Paramétrisation de fonction d'étalement du point par apprentissage génératif sur des mesures expérimentales

Théo SANTOS[1,2] Ferréol SOULEZ[1] Thomas BODIN[3] Olivier FLASSEUR[1]

[1]Univ Lyon 1, ENS de Lyon, CNRS, Centre de Recherche Astrophysique de Lyon, UMR 5574, F-69230, Saint-Genis-Laval, France

[2]Univ Lyon 1, ENS de Lyon, CNRS, LGL-TPE, UMR 5276, F-69622, Villeurbanne, France

[3]Instituto de Ciencias del Mar (ICM) - CSIC, Pg. Marítim de la Barceloneta, 37, Ciutat Vella, 08003 Barcelona

Résumé – Nous proposons une paramétrisation de la fonction d'étalement du point (PSF), qui caractérise la déformation du signal lumineux par un système optique. Cette paramétrisation est apprise par un réseau de neurones génératif s'entraînant sur des acquisitions réelles de PSFs. Nous présentons la méthode d'entraînement du générateur, notamment les spécificités pour l'utilisation de données expérimentales, puis exposons les résultats de génération, montrant une paramétrisation efficace et réaliste de PSFs.

Abstract – We propose a parametrization of the Point-Spread Function (PSF), which characterises the deformation of the light by an optical system. This parametrization is learned by a generative neural network trained on real PSFs. We present the method used to train the generator, in particular concerning the specifities for using experimental data. We also expose the generation results, showing an efficient and realistic parametrization of PSFs.

1 Introduction

Lors d'une observation avec un instrument optique, le signal lumineux enregistré par le capteur arrive déformé par le milieu de propagation et le dispositif optique. Ces altérations peuvent être modélisées mathématiquement comme la convolution du signal original par la réponse impulsionnelle $\boldsymbol{h}$ (*Point-Spread Function*, ou PSF) décrivant l'ensemble des perturbation :

$$\boldsymbol{d} = \boldsymbol{h} \star \boldsymbol{x} + \boldsymbol{\epsilon}\,, \tag{1}$$

où $\boldsymbol{d}$ est l'image mesurée, $\star$ l'opération de convolution, $\boldsymbol{x}$ l'image nette et $\boldsymbol{\epsilon}$ un terme d'erreur.

De nombreux traitements des données (photométrie, détection, déconvolution,...) requièrent la connaissance de cette PSF. Celle-ci pouvant varier d'une acquisition à l'autre, les modèles théoriques fixes ne sont pas adaptés et la PSF doit être calibrée directement sur les observations scientifiques ou des observations dédiées (*e.g.*étoile de calibration) rapprochées dans le temps. Sur ces mesures généralement très bruitées, on ajuste classiquement un modèle paramétrique de PSF avec peu de degrés de liberté. Cette paramétrisation peut être justifiée par l'optique [8, 9, 3] ou construite par une projection sur un espace de petite dimension [7, 2]. Néanmoins, ce type de paramétrisation peut être trop simple, ne modélisant pas tous les phénomènes possibles. Utiliser une modélisation physique plus complète (*e.g.* prenant en compte l'effet de l'optique adaptative, la propagation de la lumière, la réponse du détecteur) est en général extrêmement difficile.

Nous proposons ici d'utiliser un réseau de neurones génératif pour paramétriser les PSFs. L'originalité de notre travail est que cette paramétrisation est apprise non pas sur des PSFs théoriques simulées mais sur des observations, dans toute leur complexité. Nous présentons la méthode en Section 2, avant de détailler ses spécificités dues à l'utilisation de données expérimentales en Section 3, puis nous montrons les résultats obtenus pour cette paramétrisation en Section 4.

TABLE 1 : Architecture du générateur de PSFs.

Couches	Taille de sortie	*Stride*	*Padding*
ConvTranspose 4×4	$4 \times 4 \times 2048$	1	0
ConvTranspose 4×4	$8 \times 8 \times 1024$	2	1
ConvTranspose 4×4	$16 \times 16 \times 512$	2	1
ConvTranspose 4×4	$32 \times 32 \times 256$	2	1
ConvTranspose 4×4	$64 \times 64 \times 128$	2	1
ConvTranspose 4×4	$128 \times 128 \times 1$	2	1

2 Méthode

2.1 Paramétrisation apprise par un générateur

Pour obtenir une paramétrisation de PSFs réalistes, nous proposons d'utiliser un réseau de neurones génératif $\mathcal{G}$, pouvant générer de nouvelles données statistiquement similaires à des données d'entraînement (par exemple des images). Le réseau apprend à associer une distribution simple (classiquement une distribution normale centrée réduite) avec une distribution complexe dont les données d'entraînement sont des échantillons. La distribution simple est définie sur l'espace d'entrée (appelé *espace latent*) du générateur, et la distribution complexe sur l'espace de sortie. De cette manière, il est possible d'échantillonner la distribution complexe en échantillonnant des vecteurs latents vus à travers $\mathcal{G}$. Dans notre cas, la distribution complexe à échantillonner est celle des images de PSFs réalistes. On peut alors utiliser le générateur comme une paramétrisation de PSFs, où les paramètres d'une PSFs sont le vecteur latent $\boldsymbol{z}$ en entrée du générateur.

Les réseaux de neurones génératifs présentent plusieurs avantages utiles pour la paramétrisation de PSFs :

- Dimensionnalité : pour un espace latent de petite dimen-

sion, les PSFs sont définies par peu de paramètres.

- Complexité algorithmique : une fois entraîné, un réseau de neurones a un temps d'inférence bref, de l'ordre de la milliseconde.
- Relation entre paramètres et images : les réseaux de neurones génératifs montrent une relation "régulière" entre paramètres et images, dans le sens où des vecteurs latents proches résultent en des images proches. Cette propriété est particulièrement intéressante pour une paramétrisation, où une relation erratique entre les paramètres et les PSFs compliquerait la résolution de problèmes.
- La proximité à un *a priori* physique : le générateur est entraîné pour échantillonner la distribution donnée en entraînement ; par construction, la paramétrisation est contrainte à cette distribution. Dans notre cas, en entraînant le générateur sur des images de PSFs expérimentales, on s'assure qu'il ne produise que des images de PSFs réalistes.

Pour entraîner le générateur, nous utilisons la méthode des *Generative Adversarial Networks* (GAN) [4]. L'entraînement consiste à confronter le générateur $\mathcal{G}$ à un autre réseau, le discriminateur $\mathcal{D}$, qui apprend à différencier les vraies PSFs (*i.e.* tirées du jeu d'entraînement) des fausses (*i.e.* produites par le générateur). Les deux réseaux en compétition s'améliorent chacun en s'appuyant sur l'autre, de manière progressive et alternée : le générateur apprend à tromper le discriminateur quand ce dernier apprend à détecter les fausses images ; jusqu'à ce que le générateur produise des images satisfaisantes. La version de GAN que nous utilisons est le WGAN [1] avec pénalité de gradient [5]. L'architecture du générateur utilisé est présentée au Tableau 1. Chaque couche de convolution transposée est suivie d'une normalisation par batch [6] et d'une fonction d'activation ReLU, sauf la couche de sortie qui est dotée d'une simple fonction d'activation sigmoïde.

2.2 Défis spécifiques

Afin d'avoir une paramétrisation réaliste, nous entraînons le générateur sur des mesures expérimentales de PSFs. Cette particularité amène avec elle des difficultés :

- Dépendance au flux incident : les objets observés pour imager les PSFs peuvent avoir des flux très différents. Les PSFs du jeu d'entraînement on donc un niveau moyen très hétérogène alors que nous voulons générer des PSFs normalisée en flux.
- Dynamique élevée : chaque PSF est très contrastée, avec la partie centrale de l'image dominant le signal. Les structures de faible intensité, bien que porteuses d'information, risquent de ne pas être bien capturées par le générateur durant l'entraînement.
- Bruit : les images d'entraînement sont bruitées car elles proviennent de mesures expérimentales ; or l'objectif souhaité du générateur est de proposer des PSFs non bruitées.

Pour répondre à ces défis, nous proposons un pré-traitement spécifique et une modification de l'architecture d'entraînement du GAN.

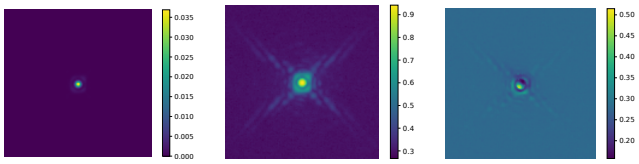


FIGURE 1 : Illustration de l'effet de l'égalisation sur une PSF. Gauche : PSF d'origine normalisée. Milieu : même PSF après log-égalisation $\mathcal{E}_{\text{l}}$. Droite : même PSF après blanchiment $\mathcal{E}_{\text{b}}$.

3 Apprentissage sur des données expérimentales

3.1 Normalisation

Pour s'affranchir du paramètre de flux incident, nous procédons pendant le pré-traitement à une étape de normalisation. Pour que toutes les PSFs aient la même somme égale à 1, nous utilisons l'opérateur de normalisation, pour une PSF $\boldsymbol{h}$:

$$\mathcal{N}_{\Sigma}(\boldsymbol{h}) = \frac{\boldsymbol{h}}{\sum_i h_i}\,. \tag{2}$$

Pour s'assurer que les PSFs générées somment à 1, nous appliquons également la normalisation en sortie du générateur pendant l'entraînement.

3.2 Egalisation

Les PSFs sont très contrastées et il est difficile pour le générateur d'apprendre à reproduire les parties faibles du signal, le risque étant qu'il ne se concentre que sur la partie la plus brillante en ignorant le reste. Pour éviter ce risque, nous proposons de relever les parties faibles des images de PSFs d'entraînement, avec un opérateur d'égalisation, noté $\mathcal{E}$. Nous définissons deux opérateurs d'égalisation, illustrés en Figure 1 sur une PSF d'exemple.

Le premier, l'égalisation logarithmique, est défini, pour une PSF $\boldsymbol{h}$, par :

$$\mathcal{E}_{\text{l}}(\boldsymbol{h}) = \frac{\log(a\boldsymbol{h}+b)}{\log(a+b)} + c\,, \tag{3}$$

où a, b et c sont des constantes ne dépendant pas de la PSF.

Le second, le blanchiment, correspond à une normalisation standard par pixel sur l'ensemble d'entraînement. Pour une PSF $\boldsymbol{h}$, cette égalisation se définit au pixel i :

$$\mathcal{E}_{\text{b}}(\boldsymbol{h}) = \frac{\frac{h_i-\mu_i}{\sigma_i} - m}{M-m}\,, \tag{4}$$

où μ_i et σ_i sont les moyennes et écart-types du pixel i calculées sur l'ensemble d'entraînement, M et m sont des constantes ne dépendant pas de la PSF.

Lors du pré-traitement, après normalisation, les PSFs d'entraînements sont égalisées avec une des deux égalisations. Le générateur apprend donc à générer des PSFs égalisées. Une fois l'entraînement terminé, pour obtenir des PSFs exploitables, il faut donc ajouter un opérateur de dé-égalisation en sortie du générateur. C'est pourquoi nous avons défini les opérateurs $\mathcal{E}_{\text{b}}$ et $\mathcal{E}_{\text{l}}$ de sorte qu'ils soient inversibles en choisissant adéquatement les valeurs de a, b et c.

3.3 Ajout de bruit simulé

Il est important que le générateur propose des PSFs non bruitées même s'il a été entraîné à partir des images de PSFs d'entraînement issues d'observations, donc bruitées. Même si la structure convolutive du générateur limite intrinsèquement la production d'artefacts de bruit [10], cette propriété n'est néanmoins pas suffisante pour éviter totalement la production d'images bruitées.

Lors de l'entraînement du GAN, le discriminateur voit les PSFs expérimentales bruitées, conduisant le générateur à produire des PSFs également bruitées pour le tromper. Pour remédier à ce problème, nous proposons d'ajouter, en sortie du générateur, du bruit identique à celui présent dans les PSFs d'entraînement. Ainsi, le générateur peut produire des images non bruitées, l'ajout de bruit simulé assurant que seules des images bruitées soient présentées au discriminateur.

Pour que les images générées puis bruités soient indiscernables des PSFs d'entrainement, il est indispensable que le bruit ajouté suive la même statistique que le bruit réellement présent lors des observations. Nous modélisons ce bruit en deux composantes :

$$\boldsymbol{\epsilon}_{\text{PSF}} = \boldsymbol{\epsilon}_{\text{lecture}} + \boldsymbol{\epsilon}_{\text{photon}} \,, \tag{5}$$

Le premier terme est le bruit de lecture ; il est indépendant par pixel et gaussien. Son écart-type varie par pixel et par observation (hétéroscédasticité) mais est connu pour chaque acquisition via une carte la variance de bruit. Le second terme est le bruit de photon, indépendant par pixel. Il suit une une loi de Poisson dont le paramètre est le produit du flux attendu par un facteur connu variant pour chaque pixel et chaque observation. Pour chaque PSF du jeu d'entraînement, on approxime alors le bruit total par un bruit gaussien dont la variance dépend de la PSF et de paramètres $\boldsymbol{\theta}$: gain du detecteur, variance du bruit de lecture pour chaque pixel, nombre de trames.

Ainsi, lors de l'ajout de bruit pendant l'entraînement, pour chaque PSF $\boldsymbol{h}$ générée, on tire aléatoirement une carte de variance du bruit d'après $\boldsymbol{h}$ et les statistiques des paramètres $\boldsymbol{\theta}$. Puis, à partir de cette carte, on tire une réalisation du bruit que l'on ajoute à l'image de PSF générée avant de la présenter au discriminateur.

3.4 Synthèse sur l'algorithme proposé

Pour répondre aux trois défis évoqués, nous combinons les trois méthodes décrites. Le pré-traitement consiste en une normalisation suivie d'une égalisation des PSFs. Puis, l'entraînement suit le schéma en Figure 2.

4 Résultats

Dans cette section nous cherchons à paramétriser les PSFs de l'imageur SPHERE-IRDIS, dédié à la recherche d'exoplanètes au *Very Large Telescope*.

4.1 Ensemble d'entraînement

Nous utilisons comme jeu d'entraînement ≈ 1600 PSFs acquises entre 2016 et 2017 et aux longueurs d'onde 1589 nm et 1667 nm. Quelques exemples sont présentés en Figure 3.

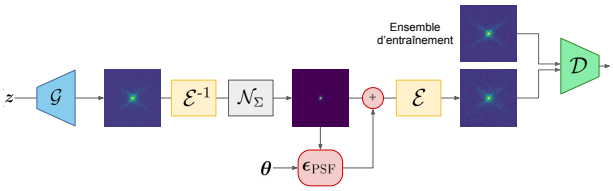
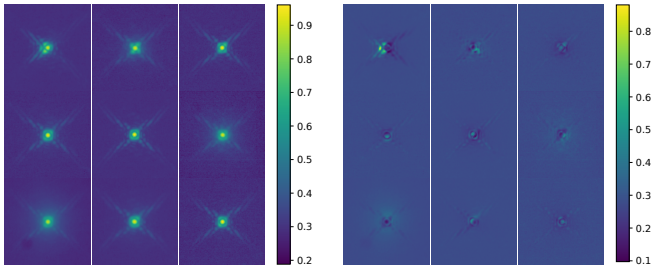


FIGURE 2 : Processus d'entraînement du générateur de PSFs. L'image générée $\mathcal{G}(\boldsymbol{z})$ est dé-égalisée avec $\mathcal{E}^{-1}$, puis normalisée avec $\mathcal{N}_\Sigma$ et bruitée. Les statistiques du bruit sont données par la PSF et les paramètres $\boldsymbol{\theta}$ tirés au hasard. L'image bruitée est égalisée avec $\mathcal{E}$ et donnée au discriminateur.



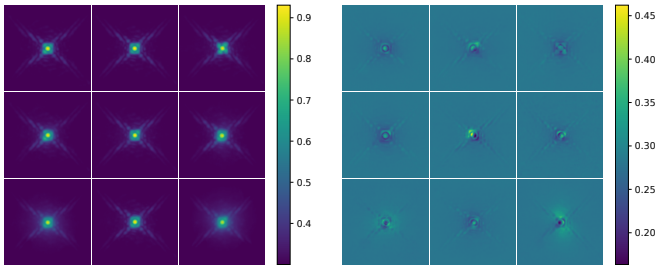
(a) *(b)*

FIGURE 3 : Exemples de PSFs égalisées de l'ensemble d'entraînement. (a) PSFs log-égalisées. (b) PSFs blanchies.

4.2 PSFs générées

Nous entraînons différents générateurs avec des PSFs log-égalisées ou blanchies. Dans tous les exemples présentés, les générateurs ont un espace latent à 20 dimensions. Après entraînement, les générateurs sont capables de produire des images de taille 128×128 pixels de PSFs réalistes. En effet, comme on peut le voir dans la Figure 4, les PSFs générées imitent la structure globale et les différentes aberrations possibles qui étaient présentes dans l'ensemble d'entraînement. Ces résultats sont visibles pour les PSFs log-égalisées et pour les PSFs blanchies.

De plus, la relation entre les paramètres et les images semble bien régulière, dans le sens défini plus haut. En effet, on représente en Figure 5 des PSFs interpolées sur l'espace latent, c'est-à-dire des images $\mathcal{G}\left((1-\lambda)\boldsymbol{z}_1 + \lambda \boldsymbol{z}_2\right)$ pour $\boldsymbol{z}_1$ et $\boldsymbol{z}_2$ des vecteurs latents tirés aléatoirement et λ un scalaire variant entre 0 et 1. On peut voir que les PSFs évoluent de manière progressive, ce qui indique une bonne relation entre image et



(a) *(b)*

FIGURE 4 : Exemple de PSFs générées aléatoirement pour deux configurations différentes. (a) Le générateur a été entraîné sur des PSFs log-égalisées. (b) Le générateur a été entraîné sur des PSFs blanchies.

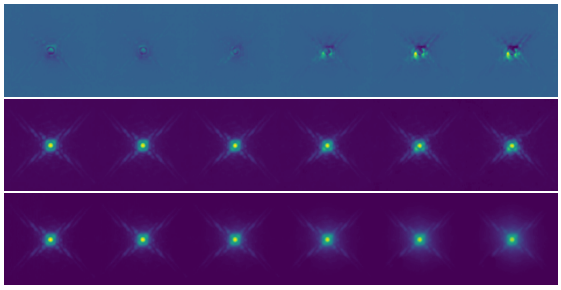


FIGURE 5 : Interpolations entre deux PSFs générées aléatoirement. Première interpolation : le générateur a été entraîné sur des PSFs blanchies. Deuxième interpolation : même interpolation après dé-égalisation et log-égalisation ($\mathcal{E}_b^{-1} \circ \mathcal{E}_l$). Troisième interpolation : le générateur a été entraîné sur des PSFs log-égalisées.

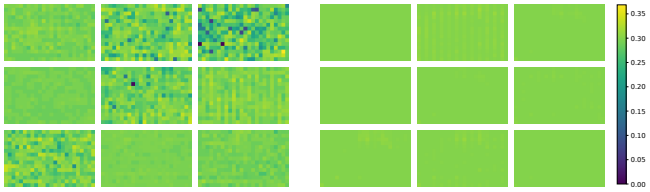


FIGURE 6 : Comparaison de zones avec peu de signal utile de PSFs, entre les images d'entraînements et les images générées. Ces fenêtres sont prises pour des pixels des lignes 1 à 15 et des colonnes 19 à 44. Les images sont représentées en échelle logarithmique. Le générateur a été entraîné avec des images log-égalisées.

paramètres.

La méthode proposée réussit à prévenir la génération du bruits dans les PSFs. En effet, en Figure 6 sont affichées des parties des images sur une zone où le flux de la PSF est très faible. On peut constater que le bruit est bien plus important sur les PSFs d'entraînement que sur les PSFs générées. Ceci peut aussi se voir en Figure 7 où sont comparés les spectres de puissance moyens des PSFs d'entraînement et des PSFs générées. Ces spectres sont similaires, jusqu'à la fréquence de coupure traduisant la capacité du générateur à reproduire la structure fréquentielle des PSFs sans reproduire celle du bruit. Au-delà de la fréquence de coupure optique, la densité spectrale de puissance des PSFs générées est d'un ordre de grandeur plus faible conformément à ce qu'on attend optiquement.

5 Conclusion

En entraînant un réseau de neurones génératif sur des PSFs de SPHERE-IRDIS, nous proposons une paramétrisation de PSFs basée sur de l'apprentissage. Elle est rapide à exécuter, de petite dimensionnalité, avec une relation régulière entre les paramètres et les PSFs, et contrainte à une distribution de PSFs réalistes. Pour l'apprentissage sur des données réelles - bruitées et très contrastées - nous avons employé plusieurs méthodes : normalisation, égalisation et ajout de bruit. Cela nous permet d'obtenir un générateur reproduisant bien les statistiques visuelles et fréquentielles de vraies PSFs, tout en évitant la production d'artefacts de bruit.

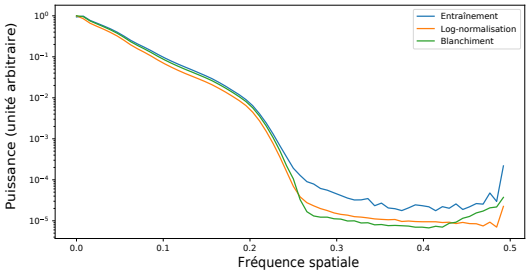


FIGURE 7 : Spectres de puissance moyens des PSFs ; comparaison entre les images d'entraînement (courbe bleue) et les images générées (courbe orange pour le générateur entraîné avec des PSFs log-normalisées, courbe verte pour le générateur entraîné avec des PSFs blanchies).

Un tel générateur ne nécessite que quelques centaines de PSFs pour son entraînement ; il est ainsi envisageable de généraliser cette paramétrisation pour d'autres télescopes ou instruments optiques. Ainsi, cette paramétrisation pourrait être utilisée dans des problèmes nécessitant une bonne estimation de la PSF (*e.g.* la déconvolution aveugle) qui pourra bénéficier de cette relation régulière, rapide et à peu de dimensions.

Références

[1] M ARJOVSKY *et al.* : Wasserstein generative adversarial networks. *In Proceedings of the 34th ICML*, volume 70, 2017.

[2] V. DEBARNOT, P. ESCANDE, T. MANGEAT et P. WEISS : Learning low-dimensional models of microscopes. *IEEE Trans. on Computational Imaging*, 7:178–190, 2020.

[3] RJL FÉTICK *et al.* : Physics-based model of the adaptive-optics-corrected point-spread function-applications to the SPHERE/ZIMPOL and muse instruments. *A&A*, 628, 2019.

[4] I GOODFELLOW *et al.* : Generative adversarial nets. *In NeurIPS*, volume 27, 2014.

[5] I GULRAJANI *et al.* : Improved training of wasserstein gans. *In NeurIPS*, volume 30, 2017.

[6] S IOFFE et C SZEGEDY : Batch normalization : Accelerating deep network training by reducing internal covariate shift. *In Proc. of the 32nd ICML*, volume 37, 2015.

[7] T KENIG, Z KAM et A FEUER : Blind image deconvolution using machine learning for three-dimensional microscopy. *IEEE Trans. PAMI*, 32(12):2191–2204, 2010.

[8] J MARKHAM et JA CONCHELLO : Parametric blind deconvolution : a robust method for the simultaneous estimation of image and blur. *JOSA A*, 16, 1999.

[9] F SOULEZ *et al.* : Blind deconvolution of 3D data in wide field fluorescence microscopy. *In 2012 9th IEEE ISBI*, pages 1735–1738, 2012.

[10] D ULYANOV *et al.* : Deep image prior. *In Proceedings of the IEEE Conference on CVPR*, 2018.