parallaxe qui est un effet visuel dans lequel les éléments les plus proches du viewer se déplacent plus rapidement que ceux situés en arrière-plan.

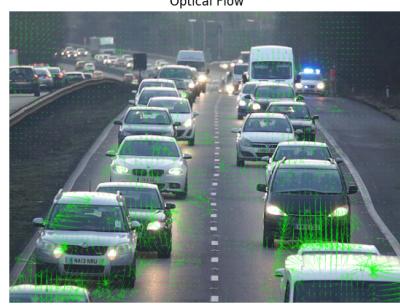
Essayons à présent de changer les paramètres. On fait varier successivement λ et θ



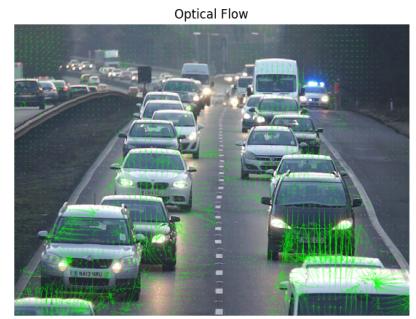
Ceci nous confirme donc le rôle des paramètres λ et θ expliqué précédemment. Plus la valeur de θ est faible, plus le flot sera précis. A contrario, plus on augmente θ , plus il y aura de bruit sur le flot et moins celui-ci sera lisse. Par



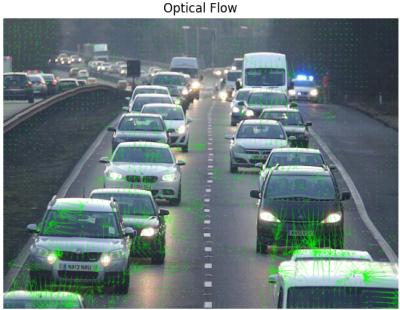
$\lambda = 0.5, \theta = 0.1$



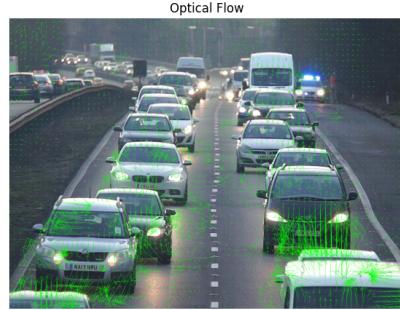
$\lambda = 0.5, \theta = 0.5$



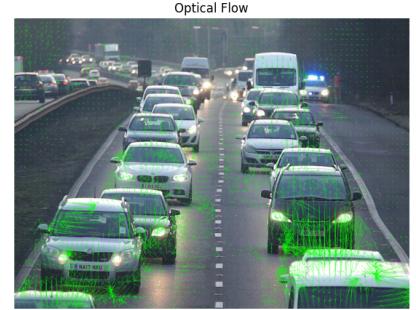
$\lambda = 0.5, \theta = 0.9$



$\lambda = 0.9, \theta = 0.1$



$\lambda = 0.9, \theta = 0.5$



$\lambda = 0.9, \theta = 0.9$

contre, plus λ est petit, moins il y a de bruit dans l'image qui est pris en compte. Par contre, plus λ est petit, plus le flot est lissé et donc possiblement un peu moins réaliste. Ceci nous invite donc à garder notre choix : $\lambda = 0.1, \theta = 0.3$. À présent essayons sur d'autres vidéos.



Première image



Deuxième image

On s'attend donc à un flot de la forme suivante :



L'exécution du programme précédent donne quant à elle :



$\lambda = 0.1, \theta = 0.1$



$\lambda = 0.1, \theta = 0.5$



$\lambda = 0.1, \theta = 0.9$



$\lambda = 0.5, \theta = 0.1$



$\lambda = 0.5, \theta = 0.5$



$\lambda = 0.5, \theta = 0.9$



$\lambda = 0.9, \theta = 0.1$



$\lambda = 0.9, \theta = 0.5$



$\lambda = 0.9, \theta = 0.9$

À nouveau, le choix $\lambda = 0.1, \theta = 0.3$ semble être le plus cohérent. L'estimation est toutefois assez mauvaise dans ce cas car on n'est pas dans le cas de faibles déplacements, contrairement à précédemment.

3 Estimation de Flot Optique par réseaux de neurones

3.1 Les Réseaux de Neurones

3.1.1 Principe

Les réseaux de neurones classiques reposent sur le modèle du Perceptron [7]. Ce dernier, développé par F. Rosenblatt est un classificateur biclassé. Il prend en entrée d valeurs scalaires (x_1, \dots, x_d) et renvoie +1 si $w_0 + x_1 w_1 + \dots + x_d w_d > 0$ et -1 sinon. Les w_i sont les paramètres du modèle.



FIGURE 1 – Le perceptron de Rosenblatt

Un réseau de neurones est une composition de neurones formels, inspirés du perceptron de Rosenblatt, organisés en plusieurs couches. La présence de couches cachées et de fonctions d'activation non linéaires permet d'apprendre des fonctions non linéaires, contrairement au perceptron, dont la fonction d'activation est une fonction seuil.

3.1.2 Les ConvNet

3.1.3 Les U-Net

[6]

3.2 Estimation de Flot Optique par apprentissage supervisé

3.2.1 L'architecture FlowNet

Le FlowNet [3] est un réseau de neurones qui permet de calculer de manière supervisée le flot optique entre deux images consécutives. Ce dernier se base sur l'architecture U-Net décrite précédemment, et l'architecture est la suivante.

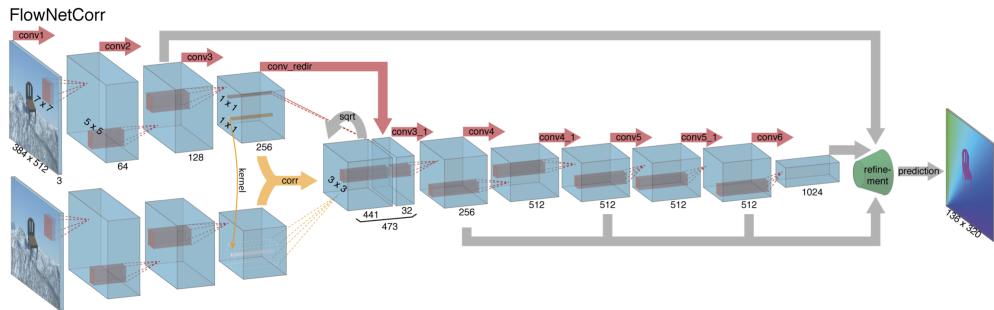


FIGURE 2 – La partie encodeur

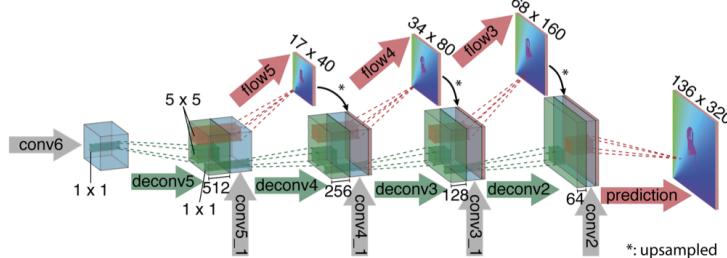


FIGURE 3 – La partie decodeur

Le modèle du FlowNet se base sur la loss Average Endpoint Error définie comme suit :

$$AEE(F, F_{gt}) = \frac{1}{HW} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \sqrt{(u(i,j) - u_{gt}(i,j))^2 + (v(i,j) - v_{gt}(i,j))^2}$$

où $F := (u, v)$ désigne le flot optique estimé et $F_{gt} := (u_{gt}, v_{gt})$ désigne le vrai flot optique. H est la hauteur (en pixels) et W la largeur (également en pixels) de l'image.

3.2.2 Modèle simplifié

Pour des raisons de simplicité, on implémente plutôt le modèle FlowNetS, qui garde le même principe mais simplifie la partie encodeur du U-Net, tout en gardant le même décodeur et la même loss.

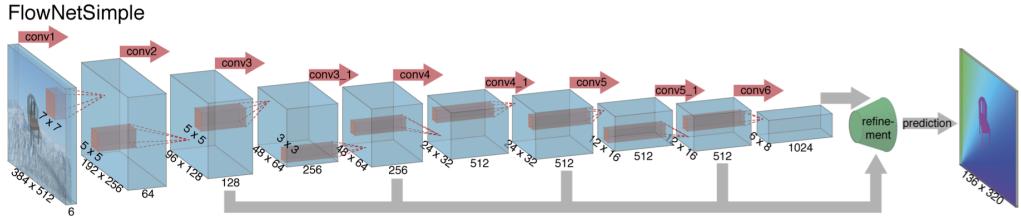


FIGURE 4 – La partie encodeur du réseau FlowNetS

De plus, en raison de contraintes informatiques, on ne garde que dans un premier temps 3 couches convolutives et 3 couches déconvolutives sur le même principe que le FlowNetS, pour accélérer l'entraînement.

3.2.3 Limites de l'apprentissage supervisé

Pour estimer le flot optique via un réseau neuronal convolutif en apprentissage supervisé, il est nécessaire d'avoir non seulement les deux images successives, mais aussi la réalité, c'est-à-dire le véritable flot optique entre les deux images. Comme on ne peut pas obtenir celui-ci pour des images quelconques, il est donc nécessaire de générer des images par ordinateur pour obtenir le véritable flot optique correspondant. On entraîne donc notre réseau de neurones sur le dataset FlyingChairs [2]. Il est constitué de 22872 paires d'images avec pour chaque paire le flot optique correspondant.

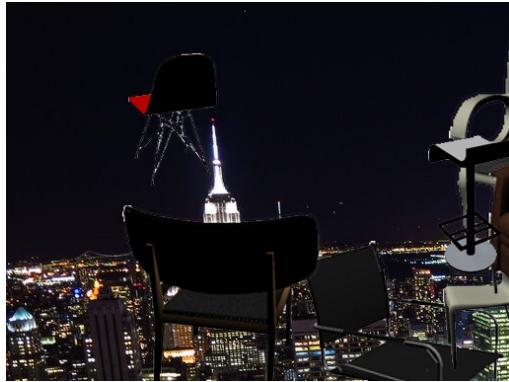


FIGURE 5 – Première image



FIGURE 6 – Seconde image



FIGURE 7 – Flot Optique correspondant

Pour lire ce flot optique, il faut utiliser la roue de couleurs suivante, appelée roue de Middlebury. La direction est indiquée par la couleur et la norme par l'intensité de la couleur.

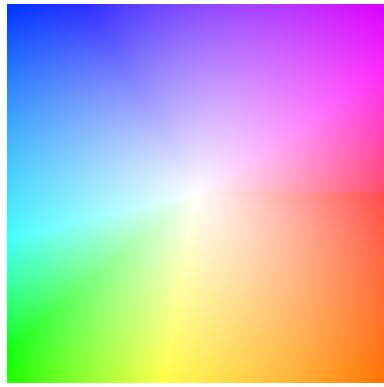
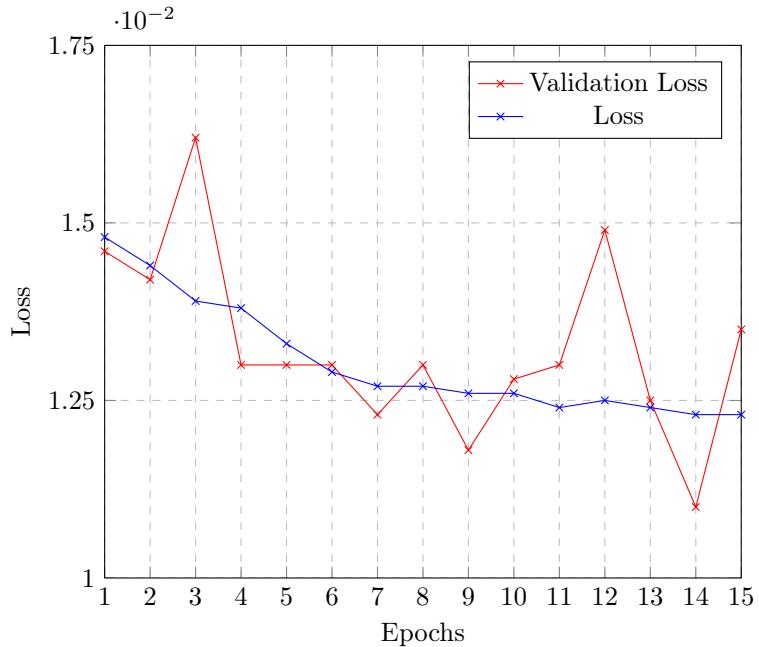


FIGURE 8 – Roue de couleurs de Middlebury

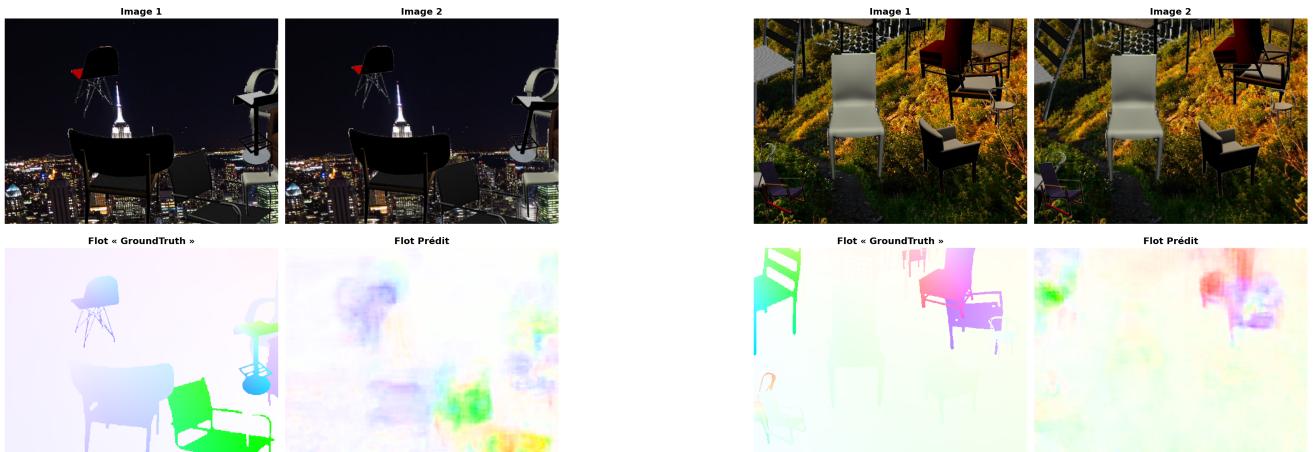
Donc dans l'image précédente ; la chaise en bas à droite se dirige dans la direction Sud-Est par exemple.

3.2.4 Résultats

Après un entraînement de 15 epochs, sur le set FlyingChairs, on obtient la courbe de loss suivante.



On constate que la loss décroît, ce qui nous confirme que le réseau a bien appris lors de ces epochs. Toutefois ces résultats sont moyennement satisfaisants, car la décroissance de la loss n'est que d'environ 20% et la validation loss ne croit qu'en pics. On peut toutefois observer sur les images que les résultats nous permettent d'identifier du mouvement et la direction globale, mais la position reste assez floue. On constate aussi que quelques erreurs sont générées : sur le premier exemple on observe une tâche jaune non présente dans la groundtruth.



3.3 Estimation de Flot Optique par apprentissage non-supervisé

3.3.1 Motivations

En vue de ce qui a été expliqué précédemment, notamment à propos de la difficulté d'obtenir des images avec le flot optique sans les générer par ordinateur, on s'intéresse donc à l'estimation de flot optique par apprentissage non-supervisé, contrairement à ce qui a été fait précédemment.

3.3.2 Approche technique

Le principe [9] pour calculer le flot par apprentissage non-supervisé est de garder l'architecture présentée plus haut du FlowNet Simple, mais d'adapter la loss :

$$\mathcal{L}(u, v, I(x, y, t), I(x, y, t + 1)) = \ell_{\text{photometric}}(u, v, I(x, y, t), I(x, y, t + 1)) + \lambda \ell_{\text{smoothness}}(u, v)$$

Où :

- $u, v \in \mathbb{R}^{H \times W}$ sont les composantes horizontales et verticales du flot prédit.
- $\ell_{\text{photometric}}(u, v, I(x, y, t), I(x, y, t + 1)) = \sum_{i,j} \rho_D(I(i, j, t) - I(i + u_{i,j}, j + v_{i,j}, t + 1))$
- $\ell_{\text{smoothness}}(u, v) = \sum_j^H \sum_i^W (\rho_S(u_{i,j} - u_{i+1,j}) + \rho_S(u_{i,j} - u_{i,j+1}) + \rho_S(v_{i,j} - v_{i+1,j}) + \rho_S(v_{i,j} - v_{i,j+1}))$
- $\rho_{S,D} : x \mapsto (x^2 + \varepsilon^2)^{\alpha_{S,D}}$ est la fonction de Charbonnier généralisée.
- λ est un paramètre de régularisation qui décide de l'importance relative pour le flot prédit soit lisse ou non.

4 Annexes

Références

- [1] Mike BIRD. <https://www.pexels.com/fr-fr/video/flux-de-trafic-sur-l-autoroute-2103099/>. 2019.
- [2] A. DOSOVITSKIY et al. “FlowNet : Learning Optical Flow with Convolutional Networks”. In : *IEEE International Conference on Computer Vision (ICCV)*. 2015. URL : <http://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15>.
- [3] Philipp FISCHER et al. *FlowNet : Learning Optical Flow with Convolutional Networks*. <https://arxiv.org/abs/1504.06852>. 2015.
- [4] Berthold K.P. HORN et Brian G. SCHUNCK. *Determining Optical Flow*. <https://dspace.mit.edu/handle/1721.1/6337>. 1981.
- [5] Bruce D. LUCAS et Takeo KANADE. *An Iterative Image Registration Technique with an Application to Stereo Vision*. <https://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdf>. 1981.
- [6] Olaf RONNEBERGER, Philipp FISCHER et Thomas BROX. *U-Net : Convolutional Networks for Biomedical Image Segmentation*. [https://arxiv.org/pdf/1505.04597](https://arxiv.org/pdf/1505.04597.pdf). 2015.
- [7] Frédéric SUR. *Introduction à l'apprentissage automatique*. https://members.loria.fr/FSur/enseignement/apprauto/poly_apprauto_FSur.pdf. 2025.
- [8] A. WEDEL et al. *An Improved Algorithm for TV-L1 Optical Flow*. https://cvg.cit.tum.de/_media/spezial/bib/dagstuhlopticalflowchapter.pdf. 2012.
- [9] Jason J. YU, Adam W. HARLEY et Konstantinos G. DERPANIS. *Back to Basics : Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness*. <https://arxiv.org/abs/1608.05842>. 2016.