Apprentissage robuste d'une carte de profondeur pour l'évitement d'obstacle dans le cas de caméras volantes, monoculaires et stabilisées

Soutenance de Thèse

Clément Pinard

25 juin 2019

U2IS, ENSTA ParisTech

Table des matières

Introduction et motivations

Contexte technologique

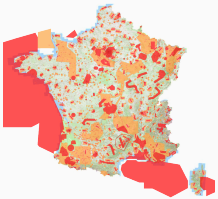
Le drone grand public

- Accessible et facile à piloter, même pour un néophyte.
- Images aériennes stabilisées et de qualité
- Modes de vol manuels et automatiques

Accessible à tous ?

- Trop dangereux en cas de crash pour être véritablement grand public !
- La croissance du marché est freinée par des lois de sécurité légitimes mais restrictives

PARROT Anafi (2018)



Carte des zones de vol interdit

But à long terme

Améliorer la sécurité

- Introduire un évitement d'obstacle automatique
- Utiliser le moins de hardware possible

Un évitement d'obstacle transparent

- Suivre la consigne, éventuellement directement fixée par l'utilisateur
- Garder la cinématographie en conservant un mouvement fluide

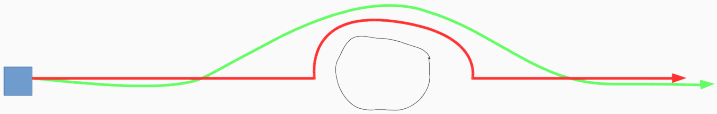


FIGURE 1 – La trajectoire verte est privilégiée

Introduction et motivations

Stratégie générale

Perception puis évitement

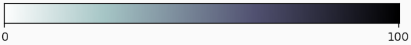
- On se concentre sur la perception des obstacles, avec seulement une caméra monoculaire.
- On essaye de trouver la carte de profondeur θ liée à une caméra stabilisée.

Carte de profondeur

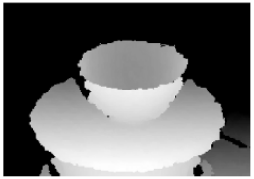
Pour chaque pixel d'une image, donner en plus de la couleur une indication de distance avec la profondeur.

Exemples de cartes de profondeur

KITTI, mesuré avec un LiDAR [Geiger et al., 2013]

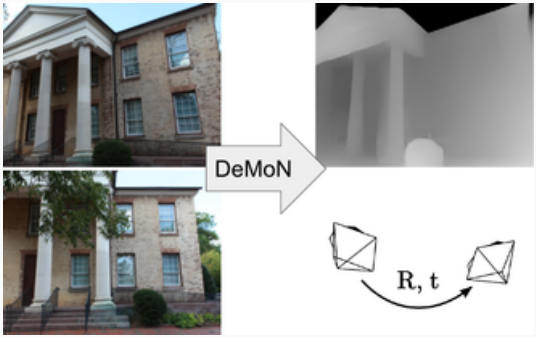


Mesurée avec une Microsoft Kinect [Lai et al., 2011]



Peut-on entrainer à un réseau de neurones à apprendre la profondeur θ ?

- Oui !
- Avec plusieurs images : DeMoN [Ummenhofer et al., 2017]
- Avec une seule image : DORN [Fu et al., 2018], MegaDepth[Li and Snavely, 2018]



Fonctionne très bien avec des vidéos de voiture, manque de robustesse pour des scènes inhabituelles.

Ces deux scènes sont plus plates que vous ne le pensez !

Si vous aviez eu une vidéo, vous n'auriez pas eu de problème ...

On ne peut estimer la profondeur $\widetilde{\theta}$ qu'à un **facteur d'échelle** près.

Il faut une grandeur physique disponible.

On propose de définir le facteur d'échelle avec une estimation de la vitesse $\widetilde{v}$, comparé à la vitesse réelle v.

	Précédemment [Zhou et al., 2017]	proposition
Predictions	Profondeur $\widetilde{\theta}$	Profondeur $\widetilde{\theta}$, Vitesse $\widetilde{v}$
Vérité terrain	Profondeur θ	Profondeur θ, Vitesse v
Évaluation	$m = \delta\left(\theta, \widetilde{\theta} \times \frac{Me(\theta)}{Me(\widetilde{\theta})}\right)$	$m = \delta\left(\theta, \widetilde{\theta} \times \frac{\lvert v \rvert}{\lvert \widetilde{v} \rvert}\right)$

Table 1 – $Me()$ est l'opérateur médian, et δ est une fonction de distance (e.g. distance L1)

Cette évaluation permet d'estimer $\widetilde{\theta}$ en supposant la vitesse constante.

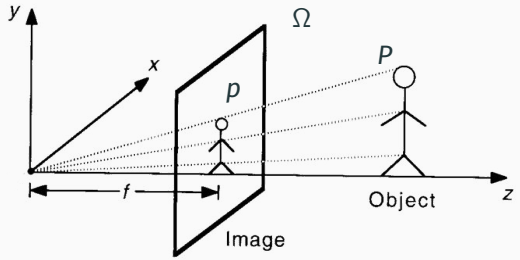
DepthNet Supervisé

Préambule

Modèle sténopé

Un point 3D $P = (X, Y, Z)$ est projeté sur le capteur en un pixel $p \in \Omega \subset \mathbb{R}^2$.

Le plan de pixel Ω a pour centre optique $p_0 = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$



Mise en équation

$$\begin{aligned} p &= \Pi(KP) \\ \theta(p) &= Z \end{aligned}$$

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \Pi : \quad \mathbb{R}^3 &\to \mathbb{R}^2 \\ P = (x, y, z) &\mapsto \left(\tfrac{x}{z}, \tfrac{y}{z}\right) \end{aligned}$$

$$\forall \alpha, \Pi(\alpha P) = \Pi(P)$$

Passage de p et θ à P

$$P = \theta(p) \times \mathrm{K}^{-1}\Pi_{-1}(p)$$

$$\begin{array}{rccc} \Pi_{-1}: & \mathbb{R}^2 & \to & \mathbb{R}^3 \\ & p = (u, v) & \mapsto & (u, v, 1) \end{array}$$

Passage de P à P_2

La caméra se déplace avec une rotation R et une translation $t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$

Nouvelle position du point P

$$P_2 = \mathrm{R}P + t$$

Nouvelle position du pixel

$$p_2 = \Pi(\mathrm{K}P_2) = \Pi\left(\theta(p)\mathrm{KRK}^{-1}\Pi_{-1}(p) + \mathrm{K}t\right)$$

Définition

Le flot optique est un champ de vecteurs représentant le déplacement des pixels d'une image à l'autre.

$$\begin{array}{rccc} F: & \Omega & \rightarrow & \mathbb{R}^2 \\ & p & \mapsto & (F_u, F_v) = p_2 - p \end{array}$$

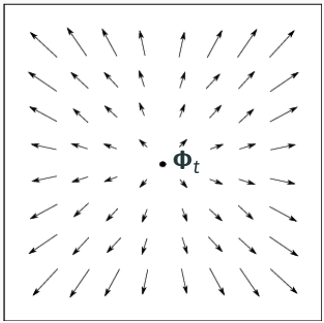
- On a déduit le flot optique à partir de la profondeur et du déplacement.
- On cherche à trouver la profondeur θ à partir du flot optique et du déplacement.

Définition

Projection du vecteur translation $\boldsymbol{t}$ sur le capteur.

$$\mathbf{\Phi}_t = \Pi(\mathrm{K}t)$$

C'est la direction vers laquelle la caméra se dirige



Le flot optique est proportionnel à la profondeur θ et à la distance entre $\mathbf{\Phi}_t$ et p

$$\forall p \in \Omega, F(p) = \Pi\left(\theta(t)\Pi_{-1}(p) + \mathrm{K}t\right) = \frac{t_z}{\theta(p) + t_z}\left(\mathbf{\Phi}_t - p\right)$$

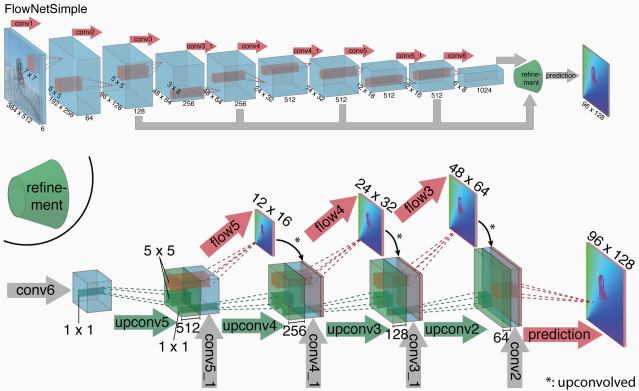
$$\forall p \in \Omega, \theta(p) = \frac{f\|t\|}{\sqrt{f^2 + \|\mathbf{\Phi}_t - p_0\|^2}} \left(\frac{\|\mathbf{\Phi}_t - p\|}{\|F(p)\|} - sign(t.u_z) \right)$$

Connaître le flot optique F et le point $\mathbf{\Phi}_t$ permet de déduire la profondeur.

DepthNet Supervisé

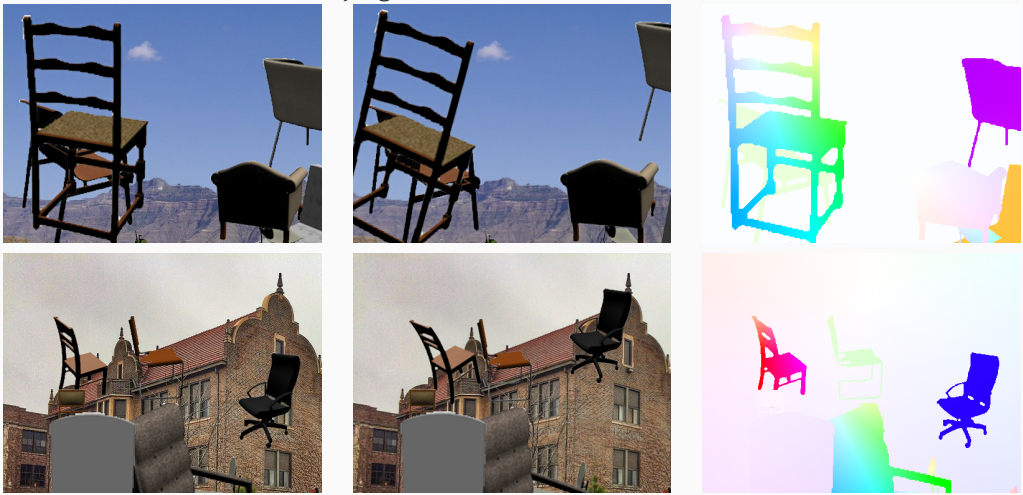
Estimation du flot optique

Réseau Convolutionnel entraîné pour estimer le flot optique d'une paire d'images



Le réseau sort plusieurs échelles du flot optique. Une des architectures les plus reprises car il est très simple.

L'entraînement est fait avec Flying Chairs, un dataset énorme mais irréaliste



Limitations de l'approche flot optique

$$\forall p \in \Omega, \theta(p) = \frac{f\|\mathbf{t}\|}{\sqrt{f^2 + \|\mathbf{\Phi}_t - p_0\|^2}} \left(\frac{\|\mathbf{\Phi}_t - p\|}{\|F(p)\|} - sign(\mathbf{t}.\mathbf{u}_z) \right)$$

Forme indéterminée en $p = \mathbf{\Phi}_t$

Une erreur sur le flot optique estimé $\widetilde{F}$ sera décuplée sur $\widetilde{\theta}$

Le vecteur $\mathbf{\Phi}_t$ doit aussi être correctement estimé

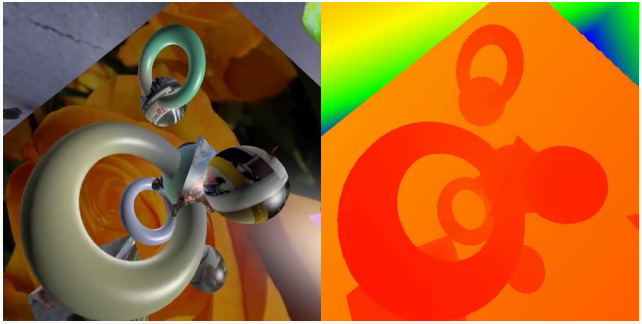
On veut faire apprendre à un réseau de neurones directement la profondeur au lieu d'apprendre le flot optique

DepthNet Supervisé

StillBox dataset

Un dataset synthétique représentatif d'une caméra volante stabilisée

- Scènes rigides avec formes primitives en suspention
- Aucune rotation dans le mouvement
- Fait automatiquement sur *Blender*



DepthNet Supervisé

Résolution du Problème

Architecture

Fully convolutional, très similaire à FlowNet, mais ne sort que la profondeur $\widetilde{\theta}$ à partir de deux images I_1 et I_2

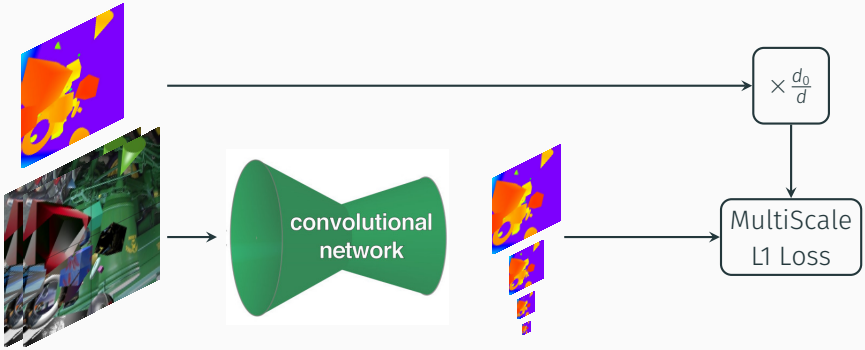
Apprentissage

La profondeur $\widetilde{\theta}$ est apprise avec un déplacement nominal d_0 constant.

La profondeur vérité terrain est compensée en fonction du déplacement effectif d entre I_1 et I_2. L'estimation $\widetilde{\theta}$ est retrouvée avec l'opération inverse.

$$\theta_{train} = \theta \frac{d_0}{d} \qquad \widetilde{\theta} = DepthNet(I_1, I_2) \frac{d}{d_0}$$

Le vecteur translation $\boldsymbol{t}$ n'est **jamais donné**, seulement sa magnitude d

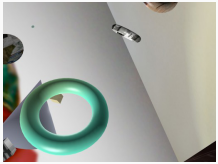
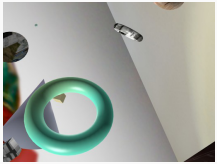
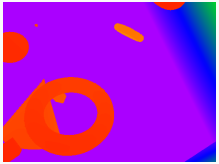
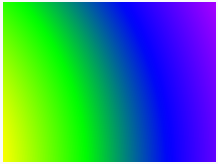
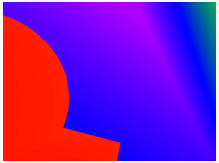
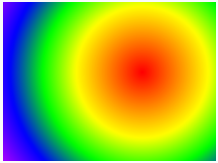


$$\mathcal{L} = \sum_{s \in scales} \gamma_s \frac{1}{|\Omega|} \sum_{p \in \Omega} |Out_s(p) - \theta_{train}(p))|$$

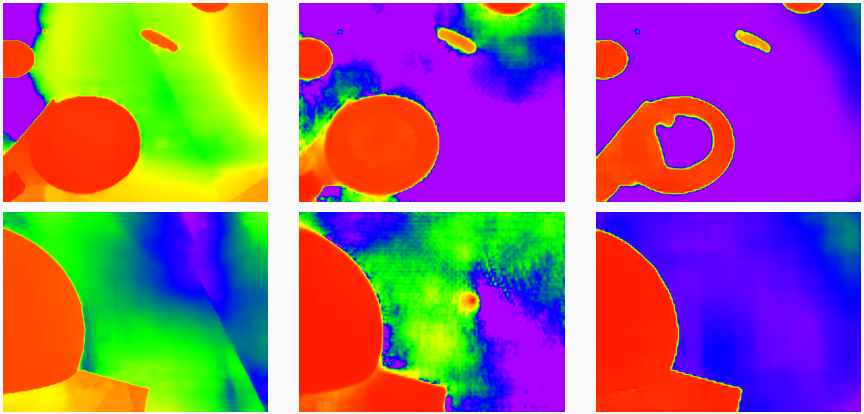
DepthNet Supervisé

Resultats

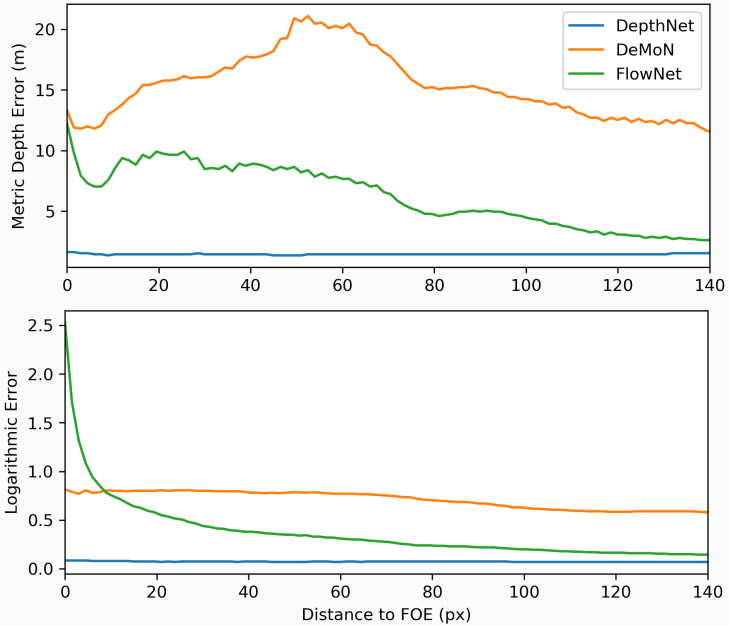
Résultats Quantitatifs sur StillBox

Img 1	Img 2	Ground Truth	distance to FOE $\|\mathbf{\Phi}_t - p\|$
			
			

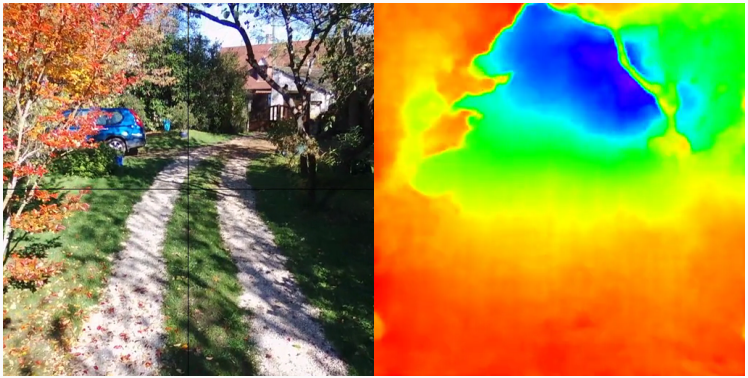
DeMoN FlowNet DepthNet



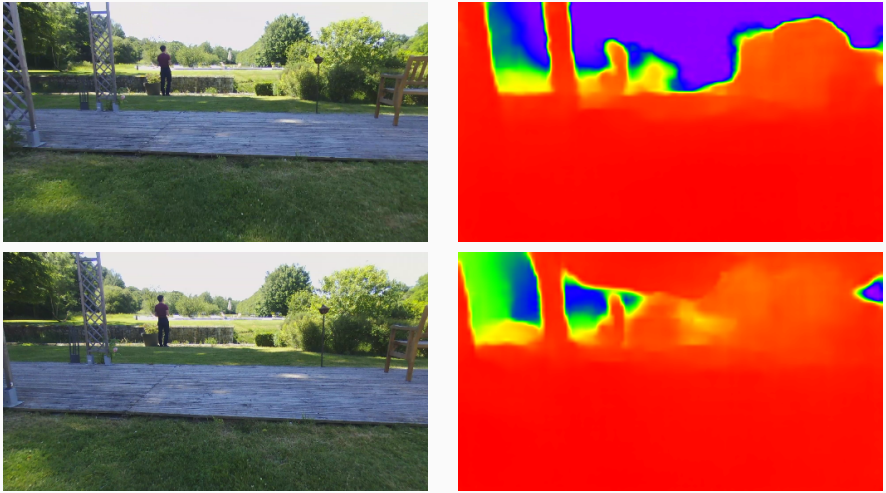
Method	Erreur absolue moyenne (m)	Erreur log moyenne
DepthNet	**2.67**	**0.132**
FlowNetS $\rightarrow$ depth	6.462	0.172
DeMoN	18.85	0.726


DepthNet
DeMoN
FlowNet
Metric Depth Error (m)
20
15
10
5
0
20
40
60
80
100
120
140
Logarithmic Error
2.5
2.0
1.5
1.0
0.5
0.0
Distance to FOE (px)

Résultats staisfaisant sur des vidéos stabilisées réelles.



Pas toujours ! Problèmes notamment avec le ciel...



Il nous faut un *finetuning sur des vidéos de drone*

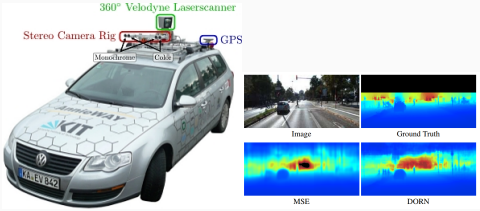
DepthNet Supervisé

Transfert aux vidéos réelles

L'apprentissage supervisé est-il viable ?

Dataset réel

- Construire un dataset avec profondeur est long et compliqué
- La profondeur est souvent incomplète.



Dataset synthétique

- Plus facile de réaliser beaucoup d'exemples.
- Le rendu peut être photoréaliste [Shah et al., 2017], mais la construction d'une scène complexe est difficile.

L'apprentissage non supervisé permettrait d'avoir un domaine d'apprentissage complexe, realiste et facile à construire

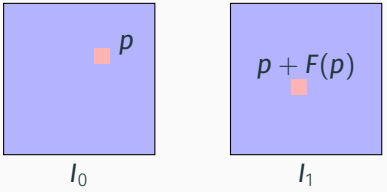
DepthNet Non Supervisé

Préambule

Hypothèse de la cohérence de la couleur

S'il y a un flot optique F entre deux images I_0 et I_1, alors on a une cohérence de la couleur.

$$\forall p \in \Omega, I_0(p) = I_1(p + F(p))$$



Méthode de Horn et Schunck [Horn and Schunck, 1981]

Tenter de reconstruire I_0 avec les pixels de I_1 pour former $\widetilde{I}_F$.

$$\forall p \in \Omega, \widetilde{I}_F = I_1(p + \widetilde{F}(p))$$

Trouver F revient à trouver le minimum de la fonction $\mathcal{L}_p$

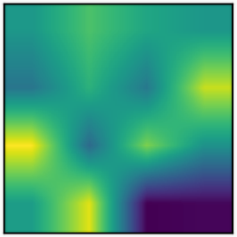
$$F = \arg\min_{\widetilde{F}} \mathcal{L}_p = \arg\min_{\widetilde{F}} (d(I_0, \widetilde{I}_{\widetilde{F}}))$$

Comment trouver le minimum de $\mathcal{L}_p$?

Optimisation

- On définit l'image sur $\mathbb{R}^2$ avec une interpolation bilinéaire.
- La construction de $\widetilde{I}_F$ est différentiable en $\widetilde{F}$.
- On peut minimiser $\mathcal{L}_p$ par une descente de gradient

$$\widetilde{F} \leftarrow \widetilde{F} + \mu \nabla_{\widetilde{F}} \mathcal{L}_p$$



Application à la profondeur

On reprend l'équation vue précédemment

$$F_{=\Pi}\left(\mathrm{KRK}^{-1}\Pi_{-1}(p)+\frac{\mathrm{K}t}{\theta(p)}\right)-p$$

- Tout est différentiable
- On peut trouver la profondeur θ d'une image par optimisation de $\mathcal{L}_p$.

DepthNet Non Supervisé

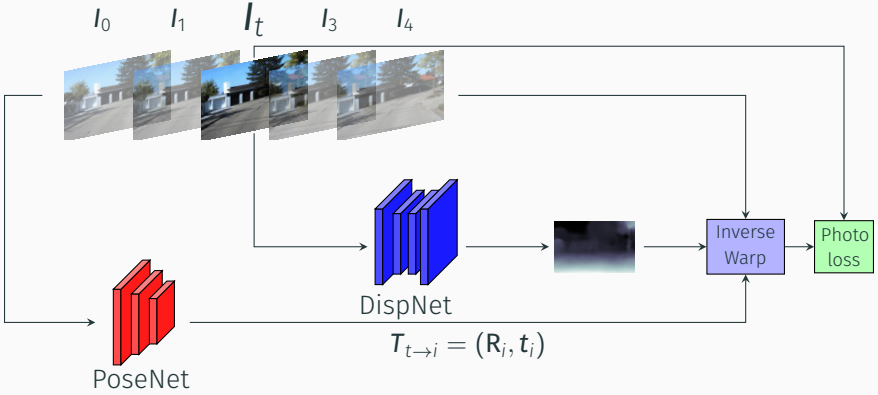
Présentation de SFMLearner

Solution naïve pour apprendre la profondeur sans la connaître

1. Estimer la profondeur $\widetilde{\theta}$ et le déplacement $(\mathbf{R}, t)$ par méthodes analytiques de flot optique. Par exemple, par optimisation de la fonction de coût $\mathcal{L}_p$.
2. Entraîner DepthNet avec pour consigne $\theta_{train} = \widetilde{\theta} \frac{d_0}{\|t\|}$

Idée de SFMLearner

- Entraîner directement un réseau DispNet pour minimiser $\mathcal{L}_p$
- Entraîner aussi un réseau PoseNet pour calculer le déplacement $(\mathbf{R}, t)$



Les réseaux PoseNet et DispNet tentent de minimiser les erreurs photométriques entre I_t et les images de réferences déformées $\widetilde{I}_{n\to t}$

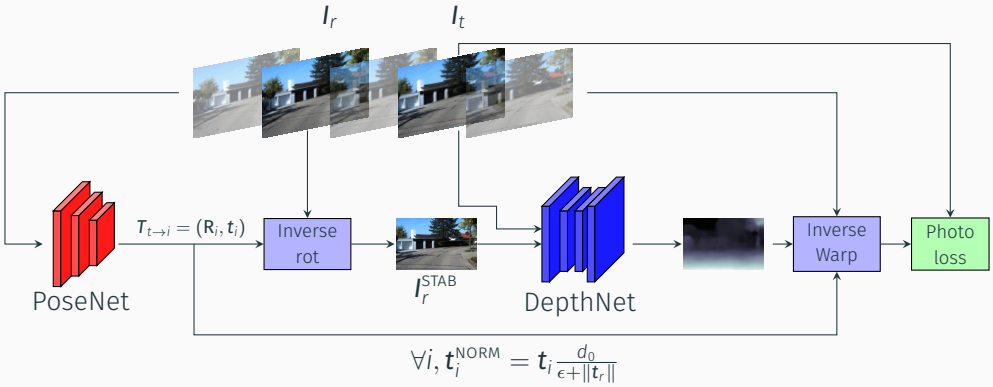
Stratégie

1. Remplacer DispNet par DepthNet.
2. Choisir dans la séquence une deuxième image I_r pour former une paire (I_r, I_t)
3. Stabiliser la paire donnée en $(\widetilde{I}_r, I_t)$ pour DepthNet grâce à PoseNet.
   (Pas besoin si les vidéos sont déjà stabilisées)

$$F_{stab} = \Pi \left(\mathrm{KRK}^{-1} \Pi_{-1}(p) \right) - p$$

4. Normaliser la translation de PoseNet. $\forall i, \boldsymbol{t}_i \leftarrow \frac{\boldsymbol{t}_i d_0}{\|\boldsymbol{t}_r\|}$

Adaptation à DepthNet



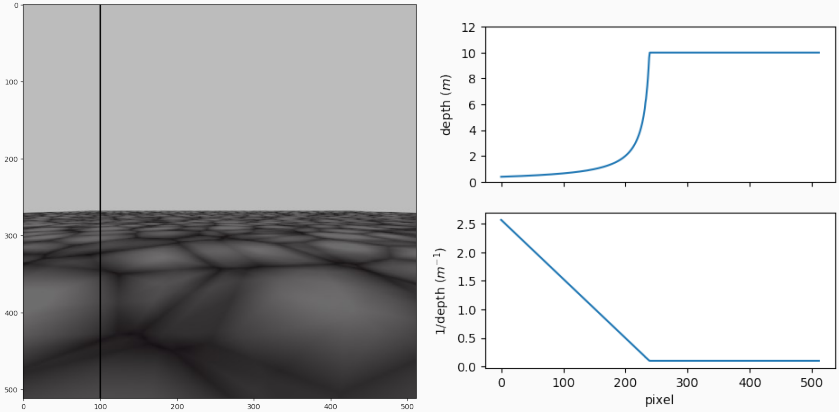
DepthNet Non Supervisé

Régularisation par lissage

Contrainte de lissage

Plusieurs solutions possibles permettent d'arriver à une erreur photométrique basse.

On choisit de contraindre l'optimisation pour avoir la solution la plus physiquement plausible.



Principe appliqué

Si le gradient d'une image $\|\nabla \boldsymbol{I}\|$ est bas, alors le Laplacien de sa profondeur inverse $\Delta\xi = \Delta\left(\frac{1}{\theta}\right)$ doit être bas aussi.

2 Idées

1. Rajouter une fonction de coût à l'erreur photométrique pour inciter $\Delta\xi$ à être bas.
2. Atténuer ce coût en fonction du gradient de l'image $\boldsymbol{I}$.

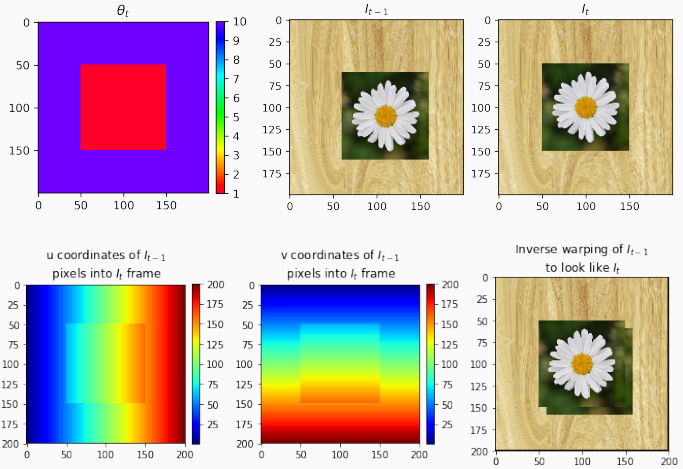
$$\mathcal{L}_{gdiff} = \iint_{\Omega} exp\left(-\left(\frac{\|\nabla \boldsymbol{I}\|}{\kappa}\right)^2\right)(\Delta\xi)^2 dS$$

DepthNet Non Supervisé

Régularisation par gestion des occultations

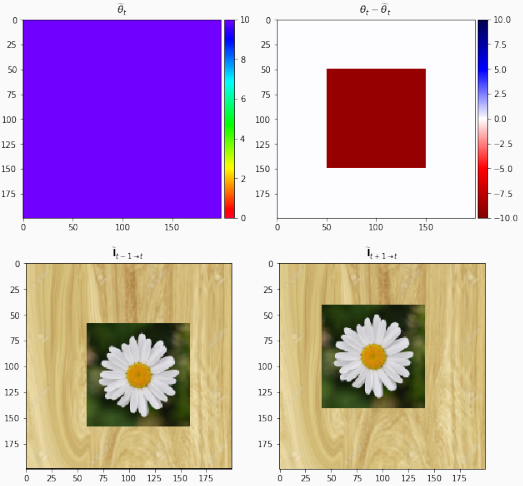
Gérer les occultations

Une profondeur parfaite ne minimise pas exactement l'erreur photométrique ! Exemple avec une scène typique.



- Ces zones ne doivent pas être optimisées car les pixels correspondant ne sont pas trouvables.
- Un autre, au mauvais endroit mais avec une meilleure couleur sera trouvé.
- Si on peut déduire les zones d'occultation, on les retire de l'erreur photométrique.

$$\mathcal{L}_{pf}(t_2) = \sum_{p \in \Omega} \begin{cases} d(I_t, \widetilde{I}_{t_2 \to t})(p) & \text{si } p \text{ pas occulté dans } I_{t_2} \\ 0 & \text{sinon} \end{cases}$$



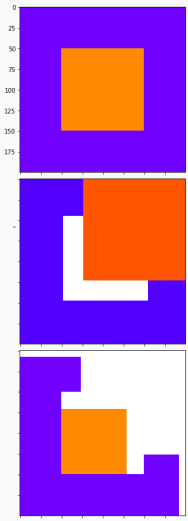
Pour un mouvement particulier :

1. Effectuer un warp de la profondeur
2. Effectuer un warp dans l'autre sens

$$\widetilde{\theta}_t \xrightarrow{T_{t\to t_2}} \widetilde{\theta}_{t\to t_2} \xrightarrow{T_{t_2\to t}} \widetilde{\theta}_{t\to t_2\to t}$$

3. les différences entre $\widetilde{\theta}_t$ et $\widetilde{\theta}_{t\to t_2\to t}$ sont les zones d'occultation.

$$\forall \boldsymbol{p} \in \Omega, occ(\boldsymbol{p}, t_2) = \begin{cases} 1 \text{ si } \widetilde{\theta}_{t\to t_2\to t}(\boldsymbol{p}) = \widetilde{\theta}_t(\boldsymbol{p}) \\ 0 \text{ sinon} \end{cases}$$

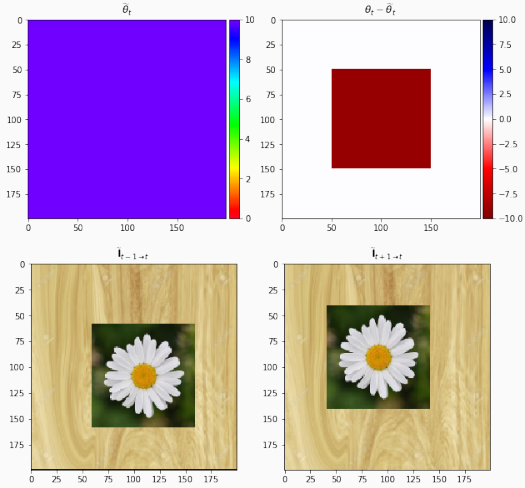


$\widetilde{\theta}_t$

$\widetilde{\theta}_{t\to t_2}$

$\widetilde{\theta}_{t\to t_2\to t}$

Retour à notre exemple



Fonction de coût finale

Erreur photométrique appliquée au différentes images $\widetilde{I}_{t_i \to t}$

$$\mathcal{L}_p = \sum_i \sum_{p \in \Omega} d(I_t, \widetilde{I}_{t_i \to t})(p) \times occ(p, t_i)$$

Coût de lissage

$$\mathcal{L}_s = \lambda \mathcal{L}_{gdiff}(\xi, I_t, \kappa) = \frac{\lambda}{|\Omega|} \sum_{p \in \Omega} \exp\left(-\frac{\|\nabla I_t\|^2}{\kappa^2}\right) \times (\Delta\xi)^2$$

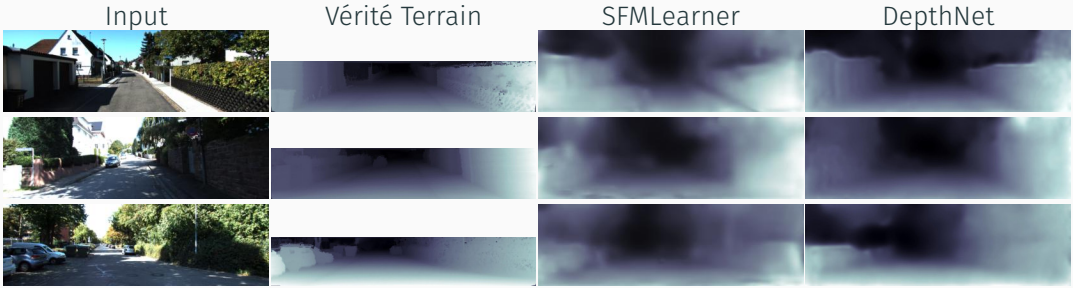
Le tout appliqué sur les différentes échelles de sortie $\widetilde{\theta}^s$ du réseau

$$\mathcal{L} = \sum_s \frac{1}{2^s} \left(\mathcal{L}_p^s + \mathcal{L}_s^s\right)$$

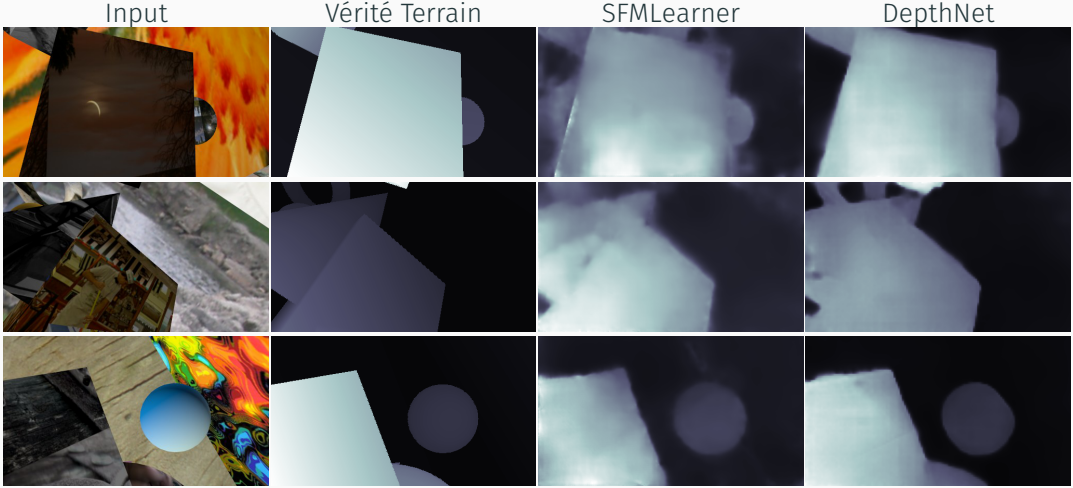
DepthNet Non Supervisé

Résultats

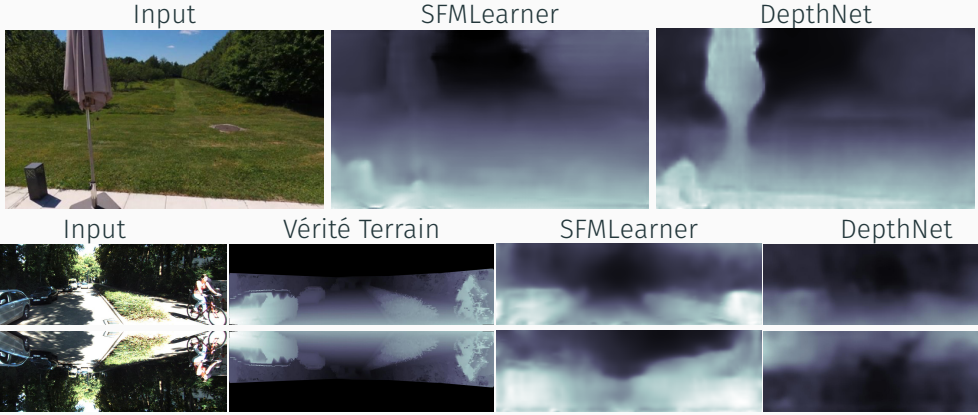
Résultats qualifiquatifs KITTI



Résultats qualitatifs StillBox



Resultats qualitatifs Drone et KITTI renversé



Fonction d'évaluation usuelles

Fonctions d'erreur

Erreur Absolue $|\theta - \widetilde{\theta}|$

Erreur Relative $\dfrac{|\theta - \widetilde{\theta}|}{\theta}$

Erreur Logarithmique $\left| ln\left(\dfrac{\theta}{\widetilde{\theta}}\right)\right|$

L'évaluation est fait en prenant la moyenne de ces fonctions sur un ensemble de validation.

Fonctions de précision

Proportion de valeurs dans un intervalle de confiance autour de θ.

$$P_\delta = P\left(\widetilde{\theta} \in \left[\frac{1}{\delta}\theta, \delta\theta\right]\right)$$

$P_{1.25}$ proportion de points avec moins de **25%** d'erreur

$P_{1.25^2} = P_{1.56}$ moins de **56%** d'erreur

$P_{1.25^3} = P_{1.95}$ moins de **95%** d'erreur

Résultats quantitatifs KITTI

Method	scale factor	Average error	Relative error	Log error	$P_{1.25}$	$P_{1.25^2}$	$P_{1.25^3}$
SFMLearner	GT	3.455	0.185	0.191	0.714	0.899	0.962
SFMLearner	P	4.629	0.296	0.287	0.542	0.817	0.918
DepthNet 1	P	7.872	0.650	0.446	0.356	0.613	0.775
DepthNet 2	P	4.078	0.268	0.222	0.683	**0.883**	**0.944**
DepthNet 3	P	**4.001**	**0.262**	**0.221**	**0.687**	**0.883**	0.943

Rappel

doit être bas

doit être haut

DepthNet 1 : entraînement supervisé sur dataset synthétique (StillBox)

DepthNet 2 : entraînement non supervisé sur KITTI

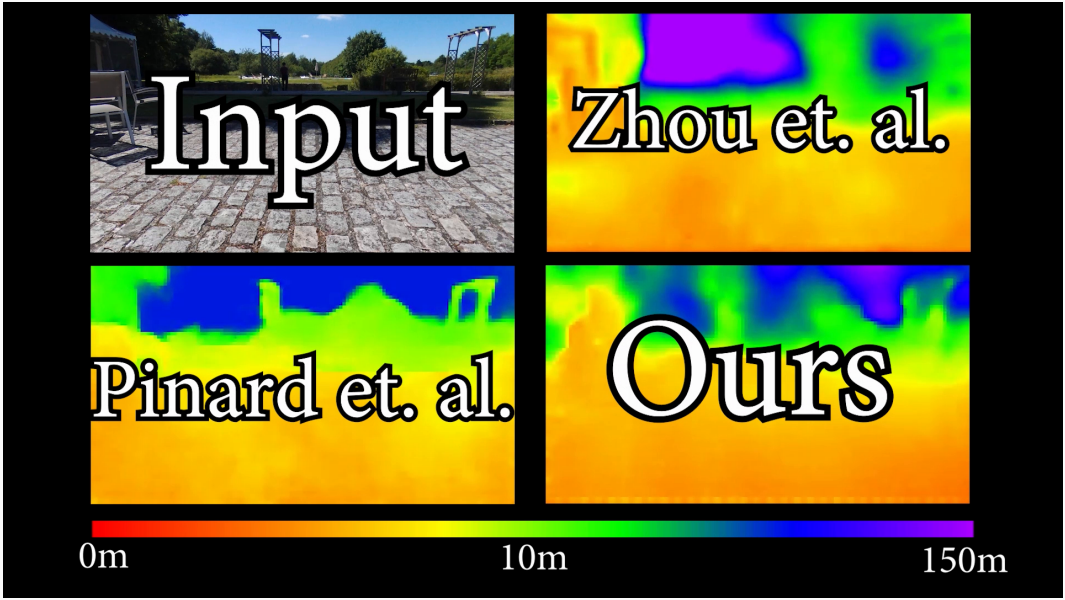
DepthNet 3 : entraînement non supervisé sur KITTI, images stabilisées

Résultats quantitatifs StillBox

Method	scale factor	Detecteur d'occultation	**Average** error	**Relative** error	**Log** error	$P_{1.25}$	$P_{1.25^2}$	$P_{1.25^3}$
DepthNet 1	P	✗	**3.822**	0.224	0.229	0.718	0.856	0.927
SFMLearner	GT	✗	11.43	0.593	0.633	0.479	0.656	0.756
SFMLearner	P	✗	14.32	0.816	0.461	0.302	0.542	0.696
DepthNet 2	P	✗	5.248	0.259	0.232	0.709	0.887	0.928
DepthNet 2	P	✓	5.060	**0.212**	**0.206**	**0.729**	**0.899**	**0.943**

DepthNet 1 : entraînement supervisé

DepthNet 2 : entraînement non supervisé


Input
Zhou et. al.
Pinard et. al.
Ours
0m
10m
150m

1. Introduction et motivations

2. DepthNet Supervisé

3. DepthNet Non Supervisé

4. Utilisation finale de DepthNet
    - Déplacement optimal
    - Multi-inférence
    - Preuve de concept avec streaming vidéo sur Bebop

5. Conclusion

Utilisation finale de DepthNet

Déplacement optimal

De DepthNet à la profondeur $\widetilde{\theta}$

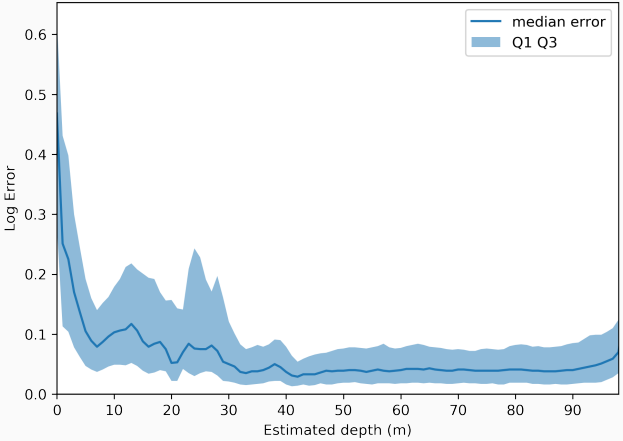
On multiplie par le ratio en d_0 et le déplacement mesuré par le drone $d(t, \delta t)$

$$\widetilde{\theta}(t) = DepthNet(I_t, I_{t-\delta t}) \frac{d(t,\delta t)}{d_0}$$

$$d(t, \delta t) = \left\| \int_{t-\delta t}^{t} V(\tau) d\tau \right\|$$

- La sortie de DepthNet est inversement proportionnelle au déplacement d.
- On peut jouer sur le décalage temporel δ_t pour contrôler la distribution de DepthNet.
- On a 30 images par seconde possibles pour jouer sur le paramètre δt

Déplacement optimal



Toutes les valeurs de sortie de DepthNet ne se valent pas !
On veut que Depthnet soit à valeurs autour de $70m$ pour qu'il fonctionne sous les meilleures conditions possibles

On introduit la fonction β à valeurs dans $[0,1]$ et les paramètres α et θ_{max} pour simplifier cette expression.

$$\begin{array}{rccc} \beta : & \mathbb{R}^{6|\Omega|} & \to & [0,1]^{|\Omega|} \\ & I_1, I_2 & \mapsto & \frac{1}{\theta_{max}} DepthNet(I_1, I_2) \end{array}$$

$$\alpha = \frac{\theta_{max}}{d_0} \qquad \theta_{max} = 100$$

$$\widetilde{\theta}(t) = \alpha\beta(I_t, I_{t-\delta t})d(t, \delta t) = \alpha\beta d(t, \delta t)$$

On veut que β ait une moyenne de $\beta_{mean} = 0.7$

$$d_{opt} = d_0 \frac{\mathbb{E}(\beta)}{\beta_{mean}} = \frac{\mathbb{E}(\widetilde{\theta})}{\alpha\beta_{mean}}$$

Stratégie

1. Estimer $\widetilde{\theta}$ avec une certaine paire d'images $I_t, I_{t-\delta_t}$
2. Calculer la moyenne de β
3. En déduire le déplacement optimal d_{opt}
4. Trouver le nouveau décalage δ'_t avec un déplacement suffisamment proche de d_{opt}

Utilisation finale de DepthNet

Multi-inférence

Certaines scènes n'ont pas qu'un plan
Il n'y a pas un décalage optimal mais **plusieurs**.

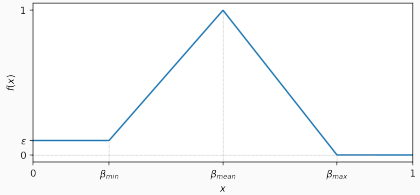
Principe de la multi-inférence

- Trouver n centroïdes $c_0, \cdots, c_{n-1}$ à partir de l'estimation $\widetilde{\theta}$ précédente et un algorithme type K-means [MacQueen, 1967]
- En déduire les décalages optimaux $\delta t_0, \cdots \delta t_{n-1}$
- Fusionner les différentes estimations $\beta_0, \cdots, \beta_{n-1}$

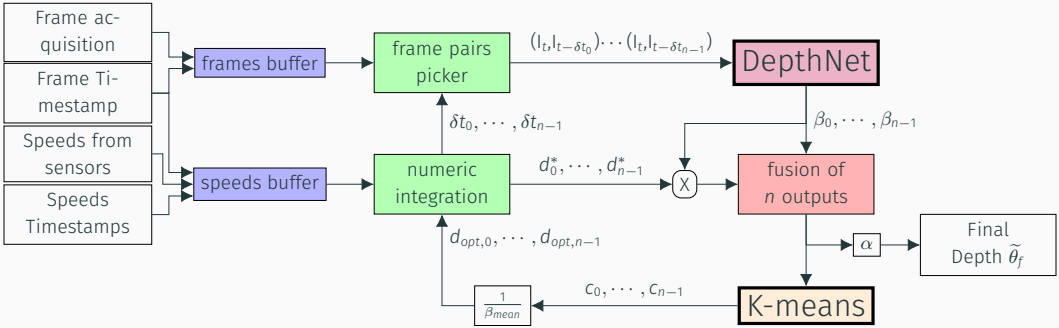
Stratégie

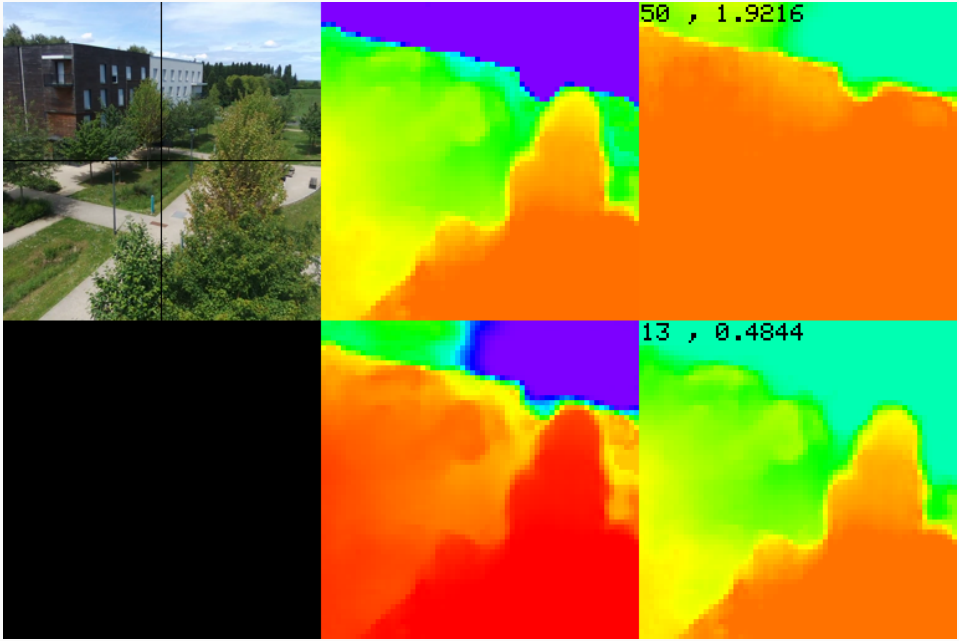
- On compense les β_i avec le déplacement correspondant
- On fait une moyenne pondérée pixel par pixel des $\widetilde{\theta}_i$
- La pondération est forte quand la valeur $\beta_i(p)$ est proche de β_{mean}, faible autrement.

$$\forall p \in \Omega, \widetilde{\theta}_f(p) = \alpha \frac{\sum_i d_i^* w_i(p) \beta_i(p)}{\sum_i w_i(p)} \qquad w_i(p) = f(\beta_i(p))$$



Architecture finale




50 , 1.9216
13 , 0.4844

Utilisation finale de DepthNet

Preuve de concept avec streaming vidéo sur Bebop

Contributions

- Introduction d'un réseau de neurones pour l'estimation de profondeur. Réseau axé sur la robustesse aux scenes inhabituelles. (*Article et présentation orale à* ***UAVg*** *2017,* ***Brevet*** *pour Parrot déposé en 2017*)
- Création de Still Box, un dataset synthétique où la robustesse est essentielle. (*Article et présentation orale à* ***UAVg*** *2017*)
- Adaptation d'un algorithme connu SFMLearner pour apprendre DepthNet sans aucune supervision. (*Workshop à* ***ECCV*** *2018*)
- Introduction d'un algorithme pour utiliser le réseau en conditions réelles et avoir une portée arbitrairement grande. (*Article à* ***ECMR*** *2017*)

Pour la suite ...

- Création d'un dataset de validation de profondeur pour les vidéos de drone.
- Gestion des scènes mobiles.
- Optimisation de l'architecture de DepthNet pour l'embarqué

Merci !

Code source disponible sur github

`perso.ensta.fr/~pinard/`

References i

Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., v.d. Smagt, P., Cremers, D., and Brox, T. (2015).
Flownet : Learning optical flow with convolutional networks.
In *IEEE International Conference on Computer Vision (ICCV)*.

Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018).
Deep ordinal regression network for monocular depth estimation.
In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013).
Vision meets robotics : The kitti dataset.
The International Journal of Robotics Research, 32(11) :1231–1237.

References ii

Horn, B. K. and Schunck, B. G. (1981).
Determining optical flow.
Artificial intelligence, 17(1-3) :185–203.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011).
A large-scale hierarchical multi-view rgb-d object dataset.
In *2011 IEEE international conference on robotics and automation*, pages 1817–1824. IEEE.

Li, Z. and Snavely, N. (2018).
Megadepth : Learning single-view depth prediction from internet photos.
In *Computer Vision and Pattern Recognition (CVPR)*.

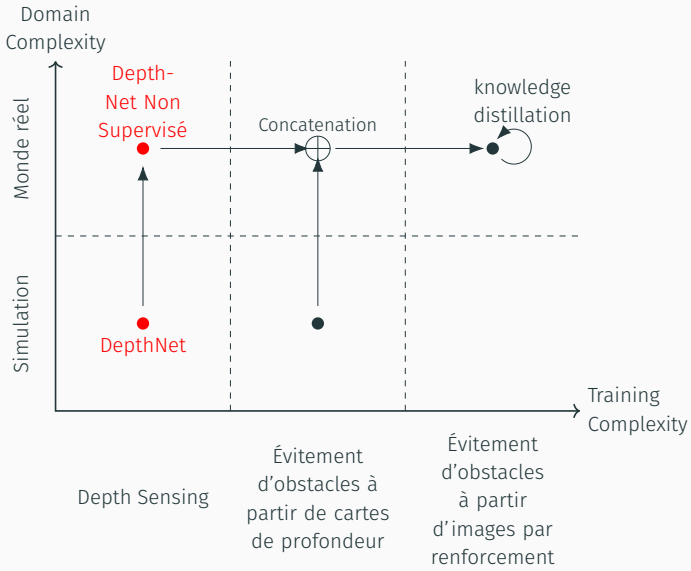
References iii

- MacQueen, J. (1967).
  Some methods for classification and analysis of multivariate observations.
  In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 : Statistics*, pages 281–297, Berkeley, Calif. University of California Press.
- Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017).
  Airsim : High-fidelity visual and physical simulation for autonomous vehicles.
  In *Field and Service Robotics*.
- Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., and Brox, T. (2017).
  Demon : Depth and motion network for learning monocular stereo.
  In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

References iv

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017).
Unsupervised learning of depth and ego-motion from video.
In *CVPR*.

Résumé de la stratégie



Apprentissage profond

Principe général

Fonction qui dépend d'une entrée (par exemple une image I) et de paramètres internes appelés poids (W)

$$output = f(W, I)$$

Apprentissage

Minimiser une fonction de coût $\mathcal{L}$.

Dans le cas d'un apprentissage, on minimise la différence avec une consigne t.

$$W_{final} = \arg\min_W \mathbb{E}_{(I,t)\sim V} \mathcal{L}(f(W, I), t)$$

Pour y arriver on fait un descente de gradient

$$W \leftarrow W - \lambda \nabla_W \mathcal{L}(f(W, I), t)$$

Fonctions de lissage

$$\mathcal{L}_{diff}(\xi) = \iint_{\Omega} exp\left(-\left(\frac{\|\nabla I\|}{\kappa}\right)^2\right) \|\nabla\xi\|^2 dS \tag{1}$$

$$\mathcal{L}_{TV}(\xi) = \iint_{\Omega} exp\left(-\frac{\|\nabla I\|}{\kappa}\right) \|\nabla\xi\| dS \tag{2}$$

$$\mathcal{L}_{gdiff}(\nabla\xi) = \iint_{\Omega} exp\left(-\left(\frac{\|\nabla I\|}{\kappa}\right)^2\right) (\Delta\xi)^2 dS \tag{3}$$

$$\mathcal{L}_{TVV}(\nabla\xi) = \iint_{\Omega} exp\left(-\frac{\|\nabla I\|}{\kappa}\right) |\Delta\xi| dS \tag{4}$$

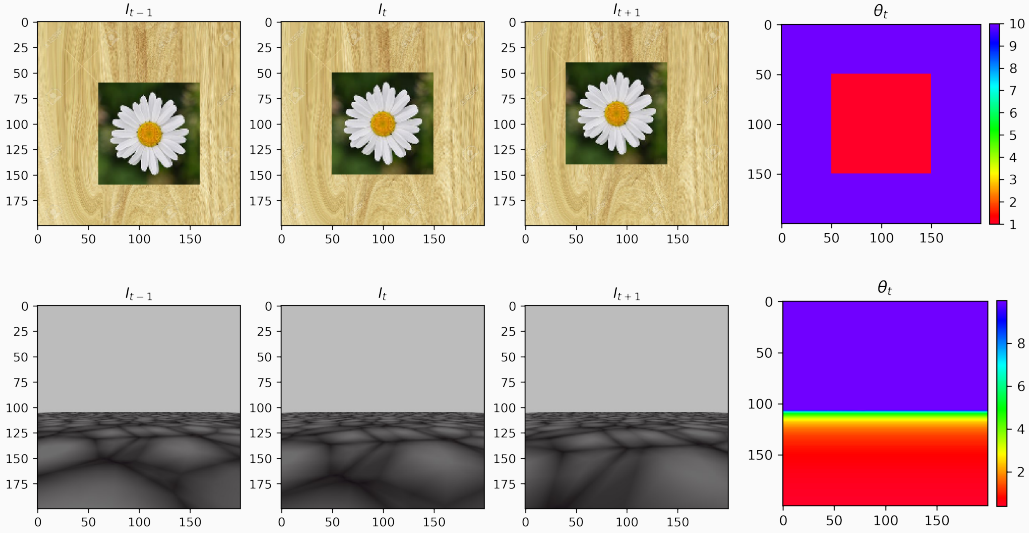
Optimisation en deux temps

$$\mathcal{L}(\widetilde{\theta}, \alpha) = \mathcal{L}_p(\widetilde{\theta}) + \lambda \mathcal{L}_s(\alpha) + \gamma d\left(\alpha, \xi = \frac{1}{\widetilde{\theta}}\right)$$

Stratégie d'optimisation "robuste"

1. estimer $\widetilde{\theta} = f(W)$ avec les paramètres optimisables W.
2. optimiser $\mathcal{L}(\widetilde{\theta}, \alpha)$ avec $\widetilde{\theta}$ fixé, trouver $\widetilde{\alpha} = \arg\min_{\alpha} \mathcal{L}(\widetilde{\theta}, \alpha)$.
3. Caculer $\nabla_W \mathcal{L}(\widetilde{\theta}, \widetilde{\alpha})$ et mettre à jour W.

Scènes synthétiques



Validation sur scènes synthétiques

$$LE(\theta, \widetilde{\theta}) = \left\| \log(\theta) - \log(\widetilde{\theta}) \right\| \qquad LE_0 = LE(\widetilde{\theta}_0^{\mathcal{S}}, \theta^{\mathcal{S}})$$

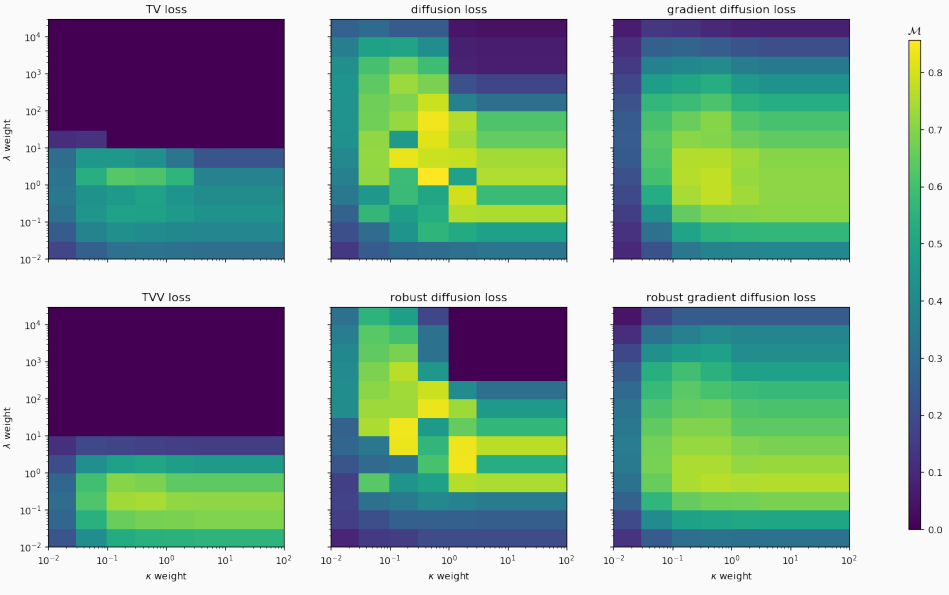
$$\mathcal{M}_{\mathcal{S}}(\widetilde{\theta}^{\mathcal{S}}) = \begin{cases} 0 & \text{if } LE(\theta^{\mathcal{S}}, \widetilde{\theta}^{\mathcal{S}}) > LE_0 \\ 1 - \frac{LE(\theta^{\mathcal{S}}, \widetilde{\theta}^{\mathcal{S}})}{LE_0} & \text{otherwise} \end{cases}$$

$$\mathcal{M} = \sqrt[|\mathbb{S}|]{\prod_{\mathcal{S} \in \mathbb{S}} \mathcal{M}_{\mathcal{S}}}$$

Rechercher d'hyperparamètres sur scènes synthétiques

- Maximiser $\mathcal{M}$
- Minimiser $\Delta\mathcal{M}$: le maximum de $\mathcal{M}$ doit être le plus étendu possible

Resultats


TV loss
diffusion loss
gradient diffusion loss
TVV loss
robust diffusion loss
robust gradient diffusion loss
λ weight
κ weight
$\mathcal{M}$
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8