


Université Pierre et Marie Curie

# Rapport projet

# PRAT : Flot optique et machine learning

Bian Xiong
3410676
bian.xiong@etu.upmc.fr

11 février 2018

# Table des matières

# Chapitre 1

# Introduction

Le calcul du flot optique consiste à établir un champs dense de vecteurs de correspondance entre deux images, permettant d'estimer le déplacement des objets d'une scène. Actuellement, le réseau de neurones convolutionnnel est très utile pour la segmentation et classification d'image. Beaucoup d'architectures de réseau de neurones convolutionnel sont proposées pour le traitement d'image : VGG, ResNet, googleNet etc. A partir de cette base, nous pouvons construire un réseau de neurones convolutionnnel (CNN : convolutional neural network) qui est capable de résoudre le problème d'estimation de flot optique .

Dans ce projet, nous avons étudié les modèles de [1] FlowNet qui sont appliqués à estimer flot optique : Les deux architectures sont par apprentissages supervisés : FlowNetS et FlowNetC. Le code source et démonstrateur sont disponibles en ligne et on peut l'implémenter facilement, mais l'environnement n'est pas facile à adapter. Alors nous pouvons faire la mise en oeuvre grâce à Pytorch. Après réaliser la méthode par apprentissages supervisés, nous implémentons la méthode par apprentissage non-supervisée basé sur l'article [6].

Enfin, nous faisons la comparaison entre les méthodes classiques et les méthodes par apprentissage. Nous trouvons que les méthodes par apprentissage sont plus rapides et précises que les méthodes classiques. Et de plus, parmi les trois méthodes par apprentissage, nous observons que pour la vitesse : FlowNet non-supervisé $>$ FlowNetS $>$ FlowNetC, et pour la précision : FlowNetC $>$ FlowNetS $>$ FlowNet non-supervisé.

# Chapitre 2

# Approche technique

## 2.1 Flot optique

Flot optique, est un champ de vitesse mesuré à partir de la luminance.
– (x,y) représente les coordonnées d'un pixel d'une image, l'origine étant le coin haut gauche.
– $(v_x, v_y)$ est la vitesse dans le plan de l'image $v_x = \frac{dx}{dt}, v_y = \frac{dy}{dt}$. On note

$$w = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{2.1}$$

## 2.2 Méthode classique : Méthode de Horn et Schunck

Nous avons étudié la méthode globale de Horn et Schunck dans les cours de VISION. C'est premier algorithme de calcul du flot optique. Le modèle consiste à trois parties :
- Invariance de la luminosité (Contrainte du flot optique)
- Régularité du champs de vitesse : une contrainte sur la variation du champs des vitesses suffit à lever l'indétermination.
- Formulation sous forme une fonction de coˆut (erreur) 'a minimiser.
Nous minimisons l'énergie suivant :

$$E(w) = \int_\Omega (w. \bigtriangledown I + I_t)^2 dxdy + \alpha \int_\Omega || \bigtriangledown w||^2 dxdy \tag{2.2}$$

Premier terme : contrainte du flot optique. Attache aux donnée, énergie externe.
Second terme : contrainte de régularisé sur $w$, Régularisation, énergie interne.
Problème : résultat assez médiocre, sensible au bruit.
En appliquant la définition de Gâteau, et intégrant par partie, nous trouvons que :

$$\bigtriangledown E(w) = \begin{bmatrix} \frac{\partial E(u,v)}{\partial u} \\ \frac{\partial E(u,v)}{\partial v} \end{bmatrix} = \begin{bmatrix} 2I_x(I_x u + I_y v + I_t) - 2\alpha \bigtriangleup u \\ 2I_y(I_x u + I_y v + I_t) - 2\alpha \bigtriangleup v \end{bmatrix} \tag{2.3}$$

Il faut donc résoudre :

$$\begin{aligned} I_x(I_x u + I_y v + I_t) - 2\alpha \bigtriangleup u = 0 \\ I_y(I_x u + I_y v + I_t) - 2\alpha \bigtriangleup v = 0 \end{aligned} \tag{2.4}$$

Nous appliquons une méthode de type Gauss-Seidel, la suite $(u^k, v^k)$ définit par :

$$\begin{aligned} (\alpha + I_x^2 + I_y^2)(u^{k+1} - \bar{u}^k) = -I_x(I_x \bar{u}^k + I_y \bar{v}^k + I_t) \\ (\alpha + I_x^2 + I_y^2)(v^{k+1} - \bar{v}^k) = -I_y(I_x \bar{u}^k + I_y \bar{v}^k + I_t) \end{aligned} \tag{2.5}$$

et $u^0 = v^0 = 0$
Après l'algorithme de Horn et Schunck est sous la forme :
1. Calculer les gradients spatio-temporel de $I(I_x, I_y, I_t)$
2. Choix du nombre d'itérations : un grand nombre (convergence lente)
3. Choix du paramètre $\alpha$ : il est fixé arbitrairement (il peut être calibré sur des données test)

4. Initialiser le champs de vecteur vitesse $(u^0, v^0)$ à zéro.
5. Itérer :

$$u^{k+1} = \bar{u}^k + \frac{-I_x(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha + I_x^2 + I_y^2}$$
$$v^{k+1} = \bar{v}^k + \frac{-I_x(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha + I_x^2 + I_y^2} \tag{2.6}$$

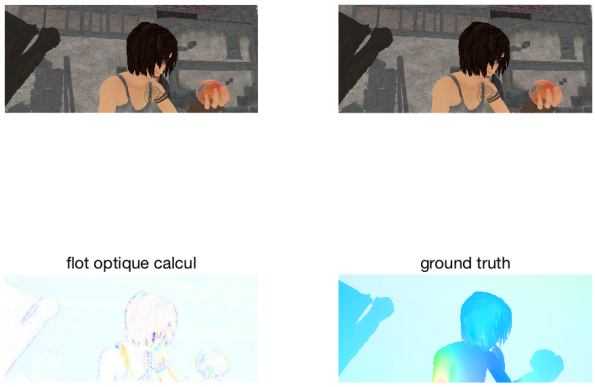



FIGURE 2.1 – Résultat de méthode de Horn et Schunck avec Itération 100 et $\alpha = 20$

Nous trouvons que la méthode de Horn et Schunck dépense un peu de temps avec un grand nombre d'itération, et le résultat donne une forme similaire mais pas précise.

## 2.3 Méthodes par apprentissage

### 2.3.1 L'architecture de CNN

Notre objet du sujet est réaliser un réseau de neurones convolutionnel pour estimer le flot optique. Nous introduisons l'architecture de CNN classique.

Dans un CNN, chaque filtre sert comme détecteur de caractéristiques, alors nous obtenons une description riche pour analyser les images.

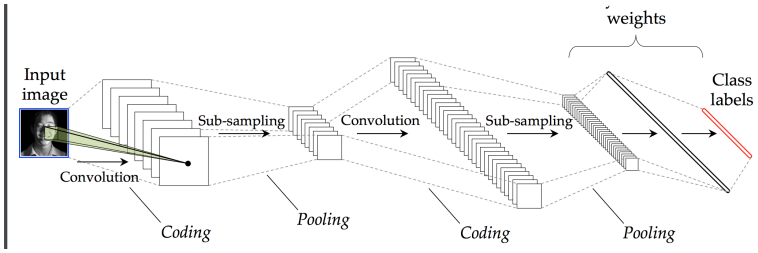



FIGURE 2.2 – CNN architecture complet

Une architecture de réseau de neurones convolutionnel est formée par un empilement de couches de traitement :

-La couche de convolution (CONV) qui traite les données d'un champ récepteur.

-La couche de pooling (POOL), qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage).

-La couche de correction (ReLU), souvent appelée par abus « ReLU » en référence à la fonction d'activation (Unité de rectification linéaire).

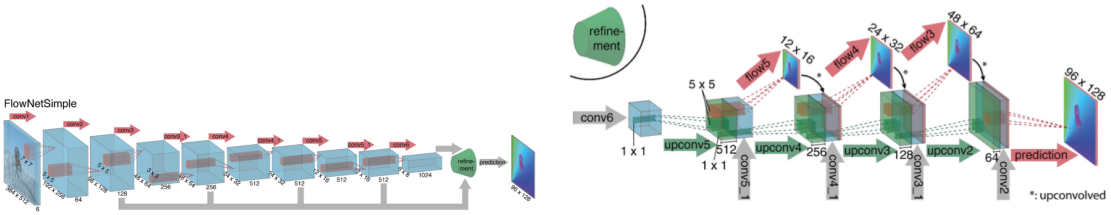
-La couche « entièrement connectée » (FC), qui est une couche de type perceptron.

-La couche de perte (LOSS).

### 2.3.2 L'architecture de FlowNetS

Un choix simple consiste à empiler les deux images d'entrée et à les alimenter à travers un réseau plutôt générique, permettant au réseau de décider lui-même comment traiter la paire d'images pour extraire les informations de mouvement. Nous appelons cette architecture composée uniquement de couches de convolution "FlowNetSimple".

L'architecture de FlowNetS consiste à deux parties : une partie à extraire les feature maps, une partie de raffinement : transformer les feature maps à flot optique, augmenter la résolution. Nous précisons la partie de raffinement. Dans cette partie, il y a des couches de upconvolution et des couches de upsampling. les deux étapes peuvent augmenter la résolution deux fois. Chaque donné consiste à trois parties dans le raffinement : une qui vient de la sortie de la couche de convolution au même niveau, une venant de la sortie de prédiction avant, une sortie de la couche de upconvolution. Enfin nous pouvons obtenir une sortie de flot optique de résolution de 96x128, mais pas 384x512 comme origine, nous allons expliquer les approches après pour avoir une même résolution que l'origine.




(a) FlowNetS (b) Raffinement

FIGURE 2.3 – L'architecture de FlowNetS

### 2.3.3 L'architecture de FlowNetC

Une autre approche consiste à créer deux flux de traitement distincts mais identiques pour les deux images et à les combiner ultérieurement, le réseau est contraint de produire d'abord séparément des représentations significatives des deux images, puis de les combiner à un niveau supérieur. Cela ressemble à l'approche par appariement standard quand nous extrairons d'abord des caractéristiques à partir de patches des deux images et ensuite nous comparons ces vecteurs de caractéristiques.

Pour aider le réseau dans ce matching processus, on introduit une couche de corrélation qui effectue des comparaisons de correctifs multiplicatifs entre deux cartes de caractéristiques. La corrélation entre deux patches centré à $x_1$ dans la première carte et $x_2$ dans la seconde carte est définie par :

$$c(x_1, x_2) = \sum_{o \in [-k,k]*[-k*k]} < f_1(x_1 + o), f_2(x_2 + o) > \tag{2.7}$$

Soit une entrée de taille $w * h$, la sortie de la couche de corrélation est de taille $w * h * D^2$, donc nous devons limiter $D$ la taille de fenêtre. si $D$ est très grande, le calcul dépense trop de temps, dans le projet, nous choisissons $D = 21$. Avec la couche de corrélation, nous pouvons construire une correspondance entre deux images en séquence. Pour garder les informations origines, nous ajoutons une petite partie de convolution qui conserve les informations de première image de référence.

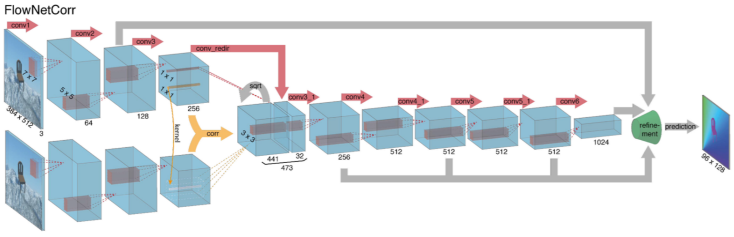



FIGURE 2.4 – L'architecture de FLowNetC

### 2.3.4 L'architecture de FlowNet non-supervisé

Les deux architectures avant sont par apprentissage supervisé. Ils ont besoin de la vérité terrain pour entraîner notre réseau. Maintenant, nous nous concentrons sur un FlowNet par apprentissage non-supervisé [6], Une fonction "Unsupervised loss" est introduite dans ce système.

$$\mathcal{L}(u,v;I(x,y,t),I(x,y,t+1)) = \ell_{photometric}(u,v;I(x,y,t),I(x,y,t+1)) + \lambda\ell_{smoothness}(u,v). \tag{2.8}$$

Où $u, v \in R^{H*W}$ sont les composantes horizontales et verticales du champ d'écoulement prédit, et $\lambda$ est un paramètre de régularisation qui pèse l'importance relative de la régularité du flux prédit. Notez que la perte photométrique peut être remplacée ou augmentée par d'autres mesures, telles que la constance du gradient d'image.

$$\ell_{photometric}(u,v;I(x,y,t),I(x,y,t+1)) = \sum_{i,j} \rho_D(I(i,j,t) - I(i+u_i, j+u_j, t+1)) \tag{2.9}$$

Où $\rho_D$ est la fonction de pénalité de données. Dans le projet, pour simplifier le processus, nous utilisons la fonction quadratique.

$$\ell_{smoothness}(u,v) = \sum_{j}^{H}\sum_{i}^{W}[\rho_S(u_{i,j} - u_{i+1,j}) + \rho_S(u_{i,j} - u_{i,j+1}) + \rho_S(v_{i,j} - v_{i+1,j}) + \rho_S(v_{i,j} - v_{i,j+1})) \tag{2.10}$$

Où $\rho_S(\cdot)$ est fonction de pénalité de finesse réalisée par la fonction de Charbonnier généralisée. Pour simplifier le processus, nous utilisons la fonction. quadratique.

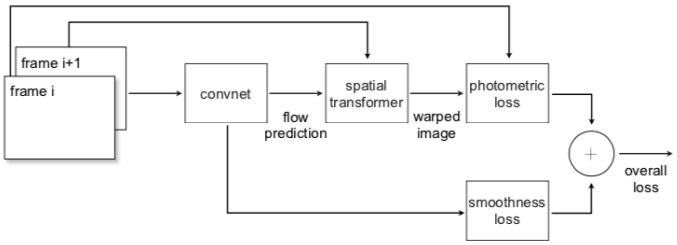



FIGURE 2.5 – L'architecture de approche non-supervisé

Après le FLowNetS, nous obtenons un flot optique, nous appliquons ce changement de flot pour transformer $I(t)$ à $I(t+1)$, et puis nous pouvons calculer les $loss_{photometric}$ et $loss_{smoothness}$ et les combiner pour avoir un $loss$ à estimer. Par minimiser ce loss, nous pouvons avoir un flot optique est optimal.

## 2.4 Méthodes utilisée pendant apprentissage

Pour entraîner notre réseau, nous utilisons les bases de donnée "flying chairs" qui sont générées par l'auteur de FlowNet, Les bases est assez grandes pour entraîner notre réseau, et ils ont un meilleur effet que les autres bases : KITTI et Sintel. Alors, nous entraîner notre réseau avec les bases flying chairs et testons sur les bases : flying chairs et Sintel.

### 2.4.1 Problème de résolution

Comme indiqué précédent, notre sortie de flot optique est de résolution 96x128, pas de 384*512, et nous trouvons que si nous continuons le processus de raffinement, le résultat n'est pas amélioré. Deux approches sont proposé.

**1. Upsampling bilinéaire** Nous appliquons une interpolation bilinéairie pour reconstruire notre flot optique ayant une résolution plus grande. C'est la méthode que nous utilisons en pratique.

**2. raffinement variationnel** Nous utilisons un modèle variationnel [11]. Nous construirons l'énergie par trois parties, et nous pouvons avoir un flot optique plus optimal en minimisant cette énergie.

$$\begin{aligned} E_{color}(w) &= \int \Psi(|I_2(x+w(x)) - I_1(x)|^2)dx \\ E_{grad}(w) &= \int_\Omega \Psi(|\bigtriangledown I_2(x+w(x)) - \bigtriangledown I_1(x)|^2)dx \\ E_{smoothness}(w) &= \int_\Omega \Psi(|\bigtriangledown u(x)|^2 + |\bigtriangledown v(x)|^2)dx \\ E(w) &= E_{color} + \gamma E_{gradient} + \alpha E_{smooth}. \end{aligned} \tag{2.11}$$

### 2.4.2 Data augmentation

Une stratégie largement utilisée pour améliorer la généralisation des réseaux de neutres est data augmentation. Même si la base Flying Chairs est assez volumineux, nous trouvons qu'il est crucial d'utiliser data augmentation pour éviter le sur-apprentissage. Les augmentations que nous utilisons incluent les transformations géométriques : translation, rotation et mise à l'échelle.

### 2.4.3 Fine-tuning

Les ensembles de données utilisés sont très différents en termes de types d'objets et de mouvements qu'ils incluent. Une solution standard consiste à affiner les réseaux sur l'ensemble de données cible. Pour la base Sintel, nous choisissons fine-tuning pour le training set.

### 2.4.4 Adam optimiseur

Adam [7](Adaptive Moment Estimation) est une méthode qui calcule les taux d'apprentissage adaptatifs pour chaque paramètre. Adam maintient également une moyenne décroissante exponentiellement des gradients passés $m_t$, semblable à momentum.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2.12}$$

$m_t$ et $v_t$ sont des estimations du premier moment (la moyenne) et du second moment (la variance non centrée) des gradients respectivement.

Dans le projet, Adam optimiseur performe mieux que les autres optimiseurs.

# Chapitre 3

# Résultat

Nous utilisons macbook pro, alors nous ne pouvons pas entraîner le réseau efficacement. Nous apprenons comment utiliser ssh qui réalise à tester avec GPU à distance. Nous connectons GPU avec ppti-gpu-1 : processeur Intel Xeon CPU E3 1275 v3 3,50 Ghz avec 8 coeurs de processeur, carte graphique NVDIA GK110 Geforce GTX Titan. Le processus dépense 12h pour entraîner le FlowNetS avec la base flying chairs. Nous faissons l'implémentation avec Pytorch.

## 3.1 Résultat de trois méthodes par apprentissage

Nous entraînons notre réseau avec la base flying chairs et testons avec la base Sintel. Nous comparons le EPE(endpoint error) pour les trois méthodes.

$$EPE = \frac{1}{mn}\sum_{i}^{m}\sum_{j}^{n}\sqrt{(u(i,j)-u_0(i,j))^2+(v(i,j)-v_0(i,j))^2} \tag{3.1}$$

Nous observons que les méthodes par apprentissage supervisé est plus précises que celle par apprentissage non-supervisé, et FlowNetC est plus pécis que FlowNetS dans certain cas. Si le déplacement est très grand, à cause de limite de taille de taille de fenêtre, FlowNetC ne donne pas un meilleur résultat que FlowNetS.

Pour la rapidité, FlowNet non-supervisé > FlowNetS > FlowNetC. FlownetS a 38,676,504 paramètres à entraîner et FlowNetC a 39,175,298 paramètres à entraîner car FlowNetC ajoute une couche de corrélation.


FIGURE 3.1 – Résultat de FlowNetS

TABLE 3.1 – Table EPE pour différentes méthodes

| méthode | flying chairs test | mpi-sintel test |
|---|---|---|
| FlowNetS | 5.5 | 7.42 |
| FlowNetC | 5.2 | 7.28 |
| FlowNet non-supervisé | 7.1 | 8.18 |


FIGURE 3.2 – Résultat de FlowNetC

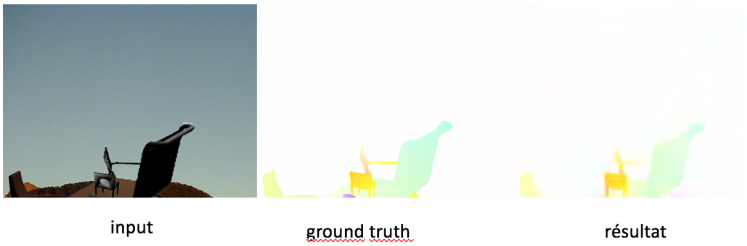



FIGURE 3.3 – Résultat de FlowNet non supervisé

# Chapitre 4

# Remerciements

Je veux adresser mes remerciements à mon directeur de projet, Dominique Béréziat , pour sa grande disponibilité et ses encouragements tout au long de la rédaction de ce projet.

# Bibliographie


[1] Eddy Ilg Philip Hausser Caner Hazırbas Alexey Dosovitskiy, Philipp Fischer and Vladimir Golkov. Flownet : Learning optical flow with convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[2] B. G. Schunck B.K.P. Horn. *Determining Optical Flow.* Artificial Intelligence.

[3] P. Gallagher Z. Zhang C. Lee, S. Xie and Z. Tu. Deeply supervised nets. In *AISTATS*, 2015.

[4] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0 : Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Z. Harchaoui J. Revaud, P. Weinzaepfel and C. Schmid. Epicflow : Edge-preserving interpolation of correspon- dences for optical flow. In *CVPR*, 2015.

[6] Adam W. Harley Jason J. Yu and Konstantinos G. Derpanis. Back to basics : Unsupervised learning of optical flow via brightness constancy and motion smoothness.

[7] Ba J. L Kingma, D. P. Adam : a method for stochastic optimization. In *International Conference on Learning Representations, 1-13*, 2015.

[8] Q. Yang L. Bao and H. Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *CVPR*, 2014.

[9] P. Hausser P. Fischer D. Cremers A. Dosovitskiy N. Mayer, E. Ilg and T. Brox. A large dataset to train convo- lutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[10] Z. Harchaoui P. Weinzaepfel, J. Revaud and C. Schmid. Deepflow : Large displacement optical flow with deep matching. In *ICCV*, 2013.

[11] T.Brox and J.Malik. Large displacement optical flow : descriptor matching in variarional motion estimation. In *PAMI.33(3) :500-513*, 2011.