



HAL
open science

Synthetic learning for neural image restoration methods

Raphaël Achddou

► **To cite this version:**

Raphaël Achddou. Synthetic learning for neural image restoration methods. Image Processing [eess.IV]. Institut Polytechnique de Paris, 2023. English. NNT : 2023IPPAT006 . tel-04164873

HAL Id: tel-04164873

<https://theses.hal.science/tel-04164873>

Submitted on 18 Jul 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNT : 2023IPPAT006

Thèse de doctorat



Synthetic learning for neural image restoration methods

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)

Spécialité de doctorat : Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 15 mars 2023, par

RAPHAËL ACHDDOU

Composition du Jury :

Florence Tupin Professeure, Télécom Paris (LTCl)	Présidente/Examinatrice
Jean-François Aujol Professeur, Université de Bordeaux (IMB)	Rapporteur
Bruno Galerne Professeur, Université d'Orléans (Institut Denis Poisson)	Rapporteur
Sabine Süsstrunk Professeure, EPFL (IVRL)	Examinatrice
Jesus Angulo Lopez Directeur de recherche, Mines de Paris	Examineur
Sira Ferradans Directrice de Recherche, DxO Marks	Examinatrice
Yann Gousseau Professeur, Télécom Paris (LTCl)	Directeur de thèse
Saïd Ladjal Maître de conférences, Télécom Paris (LTCl)	Co-directeur de thèse
Jean-Michel Morel Professeur émérite, ENS Paris-Saclay (Centre Borelli)	Invité

Résumé en français

La photographie est devenue un élément omniprésent dans nos vies. En outre, les attentes en termes de qualité d'image des consommateurs augmentent alors que la taille des appareils d'imagerie diminue. Dans ce contexte, l'amélioration des algorithmes de traitement d'images est devenue un enjeu essentiel pour les constructeurs d'appareils photographiques et téléphones portables.

Dans ce manuscrit, nous nous intéressons particulièrement aux tâches de restauration d'images. L'objectif est de produire une image propre à partir d'une ou plusieurs observations bruitées de la même scène. Les méthodes classiques de restauration d'images sont le plus souvent basées sur une hypothèse ou *a priori* sur la distribution des images naturelles. Ces hypothèses, souvent trop contraignantes, ne permettent pas de restaurer correctement des structures complexes dans les images. Récemment, les méthodes d'apprentissage profond ont connu une croissance spectaculaire, surpassant l'état de l'art pour la grande majorité des tests traditionnels. Il n'est plus ici question d'*a priori*, ces méthodes essayant de minimiser directement une erreur de reconstruction sur des bases de données volumineuses.

Bien que ces méthodes produisent des résultats impressionnants, elles présentent un certain nombre d'inconvénients. Tout d'abord, elles sont difficiles à interpréter en raison de leur fonctionnement en "boîte noire". De plus, contrairement aux méthodes classiques, elles se généralisent assez mal à des modalités d'acquisition ou de distorsion absentes de la base de données d'entraînement. Enfin, elles nécessitent de grandes bases de données, qu'il est parfois difficile d'acquérir, lorsque les problèmes traités sont très spécifiques.

Nous proposons d'attaquer ces différents problèmes en remplaçant l'acquisition des données par un algorithme simple de génération d'images, basé sur le modèle d'images feuilles mortes. Bien que ce modèle soit très simple, les images générées ont des propriétés statistiques proches de celles des images naturelles (distribution du gradient, forme du spectre en puissance, non gaussianité...) et de nombreuses propriétés d'invariance (échelle, translation, rotation, contraste...). L'entraînement d'un réseau de restauration avec ce type d'images nous permet d'identifier les propriétés importantes des images pour le succès des réseaux de restauration. Ainsi, cette approche fait le pont entre les méthodes avec *a priori* et les méthodes par apprentissage, car on impose certaines propriétés dans les images d'entraînement. De plus, cette méthode permet aussi de s'affranchir de l'acquisition des données, qui peut être fastidieuse.

Après avoir présenté le modèle feuilles mortes en détail, nous montrons qu'il est possible d'obtenir des performances de restauration très satisfaisantes en entraînant les réseaux de restauration sur ce type d'images. Nous nous intéressons dans un premier temps à des tâches relativement simples, comme le débruitage gaussien ou la super-résolution. Cette étude nous permet aussi d'identifier des propriétés importantes pour le bon fonctionnement des réseaux de restauration comme l'importance des couleurs, ou la distribution des formes dans les images. Après quelques adaptations du modèle feuilles mortes, l'apprentissage synthétique permet également de s'attaquer à des problèmes concrets difficiles, tels que le débruitage d'images RAW. Ces adaptations concernent principalement la formation d'images RAW synthétiques, et une modélisation fine du bruit RAW, bien plus complexe que son approximation gaussienne. Après avoir présenté ces résultats, nous proposons une étude statistique de la distribution des couleurs des images naturelles, permettant d'élaborer un modèle paramétrique réaliste d'échantillonnage des couleurs pour notre algorithme de génération. Cette étude permet à notre algorithme de génération de ne pas dépendre de données existantes,

et se traduit en des performances de restauration d'images très satisfaisantes. Enfin, nous présentons une nouvelle fonction de perte perceptuelle basée sur les protocoles d'évaluation des caméras pour la préservation des textures, faisant intervenir les images de feuilles mortes. L'entraînement réalisé avec cette fonction montre que l'on peut optimiser conjointement l'évaluation des caméras, tout en gardant des performances identiques sur les images naturelles.

Remerciements

Mes premiers remerciements vont aux membres du jury. Jean-Francois Aujol, qui a accepté de rapporter cette thèse, m’a fourni de nombreux commentaires et suggestions, notamment lors de la soutenance de mi-parcours. Pour son vif intérêt dans mes travaux, je lui adresse mes plus sincères remerciements. Je tiens également à remercier Bruno Galerne pour son rapport détaillé ainsi que pour ses nombreuses suggestions, qui ont permis de grandement améliorer le manuscrit. Ses conseils et encouragements lors de ma venue à Orléans, m’ont été très précieux. Sira Ferradans, Jesus Angulo Lopez, et Jean-Michel Morel m’ont fait l’honneur de faire partie du jury de cette thèse, et je les en remercie chaleureusement. En particulier je tiens à remercier Jean-Michel Morel pour ses cours de Master qui ont piqué ma curiosité pour la restauration d’image et pour avoir suivi le déroulement de ma thèse, notamment lors de la soutenance de mi-parcours. I would also like to warmly thank Sabine Süsstrunk, for being part of this jury, and for expressing her interest in my works. I am sure our future collaboration at EPFL will be very fruitful.

A mes directeurs de thèse, un immense merci. Yann, tu as été à l’origine de ce projet et beaucoup des idées de cette thèse te sont dues. Sans ton expertise et tes encouragements, cette thèse n’aurait probablement pas vu le jour. Saïd, ta rigueur et ton aide pour l’aspect expérimental ont été capitales, et m’ont débloqué à des moments cruciaux. Merci aussi pour les multiples idées très originales que tu as apportées lors de cette thèse. Enfin, merci aussi d’avoir su m’encadrer à deux, avec beaucoup de complémentarité. Vous avez fait preuve de beaucoup de bienveillance tout au long de ce doctorat et avez toujours été très disponibles même quand la situation sanitaire nous empêchait pendant presque un an de nous voir en chair et en os. J’ai hâte de poursuivre cette belle collaboration avec vous.

Je tiens évidemment à remercier infiniment l’équipe IMAGES dans laquelle j’ai eu le privilège de passer les 3 dernières années. J’y ai rencontré des personnes exceptionnelles tant sur le plan humain que sur le plan académique. Malgré la pandémie, et les multiples confinements, il y a toujours eu une superbe ambiance de travail. Merci à l’équipe de permanents composées de Florence, Isabelle, Henri, Michel, Pietro, Alasdair, Christophe, Kiwon, Amal, Elsa et Loïc, pour votre dynamisme et votre sens du collectif, qui sont un vrai moteur pour l’équipe. Je remercie aussi mes amis doctorants pour ces échanges passionnants à Télécom ou ailleurs, et pour votre soutien inconditionnel. Merci en particulier aux organisateurs successifs du Deep Learning Working Group (ou GRAP): Nicolas Go., Emanuele, Antoine, Arthur, Inès et Gwilherm en particulier, merci de m’avoir remplacé. Mes plus vifs remerciements à Nicolas C. , Pierrick, Rebeca, Erwan, Antoine, Giammarco, Zoé, Mateus, Florence, Rassim, Robin L., Xu, Elie, Camille, Robin K., Nicolas Ga., Carlo Alberto, Benoît, Raphaël R, Emma et Emile. Enfin, un grand merci à Mathis, qui me supporte avec une infinie gentillesse depuis

plus de 5 ans, que ce soit dans le bureau 5B45 ou pendant nos multiples projets de masters. Je vous souhaite à tous une bonne fin de thèse, et de vous épanouir là où il vous plaira.

I would also like to sincerely thank my colleagues at Duke University, especially Guillermo Sapiro and Matias Di Martino, with whom I had the pleasure to collaborate during a 6-months internship in 2019. This incredible experience confirmed my wish to pursue my career in research, and, for this, I will always be grateful to them.

Sans mes amis, cette thèse n'aurait probablement pas eu la même saveur. Un grand merci à vous pour tous ces moments joyeux, loin des préoccupations parfois obsessionnelles de la thèse. Je pense bien sûr à mes amis de longue date, rencontré à Rennes. Rafael, en particulier, je te remercie pour ton soutien sans faille. Un grand merci évidemment à Joe-Neil, Inès, Jean, Charlotte, Louise, Pauline, Aglaë, Hugo, Gabriel, Willka, mais aussi Carole, Caroline, Aurore, Romane, Allan, Salomé, Linda, Bérénice et bien d'autres encore.

Je n'oublie évidemment pas mes amis rencontré à Paris à Télécom Paris(tech?) ou ailleurs. Un immense merci à Antoine, Marc-Antoine, Ella, Jane, Jules, Luc, Mathilde, Olivier, Virgile, Lucile, Clément, Céline, Louise, Yassine, Pierre-Francois, Chloé, Neige, Matthieu, Zein, Lisa, Alix, Elie, Natty, Julien, Ingrid, Slim et bien d'autres encore. Pour finir ces remerciements amicaux, j'aimerais saluer mes (ex-)colocataires, Mathis et Alban, pour leur générosité et leur patience au quotidien lors de ces deux dernières années.

Last but not least, mes derniers remerciements vont à ma famille, sans qui rien de tout cela n'aurait été possible. A mes grands-parents paternels, Marc et Denise, un grand merci pour votre gentillesse et vos encouragements tout au long de mes études, je vous embrasse. Un grand merci aussi à Mickaël pour ta présence toujours très chaleureuse. Vorrei anche ringraziare la mia famiglia italiana, sia a Roma che a Rennes. Grazie a Donata, Paolo e Antonia, che ho avuto il piacere di ritrovare spesso a Rennes. Un abbraccio forte a Domitilla, Alberto, Emma e Caterina, spero venire a trovarvi presto a Roma! Finalmente, ringrazio Nonna Lisa, per ricordarmi l'importanza dell'arte e dei libri.

Je dédie cette thèse à mes parents, Yves et Nicoletta, qui m'ont tant donné et tant appris. Enfin, mes derniers remerciements vont à ma grande soeur, Juliette. Je marche dans tes pas depuis petit que ce soit pour la natation et les études. Tes conseils et ton enthousiasme m'ont toujours beaucoup aidé, et je t'en serai toujours reconnaissant.

Contents

Abstract	ii
Acknowledgments	v
Contents	vi
List of Figures	ix
List of Tables	xi
I Introduction	1
I.1 Context	1
I.2 Goal	2
I.3 Challenges	3
I.4 Outline	3
I.5 Contributions	4
II Background	7
II.1 Image acquisition	7
II.1.1 Image formation	7
II.1.2 Image distortion.	10
II.2 Image restoration	11
II.2.1 Filtering for image denoising	11
II.2.2 Non local methods	14
II.2.3 Variational methods	15
II.3 Deep learning	17
II.3.1 Neural networks: the multi-layered perceptron	18
II.3.2 Convolutional neural networks	19
II.3.3 Applications	21
III Related works	25
III.1 Deep learning image restoration methods	25
III.1.1 Standard CNNs for image restoration	25
III.1.2 Alternative deep restoration methods	27
III.1.3 Non-local and attention based architecture for image restoration	29
III.1.4 The perceptual impact of loss functions	34

III.1.5	Real-world image restoration	37
III.2	Synthetic training in deep learning	41
IV	Dead leaves images: a synthetic image prior for simple image restoration tasks	47
IV.1	Introduction	47
IV.2	Dead leaves images	48
IV.2.1	The continuous dead leaves model	48
IV.2.2	The generation algorithm	49
IV.3	Experimental results	51
IV.3.1	Denoising results	52
IV.3.2	Single-image super-resolution	55
IV.4	Conclusion and Perspectives	56
V	Fully synthetic training for real world image denoising	59
V.1	Introduction	59
V.2	Synthetic RAW dead leaves images	60
V.3	Distortion Modeling	62
V.3.1	RAW smartphone images noise model	62
V.3.1.1	Noise formula	62
V.3.1.2	SIDD dataset noise parameters estimation	63
V.3.2	Low-light image enhancement	66
V.3.2.1	Distortion Model	66
V.3.2.2	See-in-the-Dark noise parameters	66
V.4	Experiments	68
V.4.1	Noise removal on Smartphone RAW images	68
V.4.2	Low-light image enhancement	69
V.5	Conclusion and Perspectives	70
VI	A study on the color distribution in natural images and an application to synthetic training of networks	73
VI.1	Introduction	73
VI.2	Background	74
VI.2.1	Some color spaces	74
VI.2.2	Modeling the distribution of colors in natural images	76
VI.3	A density estimation of the natural image color distribution	78
VI.3.1	A 3D Gaussian mixture model	79
VI.3.2	A decoupled color/luminance model	80
VI.3.2.1	2D color distribution	80
VI.3.2.2	A color sampling algorithm	81
VI.4	Training a denoising network for smartphone RAW images	84
VI.4.1	Dead leaves image generation specifics	84
VI.4.2	Denoising result.	86
VI.5	Conclusion	88
VIIA	perceptual loss to improve texture preservation	91

VII.1 Introduction	91
VII.2 A frequential loss for dead leaves images	92
VII.2.1 Background	92
VII.2.2 Acutance loss for image restoration CNNs	96
VII.3 Image denoising results with FFDNet	97
VII.3.1 Finding the optimal perceptual loss parameter	97
VII.3.2 Spectral preservation	98
VII.3.3 Visual analysis	99
VII.4 Conclusion	101
VIII Conclusions and perspectives	103
VIII.1 Conclusions	103
VIII.2 Perspectives	106
VIII.2.1 Improving the realism of the synthetic dataset	106
VIII.2.2 Extending the statistical analysis of the color distribution	108
VIII.2.3 Investigating alternatives for the acutance loss	110
Bibliography	111

List of Figures

II.1	Color Filter Arrays	8
II.3	Camera ISP	9
II.4	Image distortions	12
II.5	Block Matching example	14
II.6	BM3D diagram	15
II.7	Non local denoising algorithms	16
II.8	TV-denoising examples	17
II.9	Perceptron	17
II.10	Convolutional layer diagram	19
II.11	VGG16 architecture	20
II.12	Computer vision tasks	21
II.13	Architecture of the Unet segmentation network [ronneberger2015u]	22
II.14	Advances in facial image generation through time presented in [face_generation]	22
III.1	Architecture of the denoising network DnCNN	26
III.2	Architecture of FFDNet	27
III.3	Blind spot network architecture	28
III.4	NLRN architecture	33
III.5	Receptive field of DGCN	33
III.6	Real denoising datasets	38
III.7	Neural ISP replacement	39
III.8	3D rendered image datasets	43
III.9	Examples of synthetic image processes used for training	44
IV.1	Illustration of the down sampling step	50
IV.2	Dead leaves images generated with different parameters.	51
IV.3	Example of images of the different training datasets.	52
IV.4	Denoising comparison of different FFDNet trainings	54
IV.5	Visual illustration of the ablation study	56
IV.6	Visual results for the super-resolution task	57
V.1	Bayer frame manipulations	60
V.2	Low light RAW dead leaves images	62
V.3	Estimation method for the shot noise parameters	65
V.4	Noise parameters of the SIDD dataset	65
V.5	Comparison of the simulated noise and the real noise	65

V.6	Parameter estimates for low-light image enhancement	67
V.7	Real denoising results.	69
V.8	See-in-the-Dark test examples	71
VI.1	Color matching functions from the CIE experiments of 1931.	75
VI.2	Chromaticity diagram from the CIE-XYZ color space	75
VI.3	Images and their corresponding color point cloud in the sRGB cube	77
VI.4	Examples of photographs from the RAISE dataset	78
VI.5	Graph of the likelihood of the 3D GMM against the number of components	79
VI.6	3D views of the distribution of colors	80
VI.7	Distribution of natural colors in the 2D representation	81
VI.8	Graph of the likelihood of the 2D GMM against the number of component	82
VI.9	Distribution of the average grey level knowing the position (x, y) in the 2D color representation.	82
VI.10	3 different color points cloud for dead leaves image generation	84
VI.11	RGB visualization of generated RAW Dead Leaves patches with different color models.	85
VI.12	Color gain inversion functions for different gain values	86
VI.13	Comparison of the different denoising models. From left to right: ground truth image, noisy image, 3DM-DL denoising,2DM-DL denoising,Nat-DL denoising.	87
VII.1	Unofficial dead leaves image target used for camera evaluation	92
VII.2	Grey level dead leaves image and its associated digital spectrum in logarithmic scales. The theoretical value is a straight line.	94
VII.3	Profile of the CSF of the human visual system for spatial frequencies expressed in cycles/degree.	95
VII.4	Diagram explaining the computation of the acutance metric	96
VII.5	PSNR vs acutance	98
VII.6	Frequential impact of the acutance loss	99
VII.7	Visual impact of the acutance loss	100

List of Tables

IV.1	Numerical comparisons of the different trainings of FFDNet. We evaluated the results on two benchmark datasets for image denoising (CBSD68 and Kodak24), and our dead leaves testset, at two noise levels. Each cell contains the triplet PSNR/SSIM/PieAPP. The best results are in blue , the second in red	53
IV.2	Impact of the parameters and ablation study	55
IV.3	Numerical evaluation of our super-resolution results	55
V.1	Numerical evaluation of the denoising of smartphone RAW images. We report the PSNR of our adapted Unet trained either on the dead leaves dataset or on the SIDDD-small dataset with real or synthetic noise.	69
V.2	Numerical evaluation of our low-light enhancement networks. We report the PSNR of our three different models at different time exposure ratios. In last three lines, we also report a corrected PSNR where we prescribe the real average of each channel to the output image.	70
VI.1	Numerical evaluation of the denoising of smartphone RAW images. We report the PSNR and SSIM the denoising models trained on dead leaves images obtained with different coloring strategies.	87



Introduction

I.1 Context

More than ever, photography is omnipresent in our daily lives. With more than fifty-four thousand four hundred pictures taken per second in the world with smartphones, the demand for high-quality cameras is ruthless. However, this quest for the best image quality faces many constraints. Among them, size is the most critical. Smartphones tend to be small, and the size left for a digital sensor and a lens is tiny, drastically limiting photon acquisition. Due to this size constraint, the rendered image quality is poor without smart processing. Another challenge is producing high-quality images in harsh conditions. For instance, smartphone users expect to have a perfectly sharp image with hand-held devices. Without any stabilization, the resulting image should contain motion blur. In even more extreme conditions, such as night-time photography, the users want to retrieve a sharp image with a good dynamic range and correct illumination. Meeting all those expectations is challenging and can not be solved by just improving the hardware properties. To restore the distorted and low-quality output of the digital sensor, we must apply complex algorithms to produce a satisfactory image.

For all these reasons, image restoration has been one of the most critical subjects in the image processing field. The main objective is to recover an image as close as possible to a perfect image, starting from a distorted observation of the scene. Formally, if x is the ground truth image, and y is a distorted observation, many deterioration processes follow the relation:

$$y = Ax + n, \tag{I.1}$$

where A is a linear operator, and n is noise. Most image restoration problems fall into this set-up, whether it is deblurring, for which A is a blur operator, Gaussian denoising, where n follows a normal distribution, or inpainting where A is a masking operator.

To tackle these problems, many methods have been developed over the years. A first class of methods is based on prior knowledge or hypotheses on the distribution of natural images.

These hypotheses usually model the regularity of the image content, and the algorithms derived from them try to enforce this regularity in the restored image. Among these methods, we can cite variational methods [167], which assume that images are locally smooth with sharp edges, or non-local methods [25, 42], which assume that the content of the image is redundant.

In the last decade, the image processing community focused on approaches based on deep learning. These methods, first very successful in computer vision tasks such as classification [77] or segmentation [75], were then adapted to image restoration problems [200]. They soon outperformed previously developed methods for most restoration tasks. Rather than assuming some prior knowledge about the distribution of the data, these methods try to extract complex features from the data itself in order to predict a restored image from a distorted observation. For them to work efficiently, these methods require large amounts of images grouped in pairs of distorted and perfect observations of the same scene. This data is then used in an optimization framework (the training) to fit the neural network weights to the task at hand.

I.2 Goal

Even though neural methods considerably improved the performances in image restoration tasks, this comes with its share of problems and limitations. First, neural networks generalize poorly to modality changes. For instance, a denoising network trained for Gaussian noise will not properly restore images distorted with Poisson noise. This makes them over-specialized and not very scalable. Second, as mentioned above, neural networks need a large amount of data to be efficiently trained. While data may be easy to collect in some simple situations, this can be very challenging for specific tasks. Moreover, it is nearly impossible to acquire a ground-truth image that has not been distorted in most real-world cases. The most simple example is denoising, for which acquiring a noiseless image is impossible because of the stochastic behavior of photons. On the other hand, prior-based methods scale quite well to new distortions, without needing to acquire data or to retrain a neural network, saving both time and effort.

Rather than acquiring real data, we propose generating fully synthetic images with very simple algorithms. By using our prior knowledge about the distribution of natural images, we wish to enforce the desired properties in our generated images so that we can efficiently train neural networks for image restoration. Such properties could be, for example, the distribution of the gradient or the distribution of the spectrum. We believe this would allow for much lighter training schemes, with the possibility to tune the generation algorithm to the task at hand. In addition to reducing the human and computational cost of training a network, this work also shed light on the inner workings of neural networks by explicitly showing what properties are sufficient for them to work efficiently. The present dissertation somehow bridges the gap between prior-based methods and deep learning methods by enforcing a prior on the training data used by the neural network.

I.3 Challenges

To achieve this goal, the most obvious question is: what synthetic image models are we going to use and why? The chosen image model has to be simple, and most importantly, we must be able to successfully train a neural network for image restoration with images generated from that model. Therefore, it has to account for both homogeneous areas, textures, edges, and the colors we find in natural images. To answer that question, we tried a variety of synthetic image models with different properties. After training our networks on those synthetic images, we empirically show which are the sufficient properties to train a neural network for image restoration.

One of our objectives was to provide a model that can be easily tuned to real-world situations. However, an image generation algorithm will only produce perfect images without distortions. As we mentioned before, both clean and distorted images are needed to train a neural network for image restoration. With these observations comes our second question: can we tune our generation algorithm to complex tasks, and can we model real-world deterioration processes accurately? To that end, we propose a way to model real-world distortions and estimate their parameters accurately. We then study how our synthetic image model and our synthetic distortions interact in the context of training a neural network for image restoration for real-world cases.

I.4 Outline

The present dissertation is organized in 6 chapters. In Chapter II, we begin by introducing helpful background for understanding the following chapters. We explain the image acquisition process and the possible distortions an image undergoes. We also give a brief introduction to classic image restoration methods, as well as deep learning and its applications. In Chapter III, we thoroughly review more modern related works, covering image restoration with deep learning methods and synthetic training of neural networks.

In Chapter IV, we present a novel approach to training neural networks for image restorations with synthetic images. We begin by presenting our image generation algorithm based on the dead leaves model. We then experimentally show the role of every property of our model to train a neural network for image restoration properly. We base our analysis on two simple tasks: additive white Gaussian noise removal and single-image super-resolution.

In Chapter V, we show that we can expand our synthetic training to more complex real-world tasks, such as the denoising of smartphone RAW images, and low-light image enhancement. As mentioned above, a synthetic generation model is insufficient to train a restoration method since we need distorted data. To that end, we provide an accurate distortion model and a simple yet efficient noise parameter estimation method for both tasks. We experimentally show that our model trained on synthetic images and synthetic distortion processes performs on par with models trained on real-world data.

One of the drawbacks of our generation algorithm is that it requires a real image to sample colors for each synthesized image. While the geometry is dictated by very few parameters, the number of color parameters is much higher. In an attempt to reduce this number, we study in Chapter VI the first-order statistics of the color distribution in natural images. Based on a simple heuristic proposed by Omer et al. [146], we propose a coloring technique

based on few parameters. This color model can be used to synthesize dead leaves images, which can in turn be used as a training set for image restoration tasks. Our experiments show that our coloring technique leads to excellent denoising performances.

All previous chapters show that the dead leaves image model can be used to train a neural network for image restoration. Prior to these works, dead leaves images have also been used for a different application: image quality evaluation of cameras. Dead leaves images indeed exhibit many invariance properties, which are a requirement to test cameras in different conditions. More importantly, they contain contrast at every amplitude and every scale. Moreover, their statistical properties make them close to natural images. Inspired by the works of Cao et al [33] and Artmann [14], we transform, in Chapter VII, the evaluation metrics presented in these papers into a loss function to train neural networks. We then use this loss as a perceptual metric to train a neural network for image restoration.


I.5 Contributions

The works presented in the following chapters present some novel contributions to the image processing field, that we believe to be of significant interest.

- To the best of our knowledge, the paper associated with Chapter IV was the first attempt to train a neural network for image restoration tasks on synthetic images, and more specifically on dead leaves images. Even though these images were known for their statistical properties, they were never used in this framework. The flexibility and ease of implementation of this model are a real benefit. The restoration results on these images show that such synthetic images are a viable alternative to heavy data acquisition campaigns.
- Motivated by the reduction of the number of color parameters for the dead leaves image generation, we studied the empirical distribution of colors in natural images. Though the distribution of colors has been studied in the literature, our work presented in Chapter VI is the first to propose a parametric model for the first-order statistics of colors in natural images, and to show the practical interest of such a parametric model.
- Since dead leaves images were never used to train a neural network, neither was the acutance loss which we define in Chapter VII. This perceptual metric, which was first used to evaluate the camera's capacity to render textures, is based on the analysis of the networks' frequential response. Most importantly, this chapter shows that a simple learned procedure can greatly improve some standard texture quality measure without affecting more traditional measures.
- In addition to these novel ideas, we put an emphasis on making our ideas applicable to real-world problems. That is why we dedicated much effort to synthesizing RAW images, as well as estimating the noise parameters of different cameras. Along with this work presented in Chapter V, the idea presented in Chapter VII potentially has a direct industrial application for camera manufacturers.

During this doctoral program, we published our first paper at the SSVM conference, entitled *Synthetic images as a regularity prior for image restoration neural networks*[6]. A

second journal paper has been accepted at *Computer Vision and Image Understanding*, entitled *Fully synthetic training for image restoration tasks*[4] regarding Chapter V. A third paper has been accepted at SSVM 2023, entitled *Hybrid Training of Denoising Networks to Improve the Texture Acutance of Digital Cameras*[5], regarding the results in Chapter VII. One other paper is under review for a conference for the results of Chapter VI. I would also like to mention my collaboration with Guillermo Sapiro and Matias di Martino, which resulted in a paper at the ICASSP conference, entitled *Nested Learning for Multi-Level Classification*[3]. Though this work has little to do with the thesis subject, I had the pleasure of working on it during my doctoral program. In a few words, this work aims at making classification neural networks produce hierarchically nested predictions, reducing overconfidence and improving robustness.



II Background

II.1 Image acquisition

As we mentioned above, image restoration is critical to provide satisfactory images. An image indeed undergoes many distortions during its acquisition and its processing, which need to be corrected. Before describing these distortions, let us recall the different steps of image formation.

II.1.1 Image formation

Photons acquisition. A scene reflects or emits light rays at different wavelengths carried as energy by photons. To photograph that scene, a camera tries to acquire these photons. They first go through a sequence of lenses before hitting a sensor in the focal plane of these lenses, which in all that follows, will be digital. Except for an ideal pin-hole camera, no photographic lens can produce a perfectly sharp image in its focal plane because all lenses violate the Gauss conditions [162]. This implies a scene-independent blur, which will occur in every image taken with that device. This blur can be modeled by a kernel B_i . In addition, extrinsic blur can happen during a shot because of motion or defocus, modeled by a kernel B_e . Letting x be a perfectly sharp representation of the scene as a function from \mathbb{R}^2 to \mathbb{R} , we observe y , which follows

$$y = B * x,$$

where $B = B_e * B_i$ and $*$ is the convolution operator. Using a pin-hole camera is obviously inefficient since almost no photons go through, forcing the user to expand the exposure time, making our image prone to motion blur and moving objects in the scene. That is why we use lenses that allow us to capture more light at the cost of not being perfectly sharp.

After passing through the lens, the photons hit the sensor. It is a rectangular array made of square photosites which accumulate photons with either a couple-charged device (CCD [22]) or complementary metal-oxide-semiconductor (CMOS [58]) sensors during ex-

posure. The photoelectric effect induces an electrical signal which can be converted into a digital one, quantized over 10, 12, 14, or 16 bits, depending on the camera. Because of the stochastic behavior of photons, the photon count in a photosite is a realization of a Poisson law $\mathcal{P}(\lambda)$, where λ is the average photon count. This implies a signal-dependent noise, commonly called *shot noise*. In addition, the sensor’s electronic read-out also introduces a signal-independent noise, called *read noise*, often modeled by a Gaussian noise of constant and device-specific variance σ^2 . The resulting noisy image y can be determined by the following formula:

$$y_{\text{noisy}} = Q[B * x + n_{\text{shot}}(B * x) + n_{\text{read}}],$$

where Q is the quantization operator, n_{shot} is the shot noise, and n_{read} is the read noise. Formally,

$$B * x + n_{\text{shot}}(B * x) \sim \mathcal{P}(B * x),$$

where \mathcal{P} is a Poisson distribution of mean and variance equal to $B * x$.

In order to retrieve color information, each photosite is covered by a color filter, allowing photons in the photosite only if their wavelengths belong to the desired range of wavelengths accepted by the filter. The color filtering is organized as an array (CFA) with a regular pattern, with red, green, and blue filters. The most common CFA is the Bayer pattern [18] (see Fig. II.1a), but other camera makers, such as Fujifilm, adopted other patterns such as the X-Trans array (see Fig. II.1b). Interestingly, there are twice as many green photosites as red or blue photosites in the original Bayer CFA. This choice was motivated to mimic our visual physiology, our cone cells being most sensitive to green wavelengths.

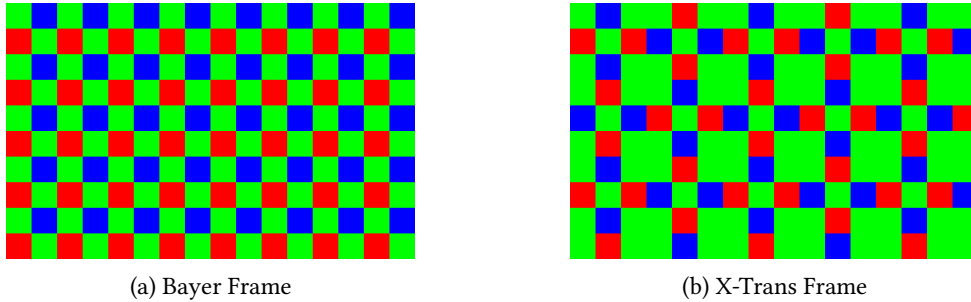


Figure II.1: Different Color Filter Arrays(CFA).

Image development. In the quantized digital signal, which we call the RAW image of size $(H, W, 1)$, each pixel value corresponds to a red, green, or blue value depending on the CFA and the position of that pixel in the frame. Formally, $y_{\text{raw}} = m \odot y_{\text{noisy}}$, where m is a masking operator accounting for the CFA. In order to get a viewable image on a digital screen, we need to interpolate the missing values at every position to get an image of the same resolution $(H,W,3)$. This *demosaicking* task can be achieved by bilinear, or bicubic interpolation [117, 81], using filter banks [130], or most recently with neural networks [64, 55].

To retrieve an image that visually corresponds to the scene, the output of the demosaicking is not satisfactory. First, the colors must be corrected to fit our perception of the scene. Because the illuminant varies, and because of the varying sensitivity functions of the color

filter arrays depending on the camera, a color space change is applied as well as a white-balance operation [59]. The colors are first projected in the standard RGB (sRGB) space [11] (usually with a simple matrix multiplication) for a uniform color representation across cameras. Then, the white balance step consists in multiplying each color channel by a computed gain G_c , which balances the colors in the image. These parameters are found to fit some global assumptions such as the white-patch prior [105], or the grey-world prior [28].

The color-corrected image still needs to fit our perception of the scene. The luminance of the image L depends linearly on the scene's radiance. However, our visual system's response to radiance is far from linear. We are more sensitive to contrast in the darker tones than in the brighter tones. To account for this discrepancy, a nonlinear transform in the shape of a power function is applied to the image. This transform, called the gamma correction, can be expressed as $\Gamma(x) = x^{2.2}$, $x \in [0, 1]^{(H,W,3)}$. In Fig. II.2, we observe the difference between a non-corrected linear gradient and a gamma-corrected linear gradient.

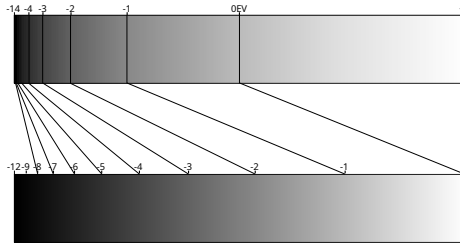


Figure II.2: Linear gradient versus Gamma corrected gradient

The final image y_{RGB} is determined by the following equation, which combines all the operations in the treatment of a RAW image:

$$y_{\text{RGB}} = \Gamma(G_{\text{wb}}((C_{\text{sRGB}}(D(y_{\text{raw}}))))),$$

where D is the demosaicking function, C_{sRGB} the color space change operation, G_{wb} the white balance gain operator. In Fig. II.3, we show the intermediate states at every step of the pipeline, starting from a RAW image.

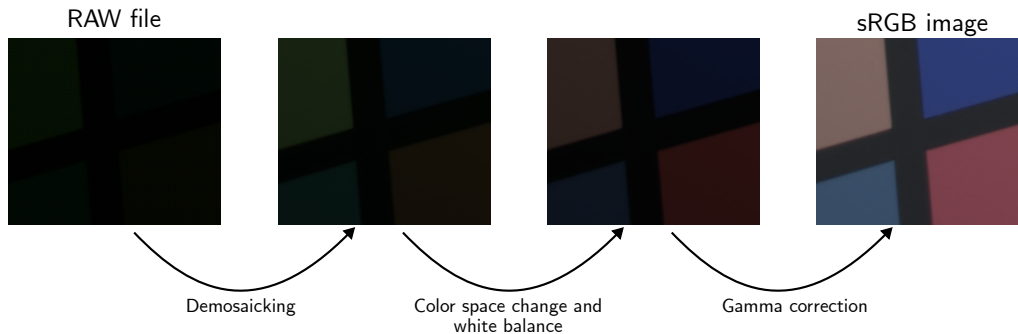


Figure II.3: The classic steps of the image signal processing (ISP) pipeline

Other steps might be present in developing a RAW image, such as lens distortion correction. Some manufacturers also include additional proprietary steps to further improve the quality of images, such as HDR algorithms, which increase the dynamic range of the image [158].

II.1.2 Image distortion.

So far, we have seen that a digital image always suffers from blur and noise up to a certain magnitude. However, other distortions may happen during or after the acquisition and development of a digital image.

Compression. First, compression often induces some artifacts. In order to save storage space, the digital signal is either compressed using a sparse representation, subsampled, or both. The most common compression technique is the JPEG compression, based on filtering the Discrete Cosine Transform coefficients of 8×8 patches [182]. This compression often results in block-like artifacts, which we can observe in Fig. II.4a along with other compression examples. Based on wavelet coefficient shrinkage, a less popular compression called JPEG 2000 is more effective and produces fewer artifacts for high compression rates [173]. Resizing the image to a lower resolution is also possible by applying a low pass filter and a bicubic interpolation to respect antialiasing conditions. This distortion induces a loss of high frequencies, making the image less sharp.

Noises. Second, noises can also distort the image. We have seen that an image is affected by signal-dependent noise, which follows a Poisson distribution, and by signal-independent noise, which we call read noise. Along with these noises, we present other possible noises in the following list. We then give examples in Fig. II.4b.

- **Gaussian noise:** this model is a good approximation of the electronic noises during the read-out phase of an image called read noise. It follows an iid Gaussian distribution $n \sim \mathcal{N}(0, \sigma^2)$. It usually accounts for different noise sources, such as thermal noise, dark current noise, and source-follower noise. However, it is also the most commonly found approximation for all digital noises. Since it is signal-independent, this model is far from reality.
- **Salt and pepper noise:** also called impulse noise, it accounts for transmission errors or dead photosites in the sensor. It either increases or decreases the value of randomly sampled pixels to the minimal or maximal value, as we can see in Fig. II.4b. The salt and pepper noise distribution are given by Poisson law:

$$P(k) = \frac{e^{-\lambda} \lambda^k}{k!},$$

where $P(k)$ denotes the probability of having k noisy pixels in a window of a specified size, with λ being both the average and variance of that law.

- **Speckle noise:** It can be found in other imaging devices than DSLRs, mostly in Synthetic Aperture Radar images (SAR). The particularity of that noise is that it is multiplicative and not additive. The speckle noise is modeled by the following formula

$$y = s \times x,$$

in which x is a noiseless image, y is the noisy observation, and s is a multiplicative noise that follows a Gamma distribution.

An exhaustive list of digital image noises is given in [125].

Other distortions. Occlusions are also distortions that one may want to remove from a photograph. These occlusions can be either macroscopic, i.e., a large object that hides the photograph's subject or dust on the sensor, or microscopic, i.e., fine suspended particles causing haze. In the case of macroscopic occlusion, a masking operator m is applied to the image with m_Ω defined as:

$$m_\Omega: [0, \dots, H] \times [0, \dots, W] \rightarrow \{0, 1\} \quad (\text{II.1})$$

$$(i, j) \mapsto \begin{cases} 1 & \text{if } (u, v) \in \Omega \\ 0 & \text{if } (u, v) \notin \Omega \end{cases}, \quad (\text{II.2})$$

where Ω is the set of positions corresponding to the occluding object. This mask is then point-wise multiplied by the original image x , $y_{\text{masked}} = m_\Omega \odot x$. Finally, old photographs or digital images may be in black and white, therefore losing color observations. We give examples of these distortions in Fig. II.4c.

Most of these distortions can be modeled employing a linear operator accounting for deterministic distortions and an additive noise accounting for stochastic distortions, thus fitting with Eq. (I.1). The goal of image restoration is to reverse this distortion process to get an estimate \tilde{x} from a distorted observation y as close as possible to the original image x . A relevant criteria for measuring the reconstruction quality is the *mean squared error*: $MSE = \|x - \tilde{x}\|_2^2$. In the following section, we will introduce different classes of algorithms which aim at minimizing the latter error.

II.2 Image restoration

In this section, we will review a wide range of classical image restoration techniques. First, we will introduce signal filtering applied to image denoising. Then, we will highlight a more recent class of techniques based on the self-similarity assumption in images: non-local methods. To conclude this section, we will address variational methods applied to image restoration.

II.2.1 Filtering for image denoising

For simplicity, we suppose that noise follows an additive white Gaussian model, i.e.,

$$y = x + n, \quad n \sim \mathcal{N}(0, \sigma^2).$$

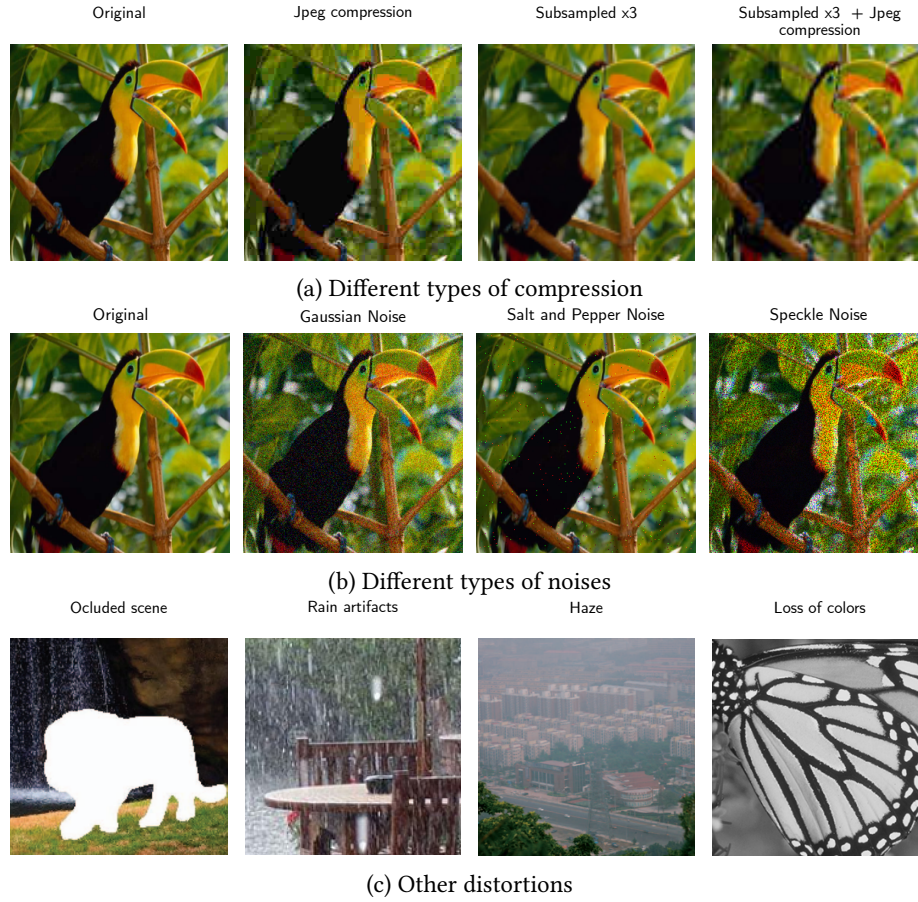


Figure II.4: Various examples of image disortions.

Inspired by signal processing, the first methods developed to remove noise from images were based on linear filtering. Shift-invariant linear filters are equivalent to convolving the image with the filter-associated kernel B . At a pixel i, j , the filtering can be expressed by:

$$\hat{x}_{i,j} = \sum_{k,l \in [-r,r]^2} y_{i-k,j-l} B_{k,l},$$

where we suppose that B has a support of size $(2r + 1, 2r + 1)$.

Linear filtering. Based on the assumption that images are locally smooth, a simple idea consists of locally averaging the pixels, with a simple averaging kernel or a Gaussian kernel. In a global setting, the reconstruction error arising when applying a weighted average filter can be written as:

$$\begin{aligned} MSE &= \mathbb{E} \|x_{i,j} - \hat{x}_{i,j}\|_2^2 \\ &= \mathbb{E} \left\| \sum_{k,l \in [-r,r]^2} B_{k,l} (x_{i,j} - x_{i-k,j-l}) - \sum_{k,l \in [-r,r]^2} B_{k,l} n_{i-k,j-l} \right\|_2^2 \\ &= \left\| \sum_{k,l \in [-r,r]^2} B_{k,l} (x_{i,j} - x_{i-k,j-l}) \right\|_2^2 + \sigma^2 \sum_{k,l \in [-r,r]^2} B_{k,l}^2, \end{aligned} \quad (\text{II.3})$$

where we assume that the the image and the noise are independent, that all weights are positive and sum to 1. This decomposition shows that such filtering reduces the noise's variance. For a simple averaging, the variance is divided by $(2r + 1)^2$. However, the reconstruction error is bad if the pixel intensities show fast variations in a small region. Though this approach works for homogeneous images, it necessarily blurs discontinuous signals, such as edges in images, resulting in a large MSE.

Concerning linear filtering in the frequency domain, we can also mention the Wiener filter [153]. It is the MSE-optimal stationary linear filter for images degraded by additive noise and blurring. To compute the coefficient of this filter, we need to suppose that the noise is signal-independent and that X and N are stationary signals of known power spectral density (PSD). However, the knowledge of the PSD of the original signal is an assumption that can not be met in real-world problems. One can either use an oracle from another restoration technique or formulate a hypothesis on the distribution.

Edge-preserving filtering. To address the edge-smoothing problem, the first option was to use nonlinear filters. Median filtering, for instance, produces better results regarding edge-preservation [140, 84]. Rather than assigning the mean of a patch to its central pixel, it assigns its median. Though this method produces better results, the patch-wise pixel sorting operation is expensive to compute and not well adapted to additive noise. A faster and more efficient nonlinear filtering that preserves edges is the bilateral filter [176]. For each pixel in the output image, it computes a weighted average of the input image centered at the same position. In this case, the weights depend on the spatial distance of the pixels and the differences in intensity. More precisely, bilateral filtering follows the following formula:

$$\hat{x}_{i,j} = \frac{1}{W} \sum_{k,l \in [-r,r]^2} B_{k,l} \cdot f_r(\|y_{i,j} - y_{i-k,j-l}\|_2^2) y_{i-k,j-l},$$

where B is a weighting kernel for spatial similarity and f_r is a weighting kernel for intensity similarity. The edge preservation is the consequence of weighting the average depending on the difference in intensity.

Sparsity based restoration. Another practical idea in filtering consists of decomposing the image on a suitable basis and threshold the decomposition of the distorted image. The underlying hypothesis is that natural signal decompositions are sparse, with only a few components. Every coefficient is affected when noise is added to the original signal, making the signal decomposition dense. This is the idea behind wavelet coefficient shrinkage techniques [51, 50], or Discrete Cosine Transform (DCT) denoising [194]. Wiener filtering has also been applied to the coefficient of the wavelet decomposition, achieving good results [151, 94]. In the case of DCT denoising, every $(16, 16)$ patch is decomposed on the DCT basis, and coefficients below a certain threshold are put to zero. To remove the block-like aspect, this filtering is done on overlapping patches, and the results are aggregated with a weighted average. A multi-scale version of this algorithm improves the performances in the low frequencies of noise [148].

II.2.2 Non local methods

Natural images are highly redundant. In a given image, structures are often repeated, for example, edges, flat surfaces, or even textures. For a given patch in an image, one can usually find similar patches in the whole image either in the neighboring patches or further away in the image, as we can observe in Fig. II.5.

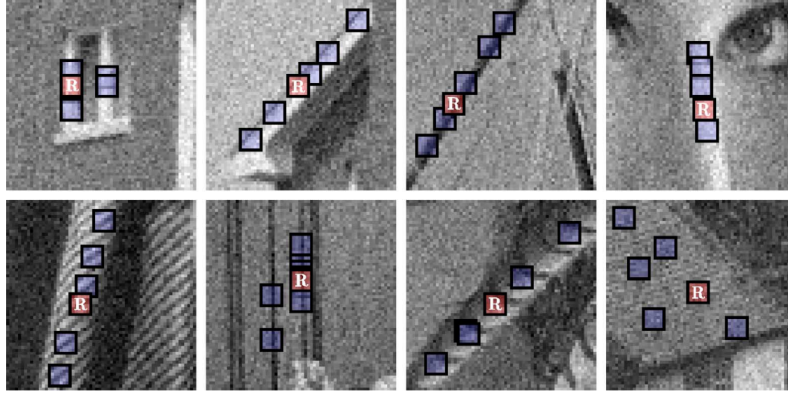


Figure II.5: Examples of patches found in an image that are similar to a reference patch (R). Diagram from the BM3D paper [42]

To exploit this property called *self-similarity*, a simple idea in image denoising consists of finding pixels with a similar neighborhood and averaging them based on a similarity measure. This idea is similar to the one used in bilateral filtering, which in contrast, limits itself to spatially close pixels. *Non-Local Means*, first presented by Buades et al. in [26], was the first method to leverage the self-similarity property fully. Previous to this work, Yaroslavky introduced neighborhood filters [193], which compute a weighted average of the noisy image pixels based on their difference in intensity. Unlike bilateral filtering, these filters discard spatial distance and use pixels in the whole image. However, because the noise affects the weights in the average, the independence conditions for Eq. (II.3) are not met, and we have no guarantees that the resulting image will be denoised.

Rather than computing the filter weights based on a pixel intensity similarity function, we can compute them by comparing the similarity of patches centered on these pixels. As explained in the experimental paper [143], given y a noisy black and white image and x the original image, the non-local means filtering at a pixel k writes as follows:

$$\tilde{x}(k) = \frac{1}{C(k)} \sum_{l \in B(k,r)} y(l)w(k,l), \quad C(k) = \sum_{l \in B(k,r)} w(k,l),$$

where $B(k,r)$ defines a large search window, centered in k of size $(2r+1, 2r+1)$. For a pixel $y(l)$, the weight depends on the similarity between a square patch of height $2h+1$ centered in k and a square patch of the same size centered in l . Letting $P_h(u)$ be the square patch of height $2h+1$ centered in u , the weights are defined as follows:

$$w(k,l) = \exp\left(-\frac{\|P_h(k) - P_h(l)\|_2^2}{2\sigma^2}\right),$$

where σ is the noise standard deviation. Therefore, the lower the patch distance, the higher the weight in the sum, and conversely.

By doing so, we are more confident that pixels close to each other with this metric will likely have the same noiseless intensity. This algorithm can also be extended to color images or even SAR images [48].

Following the same intuition, Dabov et al. proposed the BM3D algorithm [42]. The graph drawn in Fig. II.6 gives an overview of the different steps of the algorithm. Rather than computing a weighted average of the pixels with similar neighborhoods, they aggregate patches similar to a reference patch in a 3D tensor before collaboratively filtering them. This collaborative filtering is done by applying a 3D linear transform to the tensor, followed by a coefficient shrinkage on an orthogonal basis, and finally, applying the inverse 3D transform. Filtered patches are then replaced at their positions. Because patches overlap, the authors compute a weighted average of all the denoised values of a single pixel obtained on overlapping patches, with weights depending on the patch's spatial distance. This first step gives an oracle of the noiseless image, which will be used as input in a similar second step for two main reasons. First, this helps to better assess which patches are similar to the reference patch, since noise affects this estimation. Second, having an oracle allows for an accurate Wiener filtering of the coefficients, which is better than hard thresholding. This two-step procedure was also used in a Bayesian extension of the Non-Local Means algorithm called Non-Local Bayes [109]. Before the advent of deep learning techniques for image restoration, BM3D has long been the state-of-the-art denoising technique and generalizes better to different types of noises. In Fig. II.7 we compare the results of these non-local methods applied to the same image.

II.2.3 Variational methods

Variational methods impose a regularity prior to the restored image. This is usually done by minimizing an energy based on two terms: a data fidelity term D , and a regularity term R . Letting u_0 be the distorted observation and u be the candidate image, the problem can be written as:

$$\inf_u \int_{\Omega} d(A.u, u_0) dx + \lambda \int_{\Omega} R(u) dx,$$

where Ω is the image domain, A is the linear operator of the inverse problem, and λ is a scalar. The data fidelity function is usually the Euclidean distance. The difficulty here is to

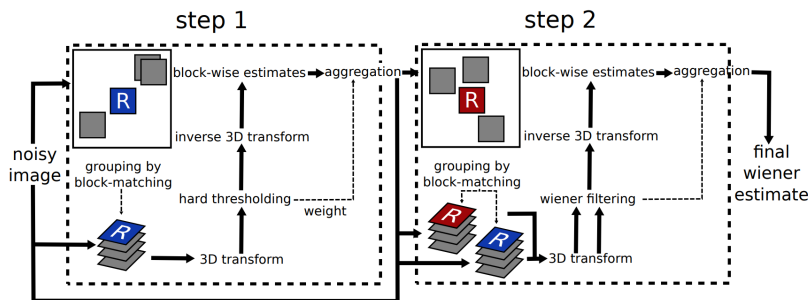


Figure II.6: Scheme of the BM3D algorithm from [108]

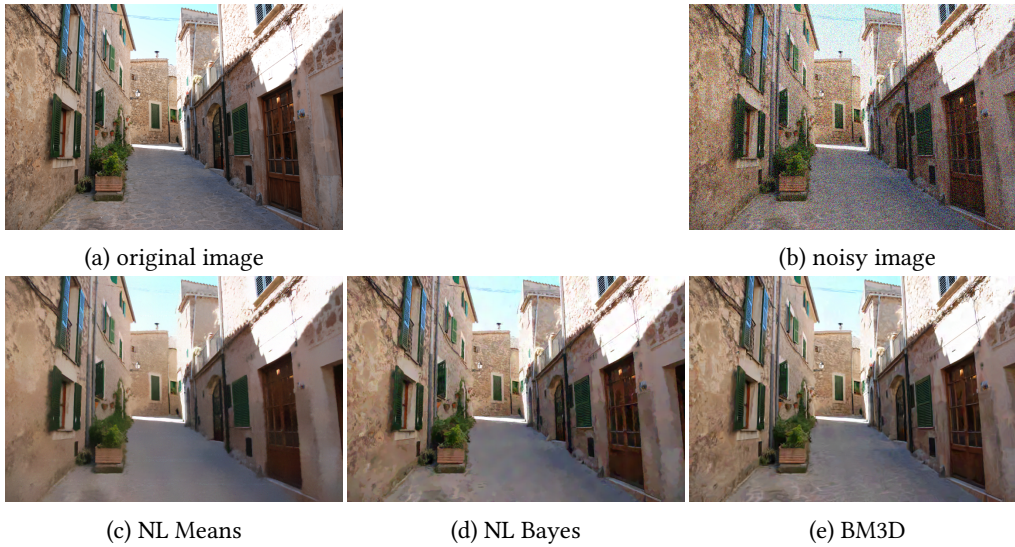


Figure II.7: Comparison of the results of various non local image denoising methods. We observe that BM3D clearly outperforms the two other methods.

design a good smoothing term. An early idea was to penalize the L^2 norm of the gradient. However, such a method tends to over smooth edges which have a high gradient. The explicit solution implies a Laplacian operator, which is isotropic and therefore blurs edges. A widely used improvement is to consider the *total variation* regularizer, which corresponds to the L^1 norm of the gradient. In a more global setting, we can consider the following problem:

$$\inf_u \|A.u - u_0\|_2^2 + \int_{\Omega} \phi(\nabla(u)) dx,$$

where ϕ is a strictly convex non-negative coercive function, growing at most linearly. In the simplest setting, relaxing the strict convexity requirement, $\phi(x) = |x|$.

Numerically, this minimization problem can be solved using iterative algorithms. In the seminal Rudin-Osher-Fatemi paper [167], the authors propose to solve this minimization by means of a gradient descent, each step of which consisting of solving the discrete version of a partial differential equation. Many other algorithms were developed to minimize this energy, such as duality-based methods [34, 211], Newton-based methods [141], and operator splitting methods [120, 70, 63]. The reader can find a detailed presentation of the mathematical foundations and implementation details of variational methods in [15].

Even though these methods are versatile and can be adapted to all kinds of noise and linear perturbation, they tend to produce close to piece-wise constant images with sharp edges and over smooth textures. Moreover, tuning the penalty term λ may be tedious, and a slight variation of this parameter significantly impacts the reconstruction of the images, as we can see in Fig. II.8.

All the presented image reconstruction techniques are based on a regularity prior on the distribution of undistorted natural images. Conversely, learning-based approaches try extracting relevant features by optimizing their response to a large set of pairs of clean and

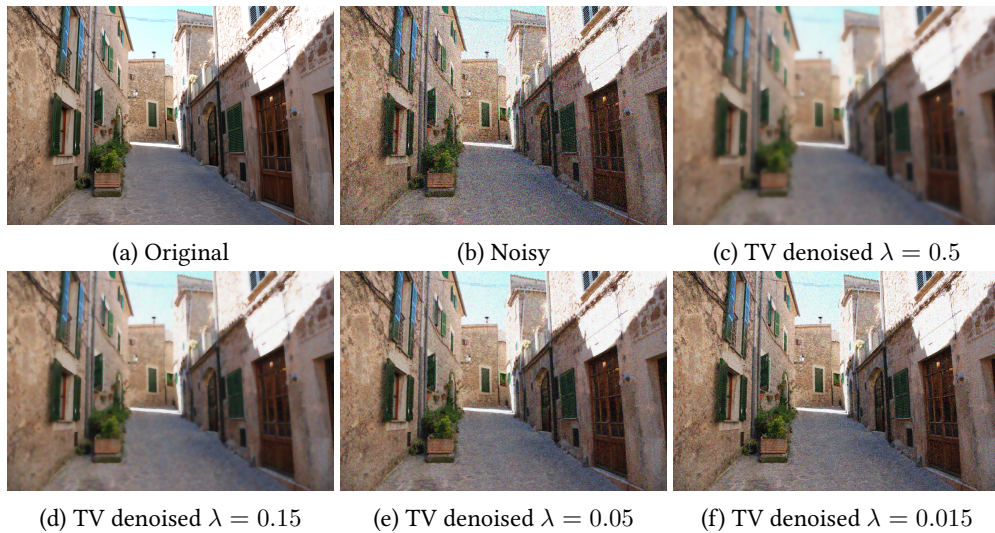


Figure II.8: Influence of the penalty term in TV-denoising with the Chambolle algorithm

distorted images. In particular, deep neural networks are trained to minimize a loss, usually the MSE between the restored image and the original image, with many examples. Before turning to deep learning for image restoration in Section III.1, we will first introduce the basic notions on neural networks and deep learning in the following section.

II.3 Deep learning

Deep learning is a popular field of machine learning based on the use of large neural networks. The scope of machine learning is to devise accurate autonomous decision-making algorithms. For them to work efficiently, these algorithms have access to large amounts of data to which they optimize their response. They can solve complex tasks such as classifying data, detecting objects of interest, etc. In this section, we will first introduce the basics of neural networks before exploring convolutional neural networks and showcasing some applications.

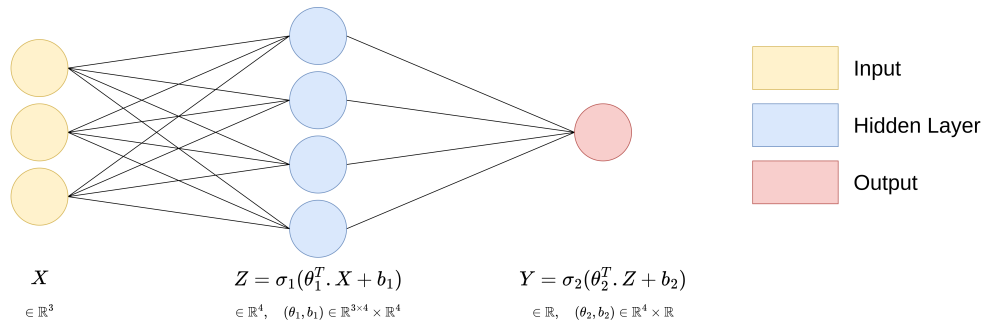


Figure II.9: Scheme of a simple 2 layer perceptron

II.3.1 Neural networks: the multi-layered perceptron

Neural networks were first introduced in 1958 by Rosenblatt [166], to mimic the behavior of the brain's neurons. In a very simple approximation, the neuron is thought of as a node that receives electrical signals from other neurons and outputs a new signal depending on the signals received. Formally if $x \in \mathbb{R}^d$ is the input signal of the neuron, the output can be written as:

$$f_{\theta,b}(x) = \sigma(\theta^T x + b),$$

where σ is a non linear function called the *activation function*. $(\theta, b) \in \mathbb{R}^d, \mathbb{R}$ are the parameters of that neuron respectively called *weight* and *bias*. Commonly used activation functions are the Rectified Linear Unit (ReLU) $\sigma(x) = \max(0, x)$, the hyperbolic tangent $\sigma(x) = \frac{2}{1+\exp(-2x)} - 1$ or the sigmoid function $\sigma(x) = \frac{1}{1+\exp(x)}$.

In a supervised learning setting for classification, that is, having access to an annotated training dataset $\mathcal{D} = \{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}\}_{i \in [0, N]}$, one may optimize the weight and bias of this neuron to fit to the training data. To measure if the weights are suitable for this task, we can evaluate a *loss function* over the training set, which quantifies the error made by the neuron. This function, which has to be differentiable, is often the mean-squared error or a cross-entropy function. However, in the case of a single neuron or *perceptron*, the data has to be linearly separable for the classification to succeed. For most real-world tasks, this is not the case. In order to classify data with a more complex distribution, we can compose multiple perceptrons, thus benefitting from the nonlinear transforms of the data by the activation functions. The obtained function is a basic neural network called the *multi-layered perceptron* or MLP. The network function becomes:

$$f_{(\theta_i, b_i)_{i \in [0, L]}}(x) = \sigma_L \left(\theta_L^T \sigma_{L-1} (\dots \sigma_0 (\theta_0^T x + b_0) \dots) + b_L \right).$$

In this equation L is the number of layers of the network, which can be arbitrarily large. In Fig. II.9, we give an example of a simple 2-layer MLP.

Optimization. Having such a function, we want to solve the optimization problem:

$$\operatorname{argmin}_{\theta \in \Theta} \mathbb{E}_{X, Y} [\mathcal{L}(f_{\theta}(X), Y)],$$

where \mathcal{L} is an arbitrary differentiable loss function, (X, Y) are random variables following the data distribution, and Θ , the parameters vector space. To simplify notations, we write: $(\theta_i, b_i)_{i \in [0, L]} = \theta$. In practice, we can only minimize the empirical error $E = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f_{\theta}(x_i), y_i)$, where $(x_i, y_i)_{i \in [1, \dots, N]}$ is the training dataset. A simple yet effective way to do so is to implement the Stochastic Gradient Descent (SGD) [163, 96]. Fortunately, the neural network's gradient is easy to compute since it is a composition of linear and non-linear functions [71].

Starting from randomly initialized parameters, the SGD iteratively updates the weights by performing a descent step in the direction of an estimate of the gradient. This estimate is obtained by averaging the current gradient function evaluated on a subset of the dataset named a *mini-batch* and constructed by random sampling. For simplicity, the mini-batches d_k may be supposed to be disjoint subsets of the dataset of the same size such that $\mathcal{D} = \bigcup_k d_k$.

At each gradient step t , a new mini-batch is sampled without replacement. The update rule of the SGD is:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta}{\#d_k} \nabla_{\boldsymbol{\theta}} \sum_{(x_i, y_i) \in d_k} \mathcal{L}(f_{\boldsymbol{\theta}_{t-1}}(x_i), y_i),$$

where η is the descent step named *learning rate* in the deep learning community.

Evaluating the gradient over the whole dataset would be more accurate, but this is computationally costly. Moreover, since the loss landscape is highly non-convex, the optimization can get stuck at a local minimum. The SGD algorithm is computationally lighter and introduces noise in the gradient estimation, which prevents getting stuck in a local minimum. Many variants of this algorithm have been developed to reach faster convergence rates by introducing momentum or learning rate decay, such as the well-known ADAM optimizer[99]. The initialization of the weights has also been an important research topic, with various techniques such as the Gaussian initialization, the Xavier initialization[68] or the Kaiming initialization [78].

II.3.2 Convolutional neural networks

MLP are less suited to large input data like images, having a large memory footprint and being overparametrized. The weights vector $\boldsymbol{\theta}$ are indeed necessarily larger than the signal dimension, making MLP heavy in this context. Moreover, the structure of the MLP limits its use to data of a single size, which is problematic when dealing with size varying signals such as images.

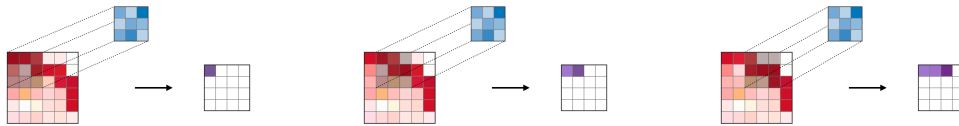


Figure II.10: Convolutional layer diagram

To better fit the image domain, neural networks based on convolutional kernels (CNN) were first introduced in [111]. A small filter slides over the whole image domain to create new feature maps, as shown in Fig. II.10. Formally, if $X \in \mathbb{R}^{H \times W \times C}$ is a 3D input feature tensor (typically an image), $g \in \mathbb{R}^{k \times k \times C}$ is a squared convolutional kernel and $b \in \mathbb{R}$ is a bias term, a single-channel output of a convolutional layer Y can be written as follows:

$$Y_{m,n} = \sigma \left[\sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{j=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{c=0}^{C-1} X_{m-i, n-j, c} g_{i,j,c} + b \right].$$

Being translation invariant, the convolution allows for detecting patterns at different locations in the input. Convolutional kernels are therefore well suited to images, which we recall are highly redundant signals. Moreover, in comparison with MLPs, they also drastically reduce the number of parameters between intermediate features. They are therefore more efficient computationally than MLPs because of their lighter memory footprint and faster convergence. The optimization framework also applies to CNNs, since convolution is a differentiable operation.

Other complementary operations are often required for a CNN to succeed. In what follows, we provide a succinct list of these *layers* and operations:

- **Padding:** as we can see in Fig. II.10, the size of an image necessarily shrinks when convolved with a filter, because of side effects. To remediate this problem, a simple operation is to pad the image or tensor with new pixels before the convolution. The usual padding technique adds zero-valued pixels. Other padding techniques are possible such as repeating the boundary pixels or mirroring the pixel values.
- **Pooling layers:** in order to compress the dimensionality of intermediate representations to output a single value in the case of binary classification tasks, *pooling layers* are frequently used. The *max-pooling* layer computes the maximal value in every $(2, 2, C)$ tensor, outputting a $(1, 1, C)$ tensor. Most successful classification CNNs such as ResNet [77] or VGG [172] use this technique. An alternative is to implement an *average pooling* layer, which outputs the average of every two by two windows. It was later shown that these layers are responsible for the loss of the shift-invariance property, which was the initial benefit of CNNs [204]. A solution to this problem is to implement a pooling operation that respects Nyquist Shannon subsampling conditions, by applying a low pass filter before pooling.
- **Batch Normalization:** To speed up the convergence of the optimization step, the hidden representations of a batch can be normalized. This technique introduced in [86] was first based on the idea of reduction of the internal covariate shift, which designates the changes in the distribution of the input of each layer between different mini-batches caused by the different sources of randomness during training. This typically slows down training. Batch normalization layers counter this by trying to impose a normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ to the output of every convolutional layer for every single mini-batch, where (μ_i, σ_i^2) are learnable parameters at the i -th layer. This drastically increases convergence speed.

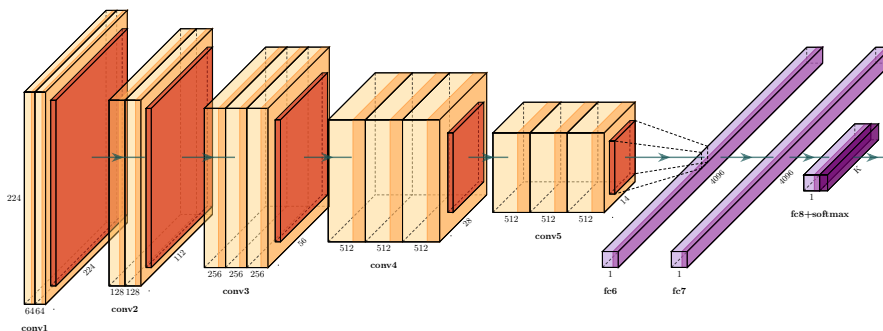


Figure II.11: Architecture of the VGG 16 network [172]

To give an example of a CNN architecture for a classification task, let us describe a simple architecture of a standard neural network, the VGG 16 network, represented in Fig. II.11. An image of size $(224, 224, 3)$ is first processed with two convolutional kernels of depth 64 with ReLU activations. The depth of a convolutional layer commonly defines the number

of feature maps created by the layer. To create 64 features starting from a tensor with C channels, a total of $64 \times C$ convolutional filters are required. The spatial dimension is the same because the feature maps were padded with zero-padding. The output of these layers is downsized with a max-pooling operation. Convolutional layers then process it with a twice as large depth. This sequence of operations is repeated four times until having a $(4, 4, 512)$ tensor. Finally, this tensor is flattened to a 4096 array. From this array, we apply a fully connected layer, another denomination for a perceptron, to reach a final vector of size 1000. It is indeed the number of classes in the classic Imagenet challenge [49]. By sequentially compressing the information, the network extracts semantical information about the original image to classify its content. In the following paragraph, we showcase some applications of CNNs.

II.3.3 Applications

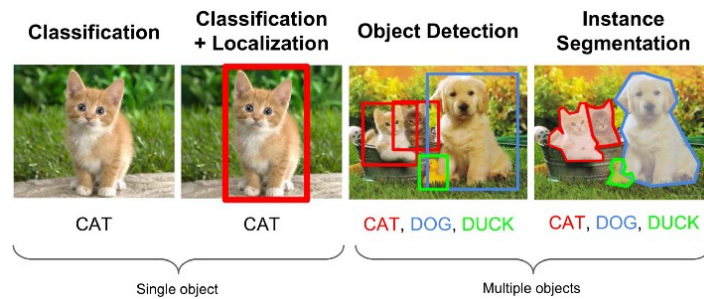


Figure II.12: Examples of computer vision tasks [147]

Computer vision applications CNNs are primarily used for image understanding and computer vision tasks. The simplest is image classification, where the goal is to map an image to a category based on its content. We already presented the VGG method, but other more recent and advanced techniques have been developed since then, with residual layers, [77], dense layers [83], attention mechanisms [180, 183], transformer networks [52], etc.

A more complex task consists of detecting objects in an image by finding and labeling a bounding box that delimits the region of interest. These methods were first based on a region proposal method combined with a classification method to refine the object detections. The region proposal step was first done with simple heuristics, but the latter were replaced afterwards by learning-based approaches. Among these methods, we can cite R-CNN [67], and its improvements Fast R-CNN [66], Faster R-CNN [159], or Mask R-CNN [75], which also does semantic segmentation.

Segmentation is an even more complex task: rather than finding bounding boxes, we now want to determine a precise boundary of the objects in the scene. The most commonly used network for image segmentation, the U-Net [164], was first developed for brain tumor segmentation. Its architecture was then used in several other applications, particularly for image restoration. A short description of this architecture is relevant to the present dissertation. The input image is passed through an encoder which compresses the spatial dimension of the representation of the images, as we can see in the diagram in Fig. II.13. Then, this hidden representation is passed through a decoder which progressively expands the spatial

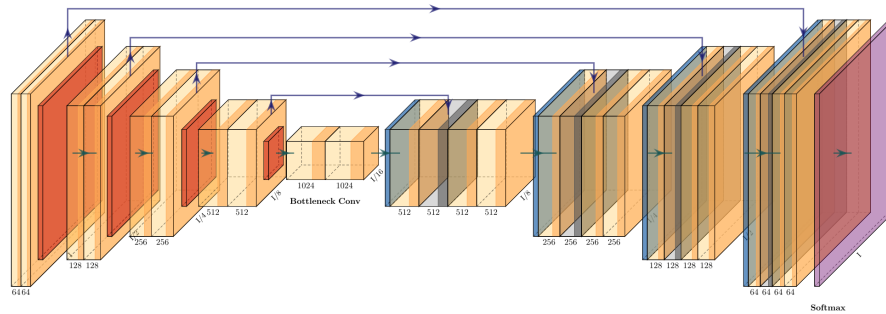


Figure II.13: Architecture of the Unet segmentation network [164]

dimensions to reach the original dimension of the input image finally. The particularity of this network is the so-called *skip-connections*, which allow for representations of the encoder to be concatenated to the representation of the same dimension in the decoder. This was added to fasten the convergence and recover information that the compression of the decoder could have lost. In the case of image segmentation, this network produces a mask of 0 and 1, where 1 denotes the presence of an object. Each image from the training set is annotated with a segmentation mask, either drawn by hand or computed from complex methods. Going a step further, we can segment different objects in a single scene, which is called semantic segmentation.



Figure II.14: Advances in facial image generation through time presented in [face_generation]

Image generation. Rather than using the expressive power of CNNs to extract high-level semantic information about the images, we can exploit it to generate new realistic or artistic images. The field of neural image generation has been very prolific since the introduction of Generative adversarial Networks (GANs) in 2014 [72]. The idea is to jointly learn a generator that produces a new image and a discriminator which assesses if the image is truthful or not. From the viewpoint of game theory, the problem can be seen as a non-cooperative game with two players (the generator and discriminator), and the solution should find a Nash

equilibrium. The latter reference has been the basis for further improvements leading to impressive results for high-resolution image generation techniques [156, 210]. Variational Auto Encoders [100] is another technique that allows for better control of the properties of the generated images. More recently, other techniques such as Normalizing Flows [160, 101] and Diffusion Models [82, 174] based on the use of neural networks produce state-of-the-art generation results.

These recent image generation techniques are capable of generating high-resolution realistic images, with astonishing precision. These methods can also be used to generate unrealistic artistic images with a wide variety of styles. Artistic image generation is indeed another application of deep neural networks. The seminal works for these methods were presented by Gatys et al. [62, 61]. The authors of these papers propose to extract the artistic style of a painting in order to apply it to a real photograph. We will describe these works more precisely in the following chapter. In few words, the proposed idea is to match the hidden representation of the style image (the painting) and the target image (the photograph) in an already trained classification network, such as VGG16. We give in Fig. II.15 some results of this style transfer methods presented by Gatys et al.

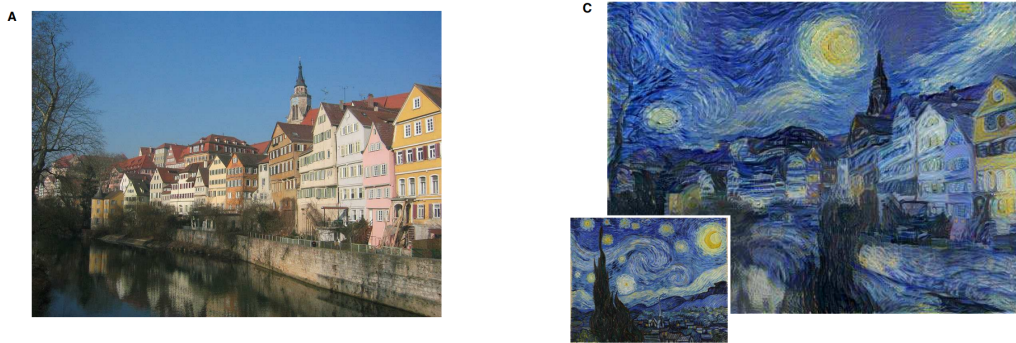


Figure II.15: Example of style transfer presented in [61].

Among other applications of deep learning, a major one is image restoration. Since CNNs are particularly well suited to analyze the content of images, it is reasonable to assume that they will be able to filter them properly, depending on the distortion model. In the following chapter, I will cover in detail the recent works in image restoration techniques based on neural networks.



Related works

III.1 Deep learning image restoration methods

As mentioned above, deep learning techniques have recently surpassed classical prior-based methods for most tasks. We give in this section a thorough presentation of these methods. We will first start by introducing the more standard deep learning approaches before showing some examples of alternative methods. Then, we will pursue our presentation by showing how architectural advances in deep learning resulted in improvements in image restoration. Additionally, we will see that the loss function has a significant role in the perceptual quality of the restored images. Finally, we will present how these methods behave in real-world scenarios and how they can be adapted to perform better in these conditions.

III.1.1 Standard CNNs for image restoration

DnCNN. The first paper tackling image denoising with a convolutional network was published in 2009 by Jain and Seung [89]. This first method did not perform very well, unlike the MLP method proposed later by Burger et al. [29], which performed on par with BM3D. A notable improvement was achieved in 2016 with the DnCNN paper by Zhang et al. [200]. The proposed architecture (see Fig. III.1) is, contrary to [89], very deep. The image is passed through a first convolutional layer of depth 64 with $(3, 3)$ filters and a ReLU activation. The output tensor is then passed through 17 blocks of a convolutional layer of depth 64, batch normalization, and ReLU. A final convolutional layer predicts the noise added to the input image. The predicted noise is then subtracted from the noisy image to predict a denoised image. This method, the first to use such a deep network for image restoration, was motivated by the success of deeper networks for computer vision tasks. Indeed, deeper networks have a larger *receptive field*, i.e., the number of pixels in the input image which contributes to the estimation of one pixel in the result image. It also adopts residual learning, that is predicting

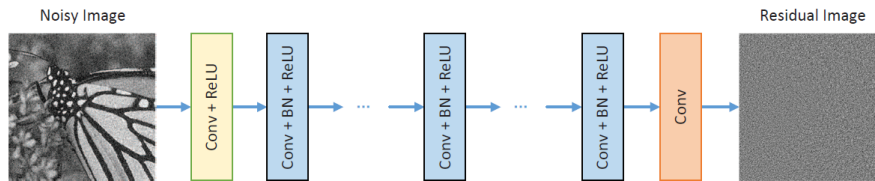


Figure III.1: Architecture of the denoising network DnCNN [200]

the noise instead of the output image. The optimization aims at minimizing the following:

$$\mathcal{L} = \sum_{i=1}^N \|x_i - y_i - f_{\theta}(y_i)\|_2^2,$$

where $(x_i, y_i)_{i \in [1, \dots, N]}$ is the dataset of clean and noisy images. It was later argued in [16] that residual learning allows for a simplification of the solution's manifold, which is easier to learn. This technique was also used in super-resolution CNNs [98]. In the case of single-image super-resolution (SISR), the low-resolution input is often interpolated to the desired output size, and the network's goal is to retrieve the missing high frequencies. Zhang et al. have also performed extensive experimental validation on color image denoising, jpeg quantization artifact removal, and SISR [200]. The network DnCNN can also do blind denoising, i.e. denoising without knowing the noise standard deviation. This network is however still unable to deal with real-world noise since it was trained with additive white Gaussian noise.

FFDNet. Going a step further, Zhang et al. also proposed the FFDNet network for image denoising [202]. Built on the DnCNN backbone, the authors propose two major modifications. First, a greyscale image is rearranged into four sub-images with an invertible transform to further increase the receptive field. Other techniques to increase the receptive field are possible such as dilated convolutions, which ignore some pixels. Second, in order to tackle spatially varying noise, a noise map is concatenated to the previously mentioned four sub-images. This map contains the noise standard deviation at every pixel. The resulting tensor is passed through a CNN backbone identical to DnCNN's, which outputs four denoised sub-images. Interestingly, the authors have not performed residual learning, but it was shown in [175] that the network performs better with residual learning. The network is trained to tackle various noise levels ranging in [5, 75] for an image coded in 8 bits. Experiments show that this network generalizes better to Poisson noise. This property is promising for real-world image denoising. However, this method is non-blind, meaning it requires knowing the spatial distribution of the noise level prior to denoise. The idea of the distortion map was used in other denoising networks [198], and adapted for super-resolution to encode different blur kernels [203].

Other architectures. The previous networks have a large number of parameters. Moreover, because the size of the intermediate representations is never shrunk, these networks have an extensive memory footprint, making them less fit to restore large images. In that perspective,

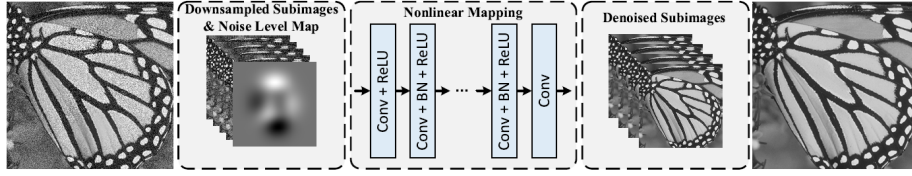


Figure III.2: Architecture of the denoising network FFDNet [202]

the U-Net architecture was first adapted for image restoration in the paper of Mao et al. [131] and adapted for Magnetic Resonance Imaging (MRI) image restoration in [80]. This network can be trained for multiple restoration tasks (denoising, SISR). The noise map idea of Zhang et al. was also used within this U-Net architecture in [198], achieving better results than FFDNet.

III.1.2 Alternative deep restoration methods

Self-supervised Learning. The methods described above are all learned in a supervised setting. However, ground truth data might not be available in real-world scenarii. In order to still leverage the capacity of neural networks to restore images, various semi-supervised or self-supervised methods have been developed. The most popular of these methods, Noise-2-Noise, was introduced in 2018 by Lehtinen et al. [116]. In a supervised setting, we want to find the optimal parameters:

$$\theta_{\text{supervised}}^* = \operatorname{argmin}_{\theta \in \Theta} \mathbb{E}_X (\mathbb{E}_{Y|X} (\|f_{\theta}(y) - x\|_2^2)),$$

where $x \sim X$ the distribution of clean images and $y \sim Y$ is its corresponding noisy observation. Rather than using pairs of distorted and clean images (x, y) during training to minimize the Euclidean distance (also called \mathcal{L}_2 loss), the authors use pairs of different noisy observations (y_1, y_2) which are drawn independently from the same random variable Y . The authors justify this choice by arguing that it is easier to obtain noisy observations of the same image instead of acquiring noiseless images. The optimization process now aims at finding:

$$\theta_{\text{self-supervised}}^* = \operatorname{argmin}_{\theta \in \Theta} \mathbb{E}_X (\mathbb{E}_{Y|X} (\|f_{\theta}(y_1) - y_2\|_2^2)),$$

Supposing that the noise process is centered, we have thanks to the independence of y_1 and y_2 that:

$$\begin{aligned} \mathbb{E}_{Y|X} (\|f_{\theta}(y_1) - y_2\|_2^2) &= \mathbb{E}_{Y|X} (\|f_{\theta}(y_1)\|_2^2) + \mathbb{E}_{Y|X} (\|y_2\|_2^2) \\ &\quad - 2\mathbb{E}_{Y|X} (y_2)^T \mathbb{E}_{Y|X} (f_{\theta}(y_1)) \end{aligned} \quad (\text{III.1})$$

$$\begin{aligned} &= \mathbb{E}_{Y|X} (\|f_{\theta}(y_1)\|_2^2) + \|x\|_2^2 - 2x^T \mathbb{E}_{Y|X} (f_{\theta}(y_1)) \\ &= \mathbb{E}_{Y|X} (\|f_{\theta}(y) - x\|_2^2) \end{aligned} \quad (\text{III.2})$$

From this equation, we understand that the minimization of the \mathcal{L}_2 loss should lead to the same solutions. The authors indeed show that supervised training and self-supervised training lead to identical performances for many noise models. Other losses might be more suitable for other noise models. For salt-and-pepper noise, for instance, the \mathcal{L}_1 loss is better since it preserves the median value of the original image.

Going a step further, self-supervised methods inspired by Noise-2-Noise propose to learn a denoiser without pairs of noisy images but with a dataset of single noisy observations. Krull et al. first proposed a simple self-supervised method called Noise-2-Void [103]. In order to use a single image as both a noisy observation and a reference image, the authors propose a modification of the training process so that the final prediction of a pixel depends on the neighboring values Ω_y but not the central pixel y . This pixel is used as a noisy reference value, just like in Noise-2-Noise. Exactly removing the pixel from the receptive field is cumbersome since it requires fully processing a patch and canceling all the gradient terms depending on that value. A simple approximation consists of replacing the central pixel value with a randomly picked pixel in its neighborhood. At test time, the denoising network does not use the central pixel value, resulting in a loss of information and poorer performances. Laine et al. propose a two-fold improvement of the Noise-2-void technique in [104]. First, the architecture of the network is modified so that it exactly excludes the central pixel of the receptive field. To do so, filters expand only in a single half-plane; for instance, if the filter size is h , the top $\lfloor (h/2) \rfloor$ can be fixed to zero. For simplicity, the image is concatenated with its 90° rotations so that filters are only masked in a single direction. Each feature map is then shifted downwards by one pixel in order to exclude the central pixel of the receptive field. In Fig. III.3, we show a diagram of the so-called blind spot architecture. Second, the method also leverages the information of the central pixel in a Bayesian denoising setting. Other blind spot architectures were also developed, such as Noise2Kernel [113], which propose another modification of filters that alleviates the need to rotate the image.

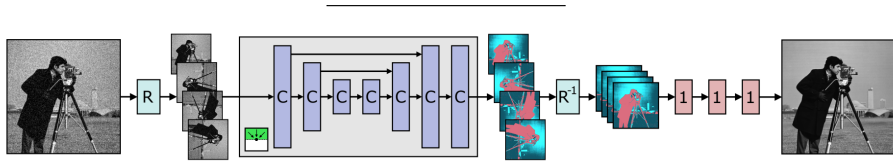


Figure III.3: Blind spot network architecture [104]

Plug-and-Play Methods. The methods presented above allow for computationally effective training without the need to acquire ground truth data, but with pairs or even single noisy images. In order to also reduce the computational cost while benefitting from the expressive power of neural networks, a class of hybrid methods called plug-and-play (PnP) use already trained denoising neural networks as a regularizer in a variational method set up (see Section II.2.3), for very different image restoration tasks.

This idea is based on variable splitting algorithms such as the Alternating Direction Method of Multipliers (ADMM) [21], which allows decoupling of the data term and the prior i.e. regularization term during the minimization. In [181, 79], it was shown that the prior term minimization corresponds to a denoising subproblem. Formally, the goal is to minimize an energy $\hat{x} = \operatorname{argmin}_x \|Ax - y\|_2^2 + \beta \mathcal{R}(x)$, where \mathcal{R} is the regularisation term and A the linear operator in the inverse problem formulation $y = Ax + n$. We can reformulate the minimization by artificially splitting the x variable into two variables x and v such that now

the problem becomes:

$$\begin{aligned} \hat{x}, \hat{v} &= \operatorname{argmin}_{x,v} \|Ax - y\|_2^2 + \beta \mathcal{R}(v) \\ &\text{subject to } x = v. \end{aligned} \quad (\text{III.3})$$

We can solve this minimization by forming the augmented Lagrangian and using the techniques derived in the ADMM paper. The Lagrangian writes as:

$$L_\lambda(x, v, u) = \|Ax - y\|_2^2 + \beta \mathcal{R}(v) + \frac{\lambda}{2} \|x - v + u\|_2^2 - \frac{\lambda}{2} \|u\|_2^2. \quad (\text{III.4})$$

The ADMM algorithm aims at iteratively refining each variable by minimizing the Lagrangian with respect to a single variable at each step for x and v and performing a gradient ascent for u . More precisely, we perform the following steps until convergence:

$$\begin{aligned} \hat{x} &\leftarrow \operatorname{argmin}_x L_\lambda(x, \hat{v}, u) \\ \hat{v} &\leftarrow \operatorname{argmin}_v L_\lambda(\hat{x}, v, u) \\ u &\leftarrow u + (\hat{x} - \hat{v}). \end{aligned} \quad (\text{III.5})$$

If we define $\tilde{x} = \hat{v} - u$ and $\tilde{v} = \hat{x} + u$, the former minimization can be rewritten as:

$$\begin{aligned} \hat{x} &\leftarrow \operatorname{argmin}_x \|Ax - y\|_2^2 + \frac{\lambda}{2} \|x - \tilde{x}\|_2^2 \\ \hat{v} &\leftarrow \operatorname{argmin}_v \frac{1}{2} \|\tilde{v} - v\|_2^2 + \frac{\beta}{\lambda} \mathcal{R}(v) \\ u &\leftarrow u + (\hat{x} - \hat{v}). \end{aligned} \quad (\text{III.6})$$

The first step returns a Maximum A Posteriori (MAP) estimate of x given y with a simple quadratic regularisation. In turn, the MAP estimate in the second step corresponds to denoising the variable \tilde{v} , as it tries to minimize a function obtained by summing the squared euclidian distance between v and \tilde{v} to a term that penalizes the singularities of v . PnP methods derived from the ADMM algorithm replace this step by applying a denoising method of their choice. Before the advent of deep learning techniques for image restoration, PnP methods were based on human-designed denoising methods such as BM3D [46]. Because they allow to tackle all linear inverse problems, BM3D was used as a prior for super-resolution tasks [35]. Other variational restoration algorithms have been modified by incorporating denoisers, such as half-quadratic splitting [79, 198], or stochastic gradient descent [107]. These classical methods were then modified by incorporating denoising CNNs which achieved better results in terms of pure image denoising. This allowed for better restoration in PnP methods, which were applied to all kinds of restoration tasks, for example, inpainting [118], super-resolution [201], or global video restoration [136].

III.1.3 Non-local and attention based architecture for image restoration

For most computer vision tasks, the performances suffer from the local nature of convolutions. While information at distant positions in the image might be relevant for prediction, it is often left unused because the receptive field is limited. Many works propose to increase the receptive field with strided convolutions, dilated convolutions etc. This may increase the

receptive field, but the relative importance of a pixel j for the computation of a pixel i still decreases significantly as j gets far from i . This is one of the major flaws of classical CNNs in computer vision applications and especially for image restoration. Inspired by Wang et al. [183], which propose non-local neural networks, many subsequent works propose to modify the architecture of CNNs to exploit the self-similarity in images, following the intuition of non-local methods for image restoration discussed in Section II.2.2. These non-local neural methods were also adapted to image restoration with different levels of complexity. The simplest hybrid methods decouple non-local filtering and neural network filtering. Then, non-local inference CNNs which unroll classical non-local denoising algorithms into a CNN with block-matching methods. Going a step further, CNN architectures include deeper non-local layers or architecture adapted from text or graph processing. In the present section, we will successively present each of these techniques.

Decoupled hybrid restoration methods. Decoupled non-local neural methods are denoising algorithms that separate non-local filtering and neural filtering. Therefore, we can use any existing CNN architecture in these approaches. The major benefits of these methods are versatility and ease of implementation.

Ahn et al. first presented a Block Matching CNN for image denoising [9]. Inspired by BM3D [42], the method can be split into a block-matching algorithm and a denoising algorithm. For each pixel, the block-matching algorithm selects the k Nearest-Neighbors (kNN) patches in a defined search window, using the euclidean distance as a similarity metric. However, the similarity is not evaluated on the noisy image but on a pilot image, obtained thanks to an already existing denoising method: BM3D or DnCNN in the paper. Computing the block-matching on the noisy image indeed leads to poor results. As explained in [9], the variance of the distance between two patches grows in $O(\sigma^4)$, where σ is the noise standard deviation. Therefore, two similar clean patches may be very different when distorted with noise. Once the kNN are found, the authors concatenate both the noisy and the pilot patches in a 3D tensor of size $(2k, N_{patch}, N_{patch})$, that is further passed through a regular and trainable CNN similar to DnCNN, which outputs a single patch. Experimentally, the paper shows that this method outperforms DnCNN by 0.1 or 0.2 dB on the Set12 database, with notable improvements in images with strong self-similarity. However, the computation time is much longer because of the nearest neighbor search.

Similarly, Davy et al. [47] present a non-local decoupled method for video denoising. The method can also be divided into pixel matching and CNN filtering. The difference is that the search window is now three-dimensional, including the temporal dimensionality. Instead of stacking patches together, they stack the noisy frame t with its N nearest neighbors feature maps. Therefore the N most similar pixels to pixel (i, t) are stacked along the z -axis at the same position i . Those $N+1$ feature maps are then passed through a CNN inspired by DnCNN.

Whereas the previous methods were two-staged, Cruz et al. [41] introduced an iterative non-local decoupled algorithm. The algorithm alternates CNN and non-local filtering on a convex combination of the noisy image and the denoising result obtained at the previous iteration. The underlying idea is to gradually mitigate the artifacts created by the CNN in order not to over smooth the resulting image.

This algorithm, inspired by the BM3D algorithm, allows for significant improvements when tested on three already-existing denoising CNNs: DnCNN[200], FFDNET [202], and WDNnCNN(Wavelet DnCNN) [16]. This method is similar to Plug-and-Play methods since it is agnostic of the chosen denoiser. Its scope is, however, limited to image denoising.

Deep unrolling of non-local algorithms. The papers presented next are CNN adaptations of existing non-local denoising methods. These adaptations can either unroll a few iterations of an optimization process or sparse dictionary-learning methods. The first paper [154] is a non-local adaptation of the trainable nonlinear reaction-diffusion (TNRD) [40]. The TNRD is a trainable variational method. Its goal is to minimize an energy functional $E(u|f)$ with Field of Experts (FoE) regularizers, where f is the noisy image and u is the output image. The energy is described by the following formula:

$$E(u|f) = \frac{\lambda}{2} \|u - f\|_2^2 + \sum_{i=1}^{N_k} \rho_i(k_i * u),$$

where the regularizers are defined by convolutional filters k_i , and non-linear penalty functions ρ_i . Minimizing this energy by steepest gradient descent leads to a denoised image. The update rule is given by:

$$u_t = u_{t-1} - \delta_t \times \lambda_t (u_{t-1} - f) + \sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1}),$$

where $\phi_i^t = \rho_i^t$. If we truncate the descent of the TNRD algorithm after T iterations, it is equivalent to a multi-layer neural-network-like method that can be formulated as

$$\begin{cases} u_0 = f, \\ u_t = u_{t-1} - (\lambda_t(u_{t-1} - f) + \sum_{i=1}^{N_k} \bar{k}_i^t * \phi_i^t(k_i^t * u_{t-1})). \end{cases}$$

The nonlinear functions were chosen to be parametrized radial basis functions, making the obtained network fully differentiable with respect to all of its parameters $\{\Theta_t : [\lambda_t, k_i^t, \phi_i^t], t \in (1..T)\}$. Therefore, having a database of clean and noisy images, one can optimize this network with a classical back-propagation. The only difference between this method and its non-local adaptation is that the authors add a non-local operator which groups the k-NN of each pixel before applying the convolutional filters k_i^t . In order to include the non-local operator easily in this framework, the authors express it as a sparse lookup matrix. Experimentally, the non-locality improves the initial TNRD both visually and numerically. However, it cannot compete with fully learnable CNNs. Even if this framework helps to better understand the network's behavior, the class of functions that can be coded by this network is over-constrained.

We can also mention the works of Lefkimmiatis [114],[115], which introduce another non-local denoising inference CNN. Similar to [154], Lefkimmiatis proposes a non-local block-matching operation. Following this, a trainable neural network applies a sequence of convolutional blocks that mimic a proximal gradient descent. This method performs on par with DnCNN and performs even better when an oracle is used for block-matching. Other papers also unroll already existing non-local methods into CNNs, such as BM3D-Net

[192], which unrolls BM3D, or [110], where a differentiable relaxation of LSSC [129] is implemented. The latter is on par with other fully-learned CNNs in terms of PSNR, with a much lower number of parameters.

Deep non local mechanisms. Unlike previously presented methods, the most common approach when developing new CNN architectures is to sequentially combine classical convolutional layers with other recently proposed layers to successively refine feature representations. These *black box* methods are trained in an end-to-end manner, making them hard to interpret. In order to impose a non-local behavior on neural networks, a reasonable idea is to create trainable non-local layers and alternate them with classical convolutional layers in an end-to-end manner. Unlike decoupled methods, non-local filtering is performed for deeper features.

The article *Neural Nearest Neighbor Networks* [150] introduces a new type of non-local layer. Inspired by [69], they first introduce a stochastic and continuous relaxation of the K-NN search. However, the latter is a non-differentiable operation and is consequently non-trainable. Therefore, it cannot be included in a standard neural network, justifying the need for its relaxation. This relaxed kNN is then implemented as a differentiable layer, which can be interleaved in a classical CNN. The authors show very promising denoising results with a non-local adaptation of DnCNN’s architecture. In a few words, the network alternates convolutional blocks with a relaxed kNN. The kNN allows us to stack similar values together so that the following convolutional block can filter them collaboratively. This method outperforms FFDNET and DnCNN in terms of PSNR on all noise levels on the following datasets: Set12, CBSD68, and Urban100.

Unlike the previous method, Liu et al. [122] introduce a non-local layer similar to the one implemented in [183] and embed it in a recurrent neural network. The non-local layer applies a soft block-matching over a defined neighborhood, i.e., they compute a weighted aggregation of all the pixels in the neighborhood, of which the weights depend on a similarity measure between the pixels. This method is similar to Non-Local Means, except that the aggregation is done on transformed features with trainable parameters. The authors tried different similarity measures. They found that the best one is the embedded Gaussian metric D , which can be formulated as $D(X_i, X_j) = \exp(X_i W_\theta W_\psi^T X_j^T)$, where (W_θ, W_ψ) are trainable parameters and (X_i, X_j) are the compared patches. The output Z of the block-matching becomes:

$$Z_i = \frac{1}{\delta(X_i)} \sum_{j \in N_i} \exp(X_i W_\theta W_\psi^T X_j^T) X_j W_g,$$

where W_g is a learnable embedding matrix, and $\delta(X_i)$ is a normalizing factor.

This module is summed up in Fig. III.4a. It is then used in the recurrent network block presented in Fig. III.4b. As we can see, the non-local module produces non-local features, which are passed through two convolutional layers before being added to the initial input of the recurrent network. This skipped connection eases the back-propagation of the gradient. The recurrent network loops T times over itself, and its output is then passed through a convolutional layer that produces the residual image, i.e., the estimated noise map.

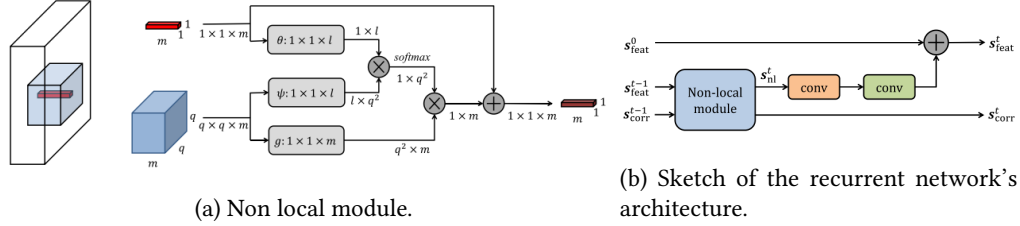


Figure III.4: Architecture and non local module of NLRN [122].

One can also conceive images as a graph where nodes are the pixels linked to their neighboring pixels by edges. This is the underlying idea exploited in the recent work of Valsesia et al. [178]. This paper is an improvement of [179] since it provides a better architecture and a more thorough analysis of the network's behavior. Valsesia et al. consider the image (or the deeper features) as a graph where the nodes are the pixels, and the edges are weighted by a similarity function between the two pixels. To fit this paradigm, the authors define a convolution operation called Edge-Conditioned Convolution (ECC) that allows the network to have a larger and more useful receptive field, thanks to the non-locality property of the ECC. The architecture of the network is a two-staged network, which predicts the residual noise. The first stage of the network consists of extracting local features at different scales with three parallel branches of convolutional layers. At the end of each branch, non-local features are extracted thanks to the non-local ECC layer. After concatenation, these features undergo a sequence of local and non-local operations.

Experimentally, the proposed method performs slightly better than NLRN. Both methods indeed outperform DnCNN by 0.3 dB on Set12, 0.2 dB on BSD68, and 1.2 dB on Urban 100. Interestingly, the authors provide a visualization (see Fig. III.5) of the non-local receptive field of the network for a chosen pixel, and how it expands through the network.

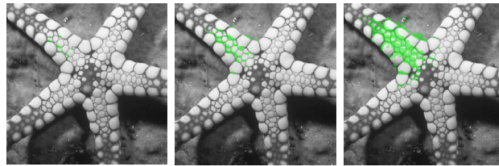


Figure III.5: Receptive field of an edge pixel at different depth of the network DGCN [178].

Self Attention mechanisms for image restoration Transformers. Inspired by Transformer networks which were first developed for natural language processing [180], Dosovitskiy et al. first presented Visual Transformer [52], implementing the Transformer architecture for images. Originally, Transformer networks aimed at leveraging long-term dependencies in sentences, where words spatially distant in the sentence are semantically linked. This is performed by the Multi-Head attention Layer. Each word in a given sequence is mapped to a Euclidian space through a learned embedding which encodes similar words closely, forming a matrix $Z \in \mathbb{R}^{n,d}$, where n is the length of the sequence and d the dimension of the

embedding space. This matrix is passed through a positional encoder which adds the word's position information to the embedding vector. The resulting matrix is multiplied by three learnable matrices W_Q, W_K, W_V to form a query matrix $Q \in \mathbb{R}^{n, d_k}$, a key matrix $K \in \mathbb{R}^{n, d_k}$, and a value matrix $V \in \mathbb{R}^{n, d_v}$. These matrices are abstract structures that are going to be combined to form a new representation of the sequence, which contains information from all the words in the sequence:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

The matrix $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{n, n}$ can be interpreted as an attention map giving more weight to words which are highly dependent from each other. This weight matrix helps to combine the values corresponding to each word V . In order to even better exploit these long-term dependencies, the (Q, K, V) triplets undergo different learned linear projections. The attention mechanism is deployed for each projection, and the results are combined with a learned weighted average.

This attention mechanism is the same Dosovitskiy et al. used for Visual transformers [52]. The authors of this paper consider that the patches that compose the images can be seen as words interacting together as in a sentence. The attention block is inserted in a classification network that outperforms ResNet on all tasks.

Building on this idea, Visual Transformers were first adapted to image restoration tasks in [37], which first encodes the image with a convolutional network, then uses a Visual Transformer block as presented above (based on the idea of patches being similar to words), and finally uses another convolutional block to go back to the final image. This method was developed for multiple tasks, denoising, super-resolution, deraining, etc. The improvements are significant for self-similar images, which are very present in urban scenes, a behavior already observed with non-local networks. However, the idea of only using patches over a predefined grid as words is limiting. Similar patterns can indeed expand across different patches. However, classical Virtual Transformers cannot fully leverage this similarity. In an attempt to both use the local behavior of convolutions and the non-local behavior of Transformers, the SWIN transformer [124] was used for image restoration problems [119, 185]. The SWIN transformer allows to perform self-attention at different scales iteratively, by dividing the image into patches of several different sizes. It also restricts the span of the attention layer to a fixed-sized window. This window shifts at every attention layer, allowing connections that were impossible before. For both super-resolution and image denoising, the SWINIR transformer [119] achieves state-of-the-art results on standard datasets.

III.1.4 The perceptual impact of loss functions

Deep learning methods rely on three pillars: data, the network architecture, and the loss function. We have seen in the previous sections how the network architecture can influence the results of image restoration results. In this section, we will present the recent advances in the design of loss functions that correlate with human perception.

Classical perceptual metrics. Although the standard choice for training an image restoration network is the \mathcal{L}_2 distance, it was shown that this loss does not correlate well with the human perception of distortions [65, 186]. When it comes to image restoration, minimizing the MSE tends toward the average of all possible solutions. If this is a desired property for the restoration of smooth images, it is not the case for high frequency content such as edges or micro-textures. Since the distribution of micro-textures is often close to a random noise, averaging all possible solutions will necessarily result in a blurred output.

To better fit human perception, many metrics were developed. In order to compute a perceptual distance, an initial idea was to project both the reference and distorted image in a perceptual space, where the Euclidean distance can be employed [188, 43]. These models derive from the analysis of the physiological response of the human visual system to different visual stimuli. These models often include a multi-scale decomposition of the image. Even though the rationale for these methods is convincing, they often require complex parametrization, which is hard to fit to new data.

Unlike the previous methods, the widely used Structural Similarity Metric (SSIM) does not map the image to a perceptual space. Its computation includes the use of local statistics such as the mean μ and standard deviation σ :

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_x\sigma_y + c_2)(\text{Cov}_{xy} + c_3)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_1)(\sigma_x\sigma_y + c_3)},$$

where (c_1, c_2, c_3) are small stabilizing parameters which are useful when the denominator is small. This metric was designed to be invariant with respect to distortions such as contrast change or illuminance change, which do not affect our perception but degrade the \mathcal{L}_2 metric. Its multi-scale variant, MS-SSIM, is most commonly used since it allows to handle features of varying sizes. Even if this metric allows us to better assess the structural distortions of an image, its formula does not correspond to any physiological prior about the visual system. It was indeed shown by Nilsson et al. [142] that the SSIM does not at all fit the human perception.

Laparra et al. introduced a Normalized Laplacian Distance fitted to human perception [106]. The metric uses a laplacian pyramid decomposition of the image, which decomposes the image at different levels of detail. In order to better fit human perception, each stage of the pyramid undergoes a contrast normalization with learned weights. The weights were optimized over a large dataset of undistorted images. The authors give experimental validation for this metric, showing that the metric linearly increases as the distortion level increases for a wide variety of distortions (blur, jpeg artifacts, etc.). Although this method righteously assesses the perceptual distance, no image restoration networks have ever used it. The MS-SSIM remains a commonly used loss for image restoration or image generation.

Deep CNN features as a perceptual space. Instead of mapping images to a human-designed perceptual space, some recent perceptual metrics propose to use a learned perceptual space. These methods all proceed from the seminal work of Gatys et al., which used features of a classification network for texture generation, and style transfer [62, 61]. Even if the VGG19 network (introduced in Section II.3) is not specifically designed to fit human perception, its architecture extracts information at different scales, similar to the Laplacian

Pyramid. To generate new texture images, the method aims at minimizing the distance between the Gram matrices of VGG19 features at different scales of the original image and the generated image, by a simple gradient descent. The Gram matrix transform was introduced to remove the spatial information of this representation in order to have stationary textures. The Gram matrix of a family of vector $u(1), \dots, u(m)$ is a squared matrix of size (m, m) of which every coefficient in (i, j) is equal to $\langle u(i), u(j) \rangle$ where $\langle \cdot, \cdot \rangle$ is a scalar product.

Having $(F_i)_{i \in \{1,5\}}$ 5 intermediate feature extracting functions of the VGG19 each at different scales, the texture distance can be written as:

$$L_{\text{texture}}(x, y) = \sum_{i=1}^5 \lambda_i \|G(F_i(x)) - G(F_i(y))\|_2,$$

where λ_i are weighting coefficients, and G corresponds to the Gram matrix transform function. To add back spatial information, this transform can be removed:

$$L_{\text{feature}}(x, y) = \sum_{i=1}^5 \lambda_i \|F_i(x) - F_i(y)\|_2.$$

For instance, this was done for style transfer which requires conserving structural information. This loss proved to be very efficient for both tasks, confirming the intuition that the VGG network mimics visual perception.

It was first used to train a neural network for fast style transfer, and SISR by Johnson et al. [92]. By combining it with other data fidelity losses, promising results were obtained. The SISR network indeed produced sharper images than other methods but included heavy checkerboard artifacts. The Enhancenet paper proposed to refine the perceptual loss for image super-resolution applications [168]. It indeed added the texture loss L_{texture} computed per patch, and a new adversarial term, leveraging the power of GANs for realistic image generation. This network produced significantly more realistic super-resolution results than the previous. To better exploit the fact that the different stages of the VGG network extract different structural information, Rad et al. proposed to decouple the VGG loss according to the content by segmenting the image in object, background, and borders [155]. For each region, the perceptual loss was computed from different layers of the VGG network, thus utilizing the semantic information to better restore the image.

The previous methods allowed networks to improve their performances. However, they do not take advantage of any information about the human visual system. Next, we present perceptual metrics optimized on annotated databases which exploit both the features of CNNs and the response of the human system to different distortions.

Learned Perceptual Metrics. In 2018, Zhang et al. presented a Learned Perceptual Image Patch Similarity (LPIPS) [205]. The paper first presents a large dataset of annotated data comprising multiple distortions at different intensity levels. The annotators chose between a pair of distorted images (y_1, y_2) the one closest to a reference image x that they see, resulting in an average vote h . From the set (x, y_1, y_2, h) , a first network such as VGG is used to compute a distance d_1 between x and y_1 and a distance d_2 between x and y_2 , similar to the $\mathcal{L}_{\text{feat}}$ presented previously. A second short network then predicts from (d_1, d_2) which

image is the closest to the reference by computing the cross entropy loss with respect to the annotation h . The authors show that the pre-trained VGG network achieves the best results in terms of perceptual evaluation. Since it is a standard neural network, it can be used as a metric for training. It was indeed used to train super-resolution neural networks [91].

Concurrently, Prashnani et al. presented a similar metric called PieAPP [152]. The data annotation method is identical, and so is the strategy. The only difference lies in the architecture specifically created for this task. The network has a large memory footprint making the metric unfit for the training of an image restoration network. We will use this metric in some of the experiments of the following chapters.

Advances in the design of the architecture and the loss functions greatly improved the perceptual results of image restoration neural networks. However, even such advanced methods fail to generalize to real-world conditions when trained on unrealistic data. Next, we will understand how real data, or carefully modelled synthetic data allows to properly restore images in real world conditions.

III.1.5 Real-world image restoration

One of the main drawbacks of neural networks is their incapacity to generalize to unseen configurations. In classical deep-learning tasks, this phenomenon is known as overfitting. It is possible to counter this with various methods, such as network regularisation or domain adaptation. The latter usually expects to have access to data from the new domain or modality. These drawbacks are inevitably present in deep learning applied to image restoration. As shown by Plötz et al. [149], a denoising model trained with an additive white Gaussian noise prior on the noise distribution fails to properly restore real noise-corrupted images, when a classical blind denoiser such as BM3D reaches even better performances. We describe here the state-of-the-art works that aim at countering this phenomenon. The first solution is to acquire real data and to train a neural network with it. If this technique allows retrieving real distortions, it is cumbersome. An alternative is to simulate distortions with a physically based model. More recent papers propose to learn from real data how to generate noise and other possible distortions.

Training with real data. In order to restore images corrupted with real-world distortions, the first idea that comes to mind is to acquire such data. In order to fit the classical supervised setting, data must come in pairs of distorted observations and ground truth noiseless images. However, noise is inherent to photon acquisition, and motion either of the camera or in the scene is difficult to control. For these reasons, acquiring such a dataset is a very tedious task. Plötz et al. first proposed a dataset of pairs of RAW images [149] called DND. In this paper, the approach consists of pairing higher ISOs images y_n and lower ISOs images y_r as noisy and reference images respectively. Noise intensity increases as the sensitivity (ISO) increases. The ISO can indeed be seen as a digital gain K_{ISO} , which increases linearly. For instance, we have the relationship $K_{200} = 2K_{100}$. In order to have image with the same pixel intensity, the reference image is shot with an exposure time multiplied by the ratio of

the sensitivities. To avoid motion, the camera is mounted on a sturdy tripod with weights to further stabilize the setup. Even with such precautions, the images are not perfectly aligned and many postprocessing operations are applied for the ground truth to fit to the noisy observation content. In addition to alignment, the authors also correct illuminance changes which are frequent when taking outdoor images.

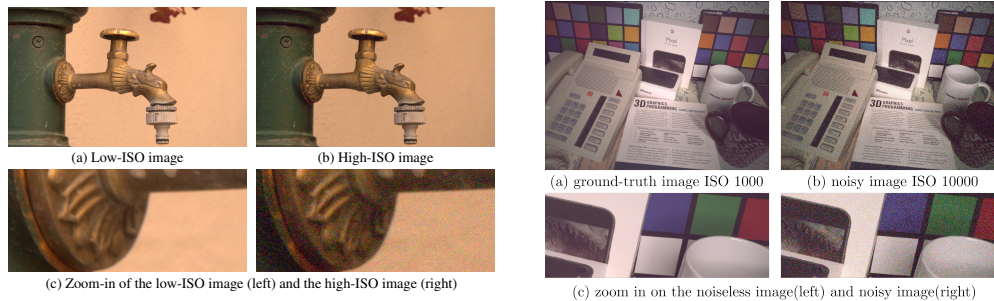


Figure III.6: Examples of the two denoising datasets DND [149] (left) and SIDD [2] (right). Images are displayed in RGB for visualisation purposes.

Similarly, Abdelhamed et al. present a large dataset of RAW smartphone images from five different smartphone cameras called SIDD [2], as an improvement of the previous dataset. Rather than lowering the ISOs to have a seemingly noiseless image, the authors decide to robustly combine a stack of 150 shots of the same scene, thus dividing the noise standard deviation by 150. To form this ground truth image, the images undergo many processing steps in order to remove dead pixels, to align the intensity, and align the images spatially. The image acquisition is even more controlled since the images are shot in a studio. To account for real-world situations, the dataset covers a much wider range of ISOs for every camera and a wide range of illuminant intensity and temperature. Both these datasets were widely used to train real RAW image restoration denoising networks [123, 197, 13, 195, 185]. It is indeed much simpler to tackle denoising on a RAW image, given that noise is spatially uncorrelated in the RAW domain, unlike in developed RGB images.

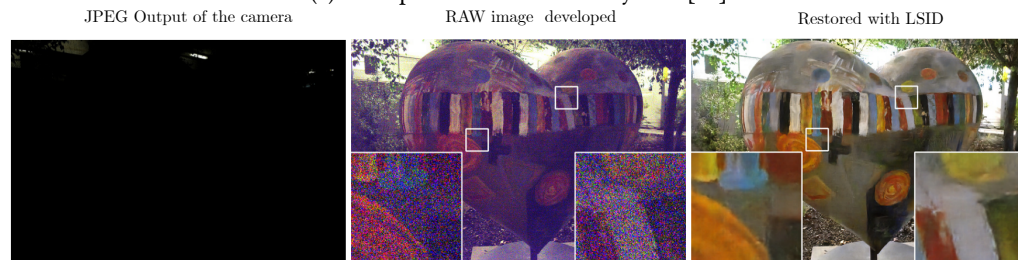
Nonetheless, a new trendy inverse problem in the image processing community is to fully replace the classical Image Signal Processing (ISP) pipeline with CNNs. The ISP presented in Section II.1.1 allows processing a RAW image into a viewable RGB image. It was indeed shown that individual tasks within the ISP can be efficiently replaced by CNNs such as automatic white-balancing [7], or demosaicking [64]. The goal is to now jointly learn the whole processing pipeline. To do so, one also needs a large dataset of pairs of (RAW - developed RGB) images. This type of dataset can be easily obtained since most cameras can save both the RAW and the RGB output of the ISP. A more advanced goal is to propose a cross-camera ISP model, meaning architecture that could properly develop RAW images from different cameras. Different camera sensors indeed have different color filters and different development algorithms, making the RAW domain camera-specific. This problem was addressed by Ignatov et al. [85], which presents a dataset of RAW-RGB pairs for a high-end DSLR camera and a smartphone camera. The presented method allows to produce DSLR looking images starting from smartphone-shot RAW images (see examples in Fig. III.7). The same approach is followed for super-resolution problems in [191]. Similarly, Zhang et al. perform RAW

super-resolution, with pairs of images taken with different focal lengths in [206], allowing for realistic modeling of a zooming operation.

Another more advanced problem is to propose an ISP that is able to jointly denoise, correct the exposure, and properly develop a low-light short-exposure image. This problem was first addressed in the paper *Learning to see in the Dark* [36]. In this paper, the authors also propose a dataset of real distorted images, where reference images are shot with long exposure (10s or 30s) and noisy observations with a short exposure (1/10s, 1/25s, 1/30s) with the same ISO. The long-exposure image is supposed to be noiseless, but it is very prone to motion. Even if the dataset was acquired in highly controlled conditions, the reference images are sometimes blurry for outdoor photographs. Despite these limitations in the image dataset, the learned network (an Unet-like architecture) produces surprisingly good-looking images. Similarly, the DeepISP paper [170] also supplies both a dataset and a network to jointly restore and develop low-light smartphone images.



(a) Example of the Results of PyNET [85]



(b) Example of the Results of Learning to See in the Dark [38]

Figure III.7: Examples of ISP replacement by neural networks

Even if acquiring real data allows us to learn the true restoration for a distortion model, it is really cumbersome. Moreover, the learned model does not necessarily generalize to new acquisition systems that were not present in the training dataset. In order to circumvent the burden of data acquisition, we present next methods that aim at realistically mimicking real-world distortions.

Physically based distortion models. As explained in Section II.1.1, an image undergoes many distortions during the acquisition and its processing. Simple distortion models such as Gaussian noise and Gaussian blur do not fit real-world conditions, and CNNs trained based on this assumption fail to restore real data. That is why carefully modeling the distortion process is required. When it comes to noise, a standard approximation of the shot noise and the read noise in the RAW domain is a *heteroscedastic* Poisson-Gaussian model [57]. From the law of large numbers, a Poisson law can indeed be approximated with a Gaussian law. In

theory, a noisy image y follows: $y \sim \mathcal{N}(x, \lambda \cdot x + \sigma^2)$ where λ, σ both depend on the ISO of the sensor. These noise parameters are sometimes given by the camera manufacturer or can be estimated with quite simple procedures. This approximation is widely employed to model raw noise and has led to quite good models for image denoising, see for instance [74, 24, 196, 150, 54], and many more references. Another interesting strategy is to apply a variance stabilizing transformation to the image, such as the Anscombe transform [12], which makes the noise quasi-signal-independent. In this set-up, it is possible to train a denoising network with additive white Gaussian noise in order to properly denoise real noisy images, see for example [184, 209]. The results are still not as good as with properly modeled noise.

Having such a noise model alleviates the need to acquire noisy-clean pairs of RAW images. The general practice is to add a synthetic noise to RAW images and train a network to minimize the \mathcal{L}_1 distance between the RAW image and the network’s output. For RAW images, the 0-1 clipping implies an offset in the small values, making the noise non-centered. This operation does not affect the median value, which is why it is chosen as the objective statistics. Many datasets of single RAW images are available such as the Adobe 5K dataset [31], the RAISE dataset [45], and all the previously mentioned paired datasets.

Some papers show that RAW images are not necessary to train good RAW image denoisers. Specifically, Brooks et al. [24] train a RAW image denoiser with a dataset of RGB images. They propose an approximate unprocessing algorithm, which inverts every standard step of a camera ISP to an artificial RAW domain. Noise is then added following the Poisson Gaussian approximation. In order to account for different cameras and imaging conditions, the authors generate noise with randomly sampled parameters, following an estimated distribution of noise parameters on the DND dataset. This random parameters sampling strategy is also applied for inverting white balance and other steps of the ISP. With synthetic RAW and noisy synthetic images, the authors could train a RAW-to-RAW denoising network with performances competing with state-of-the-art methods. Similarly, CycleISP[196] proposes to learn an RGB-2-RAW mapping instead of explicitly inverting the ISP step by step.

The noise model presented above is still an approximation of the real noise. When dealing with more difficult tasks such as low-light image enhancement, the slightest error in the distortion model has dramatic effects. This is why Wei et al. [190] proposed a more advanced RAW noise model. The goal is to solve the *Learning to see in the dark*[36] problem without pairs of real images. Having a long exposure clean image X , the short exposure noisy image Y is modeled with the following formula:

$$Y = \frac{X}{\gamma} + N_{shot} \left(\frac{X}{\gamma} \right) + N_{read} + N_{band} + N_{quant},$$

γ being the time exposure ratio. The shot noise N_{shot} is modeled by means of an exact Poisson distribution since the law of large numbers does not apply when dealing with extremely low-light images. The read noise N_{read} is modeled by means of an uncentered Tukey-Lambda distribution (a more heavy-tailed distribution than the Gaussian). The authors also include a banding pattern noise N_{band} , which appears in low light, adding a random Gaussian offset to each sensor line. All of these noises are parametrized depending on the ISO. The paper also presents an interesting noise parameters estimation procedure, supposing to have access to the camera. However, the authors do not provide the estimated parameters, which prevents one from reproducing their results. This paper shows impres-

sive results, and the authors claim to outperform the network trained with real noisy/clean image pairs.

A key takeaway of the literature is that the more realistic distortions a network are trained for, the better it generalizes to unseen distortions [199, 203, 202]. Next, we will see that it is possible to learn realistic distortion without a physical prior.

Learning the distortion process. Rather than explicitly modeling the distortion with physical priors, some authors propose to learn the model. Chen et al. were the first to do so [38], using a GAN as a realistic RGB noise generator. This paper proposes to learn a noise generator based on a dataset of noise patches extracted from a dataset of real noisy images. The network takes as input a Gaussian noise patch and outputs a supposed realistic noise patch. This noise patch is then added to an image to form noisy/clean pairs to train a denoising network similar to DnCNN. Even though the results are good, this method is highly questionable: first, the noise extraction method used only extracts noise patches from homogeneous images; second, the authors suppose that noise is signal-independent, which is clearly not the case. As an improvement to this technique, Kim et al. [97] righteously propose to condition the noise generator with the clean patch and other metadata such as the ISO, the shutter speed, etc. This makes the noise generator signal dependent. In contrast with the previously mentioned reference [38], this method requires to have access to a dataset of real noisy/clean pairs. Jang et al. propose a very similar approach but train the GAN with a Wasserstein Loss, which allows the noise distributions to be more accurately modeled [90]. Instead of using GANs as a noise generator, Abdelhamed et al. propose to use Conditional Normalizing Flows in the NoiseFlow paper [1]. This model is conditioned upon even more metadata, such as the camera constructor and the digital gain. It is also quite compact, having less than 2500 parameters, but still produces very realistic noise. The authors indeed show that the Kullback-Leibler divergence between generated noise and real noise is reduced by modeling noise with NoiseFlow instead of using a Poisson Gaussian approximation with estimated parameters. The trained denoising network even outperforms the model trained with real data, probably because the NoiseFlow model allows the generation of an arbitrary quantity of realistic samples. This approach was recently extended to modeling noise in the sRGB domain [102]. The sRGB noise is locally correlated and has a much more complex distribution than in the RAW domain. The trained model also achieves quite good results in sRGB image denoising.

III.2 Synthetic training in deep learning

In the previous section, we showed that training real-world image restoration with a synthetic distortion model is possible. Nevertheless, all these methods use natural images as a starting point to form their dataset. In the present dissertation, one of the goals is to circumvent the need for such datasets by training networks with artificially generated data that would generalize well to unseen natural images. To the best of our knowledge, the works published during this doctoral program were the first to address image restoration problems with such methods. However, synthetic image models have already been used as training sets for other computer vision tasks. We can divide them into three categories: 3D-rendered

images requiring human design, images generated with neural networks, which initially required a large set of natural images to be trained, and random image processes which require no human supervision.

Training with 3D rendered images. 3D-rendered scenes are frequently used to either pre-train or fully train computer vision models for tasks where 3D information is crucial. That is why they are mostly used in optical flow or depth estimation methods. For these tasks, the displacement map or the depth map can be directly obtained from the rendering model, making the computation of ground truth data rather easy. These maps are hard to obtain in real-world conditions because they require expensive hardware or heavy processing.

Along a CNN optical flow estimator, Dosovitskiy et al. first presented a synthetic 3D dataset called Flying Chairs, which, as expected, contains images of chairs flying all over the scenes [53]. More realistic 3D rendered datasets were proposed, such as Sintel, a realistic opensource animated movie, where depth maps and displacement maps are available for every frame [30]. Mayer et al. proposed an even larger set composed of scenes of flying objects, an unrealistic Blender-generated animated movie, and a realistic set of Blender-generated scenes of driving cars [134]. Though realism is intuitively a desired property for generalization purposes, it seems that it is not the most important feature. According to the study of Mayer et al. [133], diversity in shapes and types of motions is the most important ingredient for good performances in optical flow estimation.

3D-rendered scenes can be useful for higher-level tasks such as semantic segmentation. With the emergence of autonomous driving, there has been a great interest in using deep learning architectures to segment the car's surroundings automatically. Annotated data is hard to obtain, thus explaining the interest in synthesizing diverse and realistic data. To that end, a dataset of annotated videos from the Gran Theft Auto game was acquired, where a player can drive a car in a realistic urban environment with interactions with other cars, pedestrians, and road signs [161]. Similarly, the Synthia dataset proposes its own virtual world simulator, from which the authors of [165] also acquire a large dataset for semantic segmentation. The previously mentioned datasets have been extensively used to either pre-train a network or as a data augmentation tool for real-world semantic segmentation for self-driving cars, greatly improving the results when real data is scarce. Example of these datasets are presented in Fig. III.8.

Though the first unrealistic synthetic sets were useful for low-level vision tasks such as optical flow estimation, higher-level vision tasks such as semantic segmentation require much more realism. This is achieved by a careful and time-consuming modeling of the scenes, with advanced rendering techniques and extensive human supervision by artists. We will see next that realism can be achieved by leveraging the learned representation of natural data.

Training with data generated by neural networks Considering the recent improvements in realistic image generations with GANs [72], normalizing flows [101], or diffusion models [82], it seems very appealing to use such models for data augmentation or pre-training. Since it is also difficult to quantitatively evaluate the quality of generated images, a recent

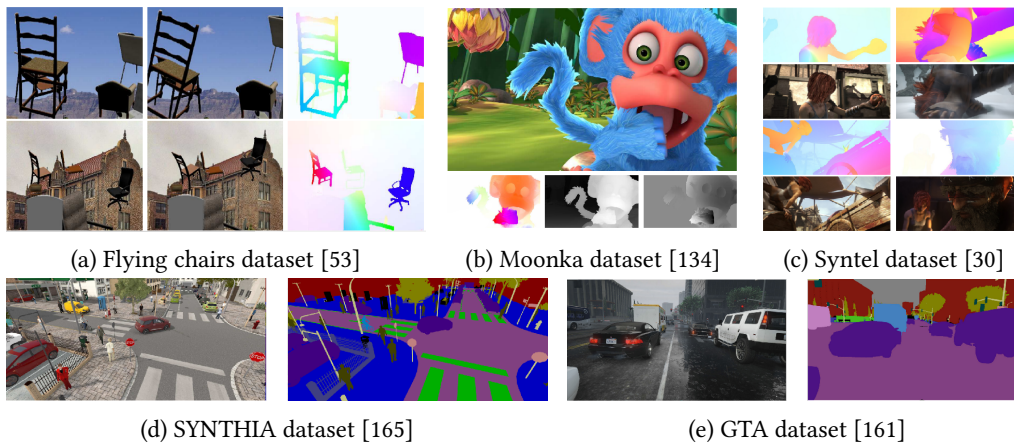


Figure III.8: Different examples of human designed 3D rendered scenes for synthetic training

idea was to evaluate the capacity of generative methods by training an image classifier on the generated data and report the classification score as the evaluation metric [171, 157]. These papers showed that although the images generated by BIGGAN (a large-scaled image generative network presented in [23]) are realistic, they are not diverse enough to efficiently train a classification network, echoing the remarks made for optical flow estimation. Nonetheless, Besnier et al. showed that continuously resampling data during training instead of generating a fixed set greatly improves the results [19], reducing the gap with classically trained methods.

While the results are still not as good in a supervised training setup, GAN-trained networks are very competitive in an unsupervised or semi-supervised set up. For unsupervised semantic segmentations, it was shown that annotations could be extracted from the latent space of GANs. This property eased the training of segmentation networks, reaching state-of-the-art performances in unsupervised learning [135, 208].

Since generative models seem useful for performing data augmentation, they were naturally used for contrastive learning. The field of contrastive learning aims at pre-training a network from unannotated data by applying diverse data augmentation techniques to a single image. By bringing closer the embedding of different views of a single image obtained with different distortions by optimizing a contrastive loss, the learned neural network can learn useful discriminative features. The last layer of the pre-trained network can then be specifically trained on a small set of real data [39]. This finetuning leads to very good performances, which would have been unreachable with a small quantity of annotated data. However, most augmentations are very basic and unrealistic transforms. Jahanian et al. propose to create a realistic multi-view of a single scene with the help of GANs [88]. This is done by sampling in the latent space close to the representation of a single object in order to get much more realistic views.

Generative networks have been used as a data augmentation tool for many other tasks beyond computer vision, in chemistry material simulation [44], or medical data generation [144]. However, It was recently shown that even though GANs produce realistic images, they experience difficulties reproducing multi-modal distributions [169], partially explaining the lack of diversity in the generated images. To the best of our knowledge, diffusion models,

which better handle multi-modal distribution, have not yet been used as a data generation tool for training neural networks.

Even if pre-training with generated data leads to good performances, training a GAN requires very large amounts of natural data. In the following paragraph, we will address methods completely agnostic of natural data or human design.

Training with artificial random processes. When it comes to training a neural network without realistic data either obtained with generative models or human-designed realistic scenes, the literature is quite scarce. Kataoka et al. proposed to use formula-based fractals to pre-train neural networks [93]. The underlying justification for this choice is that fractals mimic the properties of natural objects. The authors derive fractals of different categories by slightly varying the parameters of a formula-based fractal material. Having multiple samples of varying aspects for a single artificial class, Kataoka et al. were able to learn a classification network. The trained network then serves as the initialization for finetuning on real data. Even though the results are not as good as pre-training with natural images for some tasks, it is still often much better than training a network from scratch with limited amounts of data. Nakashima et al. further extended this work to pre-training visual transformers, improving their models by adding color [139].

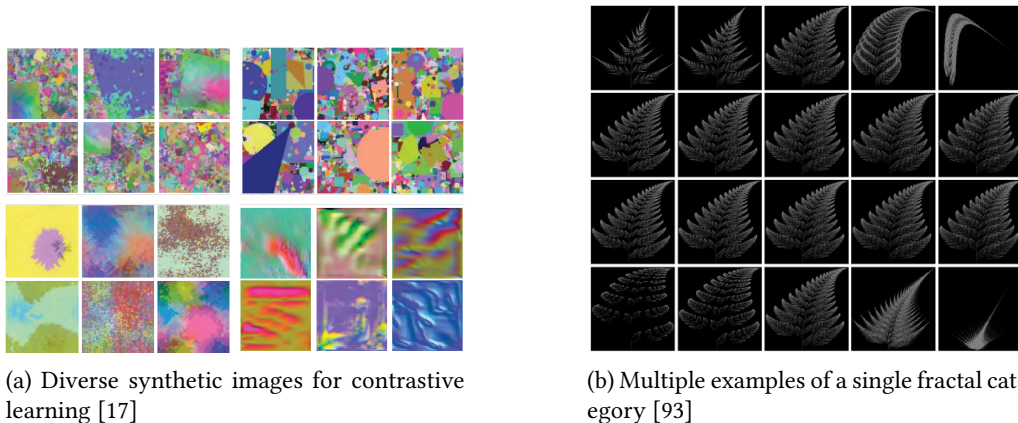


Figure III.9: Examples of synthetic image processes used for training

Recently Baradad et al. investigated which random image processes produced better results in a contrastive learning framework [17]. The processes included a variety of dead leaves models, statistical texture models, and hidden representations of an untrained neural network. They report that even though naturalism is crucial to pre-train natural-image-based tasks, it is not as important for specialized tasks (such as medical image analysis) or structure tasks that require an understanding of geometry and shapes.

Inspired by our work [6], Madhusudana et al. used 3D Dead Leaves images generated with Blender to train an optical flow and disparity estimator [128]. Madhusudana et al. extended the use of dead leaves images to train an image quality assessment metric in a contrastive setting [127].

To the best of our knowledge, the works presented in this dissertation are the first to propose training image restoration networks on synthetically generated data.



IV Dead leaves images: a synthetic image prior for simple image restoration tasks

IV.1 Introduction

Before the advent of deep neural methods for image restoration tasks, most approaches relied on relatively lightweight and explicit image priors. As presented in Section II.2, the use of total variation [167] as a regularization term is a consequence of a Laplacian prior on image gradients. Methods involving wavelet shrinkage [51] assume a regularity prior on wavelet coefficients related to Besov spaces. Non-local methods [26] rely on an auto-similarity hypothesis.

More recently, deep neural networks have achieved impressive results in all fields of image restoration: denoising, single image super-resolution, deconvolution, etc. In order to achieve such results, networks need to be trained on voluminous image databases. The resulting trained networks can be interpreted as image priors, even though it has been shown that the mere structure of networks can already be considered a prior [177]. In any case, such priors are non-explicit and involve a huge number of parameters. They also need to be retrained for each new acquisition conditions or specific imaging device [36].

In this chapter, we show that trainings on large image databases can be efficiently replaced by trainings on synthetic images. To achieve this, we rely on a mathematical model that is physically grounded and depends only on a few parameters, the scaling dead leaves model [10, 112, 73], which combines an occlusion-based dead leaves model with a scaling size distribution for objects. This model offers a good balance between simplicity and accuracy in accounting for natural images statistics. Moreover, we show that this model can be efficiently combined with natural image databases to enhance the capacity of deep neural networks to preserve details, without impairing their classical performance evaluation.

We believe that such a study both sheds light on the way convolutional neural networks can address restoration problems and opens interesting perspectives. First, this result shows that the mere structure of such networks is adapted to image restoration tasks and that despite their huge number of parameters they can be made near-optimal from just a few

principles and hyper-parameters. This result, and the fact that simpler, less structured models cannot achieve satisfying restoration performance, also highlights the type of geometric structures a neural network needs to see to be efficiently trained. Second, the proposed learning database has the potential to be modified according to specific acquisition devices and in particular to their point spread function, dynamic range, noise modality, etc. This opens the way to flexible, generic and relatively light learning schemes.

This chapter is organized as follows. We first define the dead leaves model and explain how it is used to generate synthetic databases in the following section. We then present the results of our synthetic training of an image restoration network in Section IV.3. In this experimental section, we justify the choice of each component of our generation’s algorithm by performing an ablation study. We then show how the same network can be trained with both natural images and synthetic images without impairing its performances on natural images. We then show our model’s versatility on one other restoration task: single-image super-resolution.

IV.2 Dead leaves images

The dead leaves model was originally introduced by the mathematical morphology school, with the aim of modeling porous media [132]. Despite being a particularly simple model, it was later shown to account for many statistics of natural images [10, 112]. Its structure is inherited from the sequential superimposition of random shapes, thereby mimicking a simplified image formation process, in which closer objects hide further ones. In this section, we first recall the mathematical definition of the model [20, 73], before detailing the algorithm and parameters that we will use to generate a synthetic image training dataset.

IV.2.1 The continuous dead leaves model

The dead leaves model is defined from a set of random positions, times and shapes $\{(x_i, t_i, X_i)_{i \in \mathbb{N}}$, with $\mathcal{P} = \sum \delta_{x_i, t_i}$ a stationary Poisson process on $\mathbb{R}^2 \times (-\infty, 0]$ and the X_i are random sets of \mathbb{R}^2 that are independent of \mathcal{P} . The sets $x_i + X_i$ are called *leaves* and for each i , the *visible part* of the leaf is defined as

$$V_i = (x_i + X_i) \setminus \bigcup_{t_j \in (t_i, 0)} (x_j + X_j),$$

where by definition $A \setminus B = A \cap B^c$, with B^c the complementary set of B . That is, the visible part of leaf (x_i, t_i, X_i) is obtained by removing from this leaf all leaves $x_j + X_j$ that are indexed by a time t_j greater than t_i (that falls after it). The dead leaves model is then a tessellation of the plane, defined as the collection of all visible parts. A random image can be obtained from this tessellation by assigning a random color to each visible part. In the following of this paper, the term *dead leaves* model will refer to this random image. Examples of dead leaves models can be seen in figure IV.2. The example in Figure IV.2a is a simple example where the leaves are disks with constant radius. Such a model (only depending on one parameter, the disk size) already mimics two important property of natural images, namely the presence of edges and homogeneous area. Nevertheless, it lacks details.

In order to get more faithful synthetic images, it has been shown that one could use power functions for the distribution of objects sizes [10, 112]. This way, the resulting images inherits scaling property and have been shown to reproduce many statistical properties of natural images. Such models are obtained by considering random leaves $R.X$, where X is a given shape and R is a real random variable with density $f(r) = C.r^{-\alpha}$, with C a normalizing constant. The case $\alpha = 3$ corresponds to a scale invariant model [112]. In order for such models to be well defined, values of R have to be restricted to values in (r_{min}, r_{max}) [73], resulting in a model with 3 parameters: r_{min} , r_{max} and α .

This model is especially appealing for natural images, because it incorporates two of their most fundamental property, non Gaussianity (as a result of edges) and scaling properties [138], in a very simple setting. Because this model contains details and edges at all scales, potentially of arbitrary contrast, it has been proposed as a tool for the evaluation of the ability of imaging devices to respect textures [32, 33] and was recently retained as a standard for quality evaluation [87].

IV.2.2 The generation algorithm

Algorithm 1: Dead leaves image generation algorithm

Parameters: $(r_{min}, r_{max}, \alpha, w)$, color_image
Output : X
mask = ones(w, w);
X = zeros($w, w, 3$);
while $\|mask\| > 0$ **do**
 tmp = $r_{max}^{1-\alpha} + (r_{min}^{1-\alpha} - r_{max}^{1-\alpha}) \times \text{random}()$;
 r = tmp $^{-\frac{1}{\alpha-1}}$;
 x,y = randint(0,w), randint(0,w);
 color = color_image(randint(0,w), randint(0,w));
 new_disk = disk(r, color);
 X = update_image(X, x, y, new_disk);
 mask = update_mask(mask, r, x, y);
end
X = downscale(X,5).

We now detail how to generate digital samples of the dead leaves model, following the procedure summarized in Algorithm 1. At each step, a random discrete disk of radius r and center (x, y) is generated as the set of discrete positions satisfying the corresponding disk equation. Centers are uniformly distributed in the image domain and radiuses are distributed according to a power law density with exponent α , as discussed in the previous paragraph. Radiuses are limited between r_{min} and r_{max} . To generate the image, we rely on a *perfect simulation* technique [95] and sequentially put the disks *below* the previously drawn disks, until the image domain has been fully covered. That is, at each step, pixels which have not been colored yet are given the color of the disk added at this step. The choice of the disk color will be shortly discussed. The used definition of discrete disk is crude and in particular does not include any anti-aliasing scheme. Therefore we first generate a large image that is then downsampled by a factor 5 after convolution with a Gaussian filter with $\sigma = 5/3$ (roughly

ensuring Shannon conditions). This step is a critical component of our algorithm. It allows for sub-pixel sized objects and for more natural boundaries. In Fig. IV.1a, we display a full size (2000,2000) dead leaves image before downsampling. A (20,20) crop on that same image (see Fig. IV.1b) exhibit very sharp boundaries and piecewise constant zones. A (20,20) crop on the downsampled image has a more realistic aspect (see Fig. IV.1c). The whole procedure can be seen as a very simple simulation of the camera acquisition of a dead leaves model with tiny objects.

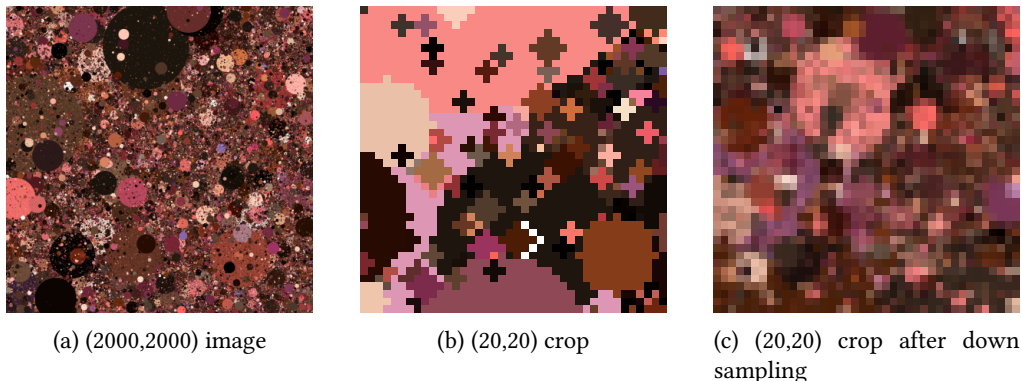


Figure IV.1: Illustration of the down sampling step

Color sampling. Our aim is to produce synthetic image databases accounting for the statistics of natural images. In particular, the marginal of color distribution should be as faithful as possible. In order to do so, we sample the colors of disks from natural image databases. As we shall see, this yields better restoration performance than sampling colors uniformly in the RGB cube. We can see in Fig. IV.2 that Fig. IV.2c, Fig. IV.2d have more realistic colors than Fig. IV.2b (uniform distribution prior on colors). In fact, for each generated dead leaf image, we sample the colors from *a single* natural image. This yields more coherence in the color of the generated images and also improves performance. This last fact is indeed an interesting observation, since it suggests that neural networks benefit from color combinations that are likely to be encountered in natural images. We will study this coloring procedure in detail in Chapter VI, in which we propose a data-agnostic alternative.

Size parameters. Since the shape of the leaves is fixed in our model (these are disks) the geometry of the generated images is solely controlled by the parameters of the size distribution, r_{min}, r_{max}, α . Images generated with different parameters can be seen in Fig. IV.2. As we can see, at fixed $\alpha = 3$ and $r_{max} = 2000$, the visual appearance strongly depends on the value of r_{min} . A large value of r_{min} yields structured images, with visible edges and homogeneous zones, whereas smaller r_{min} yields texture-like, cluttered images (see Fig. IV.2e, Fig. IV.2f). Similar observations can be made when varying the scale parameter α (see Fig. IV.2g, Fig. IV.2h). In our experiment, we chose the value $\alpha = 3$, which corresponds to scale-invariance. We also chose to fix r_{max} and to vary only r_{min} , which enables to set the structure/texture balance of the image.

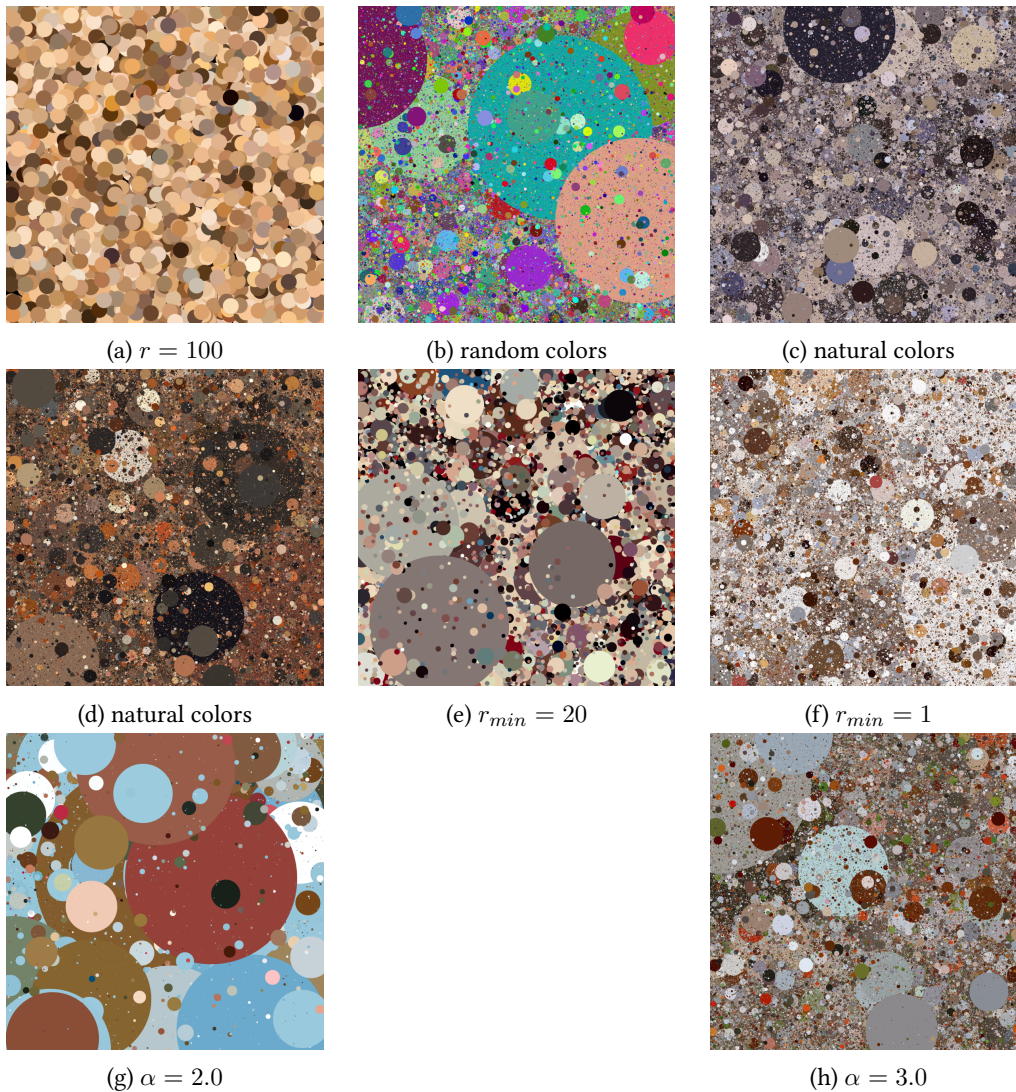


Figure IV.2: Dead leaves images generated with different parameters.

IV.3 Experimental results

In this experimental section, we first introduce the synthetic image dataset we consider. We then analyze and compare, numerically and qualitatively, the performance obtained when using only synthetic images, only natural images or a mix of both. To assess the relevance of some important features of our generation algorithm, we perform an ablation study in which we train FFDNet on a dataset of dead leaves images generated without particular components of the synthesis method described in Section IV.2. Finally, we illustrate the versatility of the proposed dataset by training the super-resolution network RDN [207].

Dead leaves dataset. In order to account for both homogenous areas and micro-textures, we build a dataset made of images generated with either $r_{min} = 1$ or $r_{min} = 16$, in both cases combined with parameters $\alpha = 3.0$ and $r_{max} = 2000$. Micro-textures being harder to

restore than homogeneous areas, we chose to have a 2 to 1 ratio between the two possible r_{min} values. The color distribution of the disks is given by the histograms of the natural images from the Waterloo database [126]. As shown previously, this leads to a more coherent color distribution than randomly sampling the RGB cube. Finally, we decided to apply a Gaussian blur to a 10th of the dataset, with a standard deviation uniformly sampled between 1 and 3. Indeed, most natural images tend to contain blurry zones due to the depth-of-field of cameras. By adding a very simple blur model to some of the images of the dataset, we expect blurry areas in natural images to be better restored.

IV.3.1 Denoising results

In order to assess the capacity of the proposed synthetic dataset to successfully train a denoising network, we consider the network FFDNet. It is a state-of-the-art image denoising CNN, which was introduced by Zhang et al. [202] and thoroughly examined in [175]. Its main specificity relies in the first layer of the network: to increase the receptive field and to handle a wide range of noise levels, the image is divided in four sub-images which are concatenated to a noise map indicating the local noise standard deviation. This tensor is then passed through a more classic network of batch normalized convolutional layers, with an architecture similar to that of DNCNN's [200]. It then outputs the four denoised sub-images, which are reassembled to create the final denoised image.

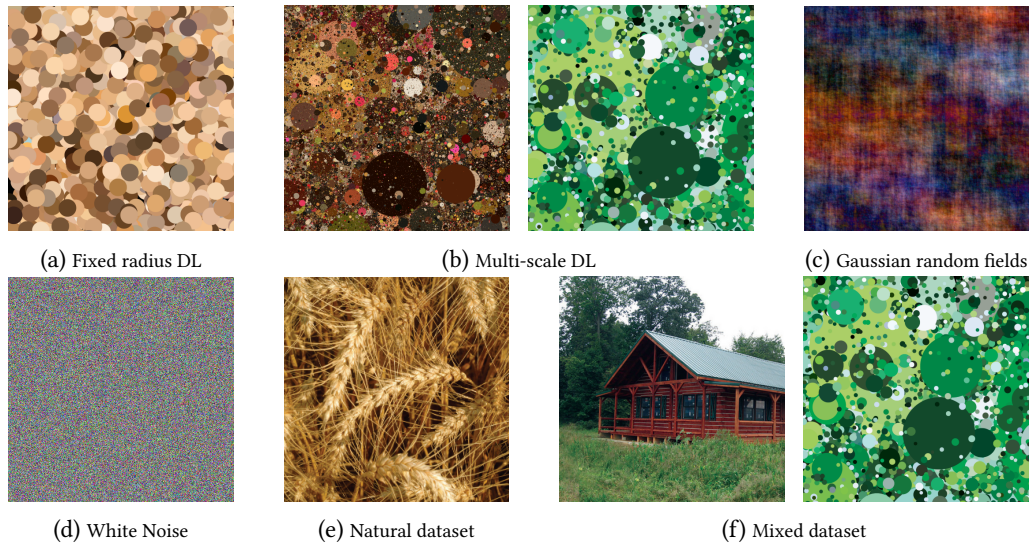


Figure IV.3: Example of images of the different training datasets.

FFDNet results. To compare different trainings fairly, we use the same optimization algorithm for all trainings. It consists of 80 epochs with the Adam optimizer and the L2 loss, starting with a 10^{-3} learning rate l_r . There is a decay of factor 10 at epoch 50 ($l_r = 10^{-4}$), and another decay of factor 100 at epoch 60 ($l_r = 10^{-6}$). For each training, we used 350k (50, 50, 3) patches, extracted from either the dead leaves dataset, or the natural image dataset, or a mix of both. The mixed dataset contains $\frac{1}{3}$ dead leaves images, and $\frac{2}{3}$ natural images. To show that scaling properties are needed to model natural images, we also trained FFDNet on dead leaves images generated from disks with a fixed radius of 100.

In addition, we also consider two alternative training schemes from datasets of synthetic images: white noise images and Gaussian random fields [60]. We give examples of different training sets we used in Fig. IV.3. Numerical evaluation is performed on 2 test sets of natural images (CBSD68, Kodak24) and one set of 24 dead leaves images, generated from the colors of Kodak24. For each test, we compute the average PSNR, SSIM [187] and PieAPP metric [152], a recent perceptual metric based on human annotation, which tends to fit very well with human perception.

Table IV.1: Numerical comparisons of the different trainings of FFDNet. We evaluated the results on two benchmark datasets for image denoising (CBSD68 and Kodak24), and our dead leaves testset, at two noise levels. Each cell contains the triplet PSNR/SSIM/PieAPP. The best results are in blue, the second in red.

σ	Dataset	CBSD68	Kodak24	Dead leaves testset
$\sigma = 25$	White Noise	19.52/0.416/2.386	19.68/0.365/2.502	20.36/0.607/2.043
	Gaussian field	29.63/0.845/1.402	30.24/0.835/1.471	26.23/0.826/1.254
	DL $r = 100$	29.56/0.820/1.218	30.49/0.819/1.024	26.13/0.799/1.263
	Dead leaves	30.58/0.867/0.711	31.27/0.859/0.739	27.46/0.865/0.573
	Mix	31.07/0.881/0.639	31.98/0.876/0.603	27.33/0.860/0.567
	Natural Images	31.09/0.882/0.629	32.00/0.878/0.599	27.05/0.851/0.576
$\sigma = 50$	White Noise	15.58/0.247/4.682	15.71/0.209/4.785	16.24/0.387/2.932
	Gaussian field	26.68/0.738/2.203	27.41/0.737/2.353	23.31/0.694/2.158
	DL $r = 100$	26.85/0.720/1.563	27.91/0.739/1.314	23.24/0.654/2.005
	Dead leaves	27.40/0.762/1.088	28.21/0.765/1.154	24.21/0.737/1.020
	Mix	27.86/0.782/0.997	28.86/0.789/0.985	24.12/0.732/1.015
	Natural Images	27.87/0.786/0.991	28.89/0.792/0.978	23.90/0.722/1.053

On both natural image testsets and on the dead leaves testset, we observe that the model trained on dead leaves outperforms by a large margin all other models trained on alternative synthetic image datasets (0.9db for the Gaussian model and, without surprise, 11 dB for the white noise model), see Table IV.1. Visually, the Gaussian field model leads to denoised images still containing noise and grid-like artifacts, which severely impact the PieAPP metric. Observe that for both image models of white noise and Gaussian noise, the optimal solution is known and given by the Wiener filter (multiplication by a constant in the first case and linear filtering in the second). It is interesting to note that the network did not learn to apply this theoretical optimal solution to natural images in either cases. Confirming our intuition that an image model with scaling properties is needed, the dead leaves model with a fixed radius tends to strongly over-smooth the image, thus losing all texture information. This amounts to a loss of 0.65 dB on natural image testsets, and 1.2 dB on dead leaves images.

More surprisingly, the model trained exclusively on dead leaves images performs only 0.6dB lower than the model classically trained on natural images. Visually, the results are still almost as good, despite some limitations. In particular, the synthetically trained model has some difficulties with thin and low contrast lines, and occasionally creates dot artifacts. In other situations, the synthetic training improves the results, as can be seen in Fig. IV.4, where the texture of the rusty artwork is quite well restored, with a better preservation of fine details than with the model trained on natural images. Another very interesting result is the fact that training on a mix of dead leaves images and natural images does not affect the result of the denoising model on testsets of natural images, the difference in PSNR being less than 0.02dB. Visually, the results are almost identical, with a slight advantage for the mixed trained model on texture areas. On the dead leaves test set, the mixed trained model

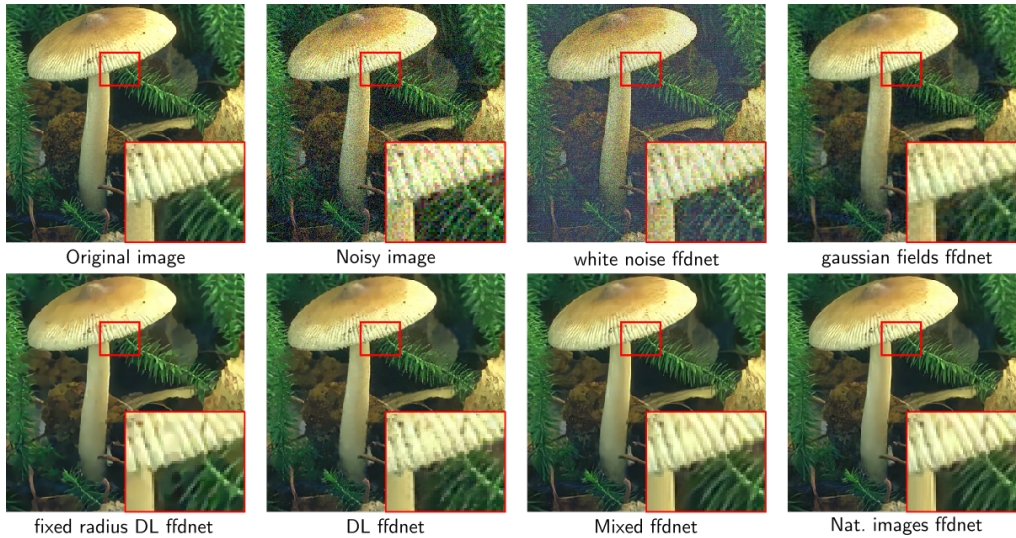


Figure IV.4: Denoising comparison with different FFDNet trainings. Top row from left to right: clean image, noisy image with $\sigma = 25$, model trained on white noise, model trained on Gaussian fields. Bottom row from left to right: model trained on dead leaves images with fixed radius $r = 100$, model trained on the dead leaves dataset, model trained on the mixed dataset, model trained on natural images.

clearly outperforms the natural image trained model by 0.25dB. This result suggests that jointly optimizing the response to this kind of mixed datasets has the ability to increase some aspects on which imaging devices are evaluated. Indeed the scaling dead leaves model is classically used to evaluate the ability of imaging devices to preserve texture areas [32, 33] and the corresponding scale-invariant test chart has recently become an ISO standard (ISO/TS 19567-2:2019).

Ablation study. To confirm the choices made to build the synthetic dataset, we compare different trainings performed with different parameters or design choices, both visually and numerically.

We first illustrate the impact of r_{min} on the denoising results. As shown in Fig. IV.5, the smaller the r_{min} , the better micro-textures are restored. Conversely, they are smoothed when r_{min} gets larger. On the other hand, homogeneous zones contain artifacts when r_{min} is too small, and are well restored when r_{min} is larger. This behaviour is expected since a large r_{min} leads to dead leaves images with homogeneous zones, and a small r_{min} to more micro-textures zones. Referring to Table IV.2, the optimal r_{min} seems to be between 4 and 8. However, by mixing images generated with $r_{min} = 1$ and $r_{min} = 16$, we get a noticeable improvement in PSNR (0.17dB) and in image quality, as can be seen in Fig. IV.5.

Other important features of our algorithm are: the color distribution, the downscaling step, and the blur. As we can see in Fig. IV.5, when we sample the disks colors uniformly in the RGB cube, the denoised images show many color artifacts. The additive Gaussian noise creates unnatural colors that the network doesn't identify as such, since it has not been trained on images with natural colors. This leads to a performance gap of more than 1 dB in PSNR. The downscaling step is also critical, as it allows sub-pixel objects and more natural boundaries. In Fig. IV.5, we can see that the network trained on the dead leaves

Table IV.2: Impact of the parameters and ablation study. For each cell, we report the PSNR of the model tested on the Kodak24 dataset. In the first 5 columns(DL1 to DL16), we fixed the parameters to $r_{max} = 2000$, $\alpha = 3.0$, with natural colors and the downscaling step. From column 6 to 8, we keep the same parameters as in the final dataset, but we remove some important features of the generation. The last column corresponds to the final result.

σ	DL-1	DL-2	DL-4	DL-8	DL-16	Rand. col	No sub	No blur	Final
25	31.03	31.03	31.09	31.07	30.98	29.99	30.79	31.25	31.27
50	27.98	27.96	28.04	28.06	28.05	27.16	27.74	28.20	28.21

dataset without subsampling tends to over-smooth texture areas, and to produce stair-casing artifacts. We can also identify some disk-like objects with hard boundaries in the images, creating an unnatural aspect. In terms of PSNR, this amounts to a loss of 0.5 dB compared to the training on our final dead leaves dataset. For the final synthetic dataset, we decided to blur 10 % of images. As we can see in Table IV.2, removing this step has almost no impact on the PSNR. Nonetheless, we observe in Fig. IV.5 that removing this step makes blurry zones look sharper than they really are. Overall, the final dead leaves dataset yields better results, numerically and visually, than the ablated datasets.

IV.3.2 Single-image super-resolution

Network and Training. Next, we turn to the task of super resolution. We chose to retrain the Residual Dense Network (RDN)[207], a classical super-resolution network. Its architecture is based on residual dense blocks, a combination of dense blocks introduced in [83] and residual connections. The model trained on dead leaves (DL-RDN) is trained with the same dataset used for AWGN removal. The model trained on natural images (called Nat-RDN) is trained on a portion of the DIV2K dataset. The training is performed for 800 epochs with a batch-size of 16. We optimize the L_1 distance between the predicted image and the high resolution ground truth with the ADAM optimizer. We based our experiments on an un-official yet exact github implementation.¹

The numerical evaluation shows a similar behaviour to the one observed for AWGN removal presented in Section IV.3.1. The loss in performance when using the dead leaves model is of 1.2dB and 0.6dB for a super-resolution of scale 2 and 3 respectively. The results on the Set5 and Set14 datasets, which are common benchmarks for super-resolution, are given in Table IV.3.

Table IV.3: Numerical evaluation of the super-resolution results. We report the PSNR of RDN trained either on the dead leaves dataset or on the DIV2K dataset [8].

Dataset	Set 5		Set 14	
	×2	×3	×2	×3
Dead leaves	36.76	33.82	32.93	30.42
Natural Images	38.18	34.71	33.88	30.73

Visually, the super-resolution results are similar between the two trainings as we can see in Fig. IV.6. Looking closer to the details of the restored images, we see that the output of DL-RDN tends to reproduce better the white dots in the butterfly wing and the texture of

¹<https://github.com/yjn870/RDN-pytorch>

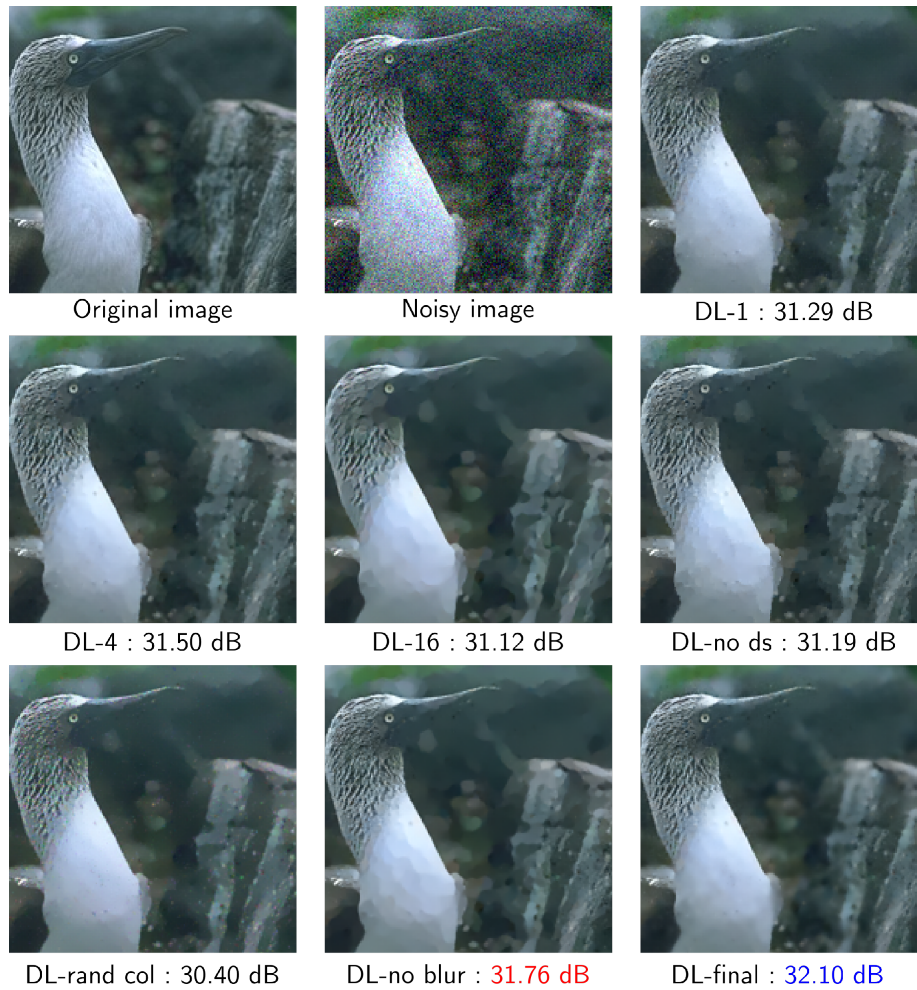


Figure IV.5: Visual illustration of the ablation study. From left to right: top line: clean image, noisy image, $r_{min} = 1$, /center line: $r_{min} = 4$, $r_{min} = 16$, no subsampling / bottom line: uniform color distribution, no blur, final result.

the bird's beak. Conversely, the thin lines in the yellow regions of the butterfly wing have a "dotted" aspect in the DL-RDN image, whereas they are properly restored in the Nat-RDN image. Indeed, the dead leaves model does not contain any straight and thin lines, making it harder for this model to retrieve them. Enriching the dead leaves model with elongated shapes or adding different local structures in the training are perspectives to solve this issue.

IV.4 Conclusion and Perspectives

To the best of our knowledge, this work is the first effort to train an image restoration network on synthetic images. After introducing the dead leaves model and its digital implementation, we carefully studied the role of each component of the image generation method, and their impact on the restoration performances. Both for denoising and super-resolution, models trained on our dead leaves dataset are surprisingly close to those trained on natural

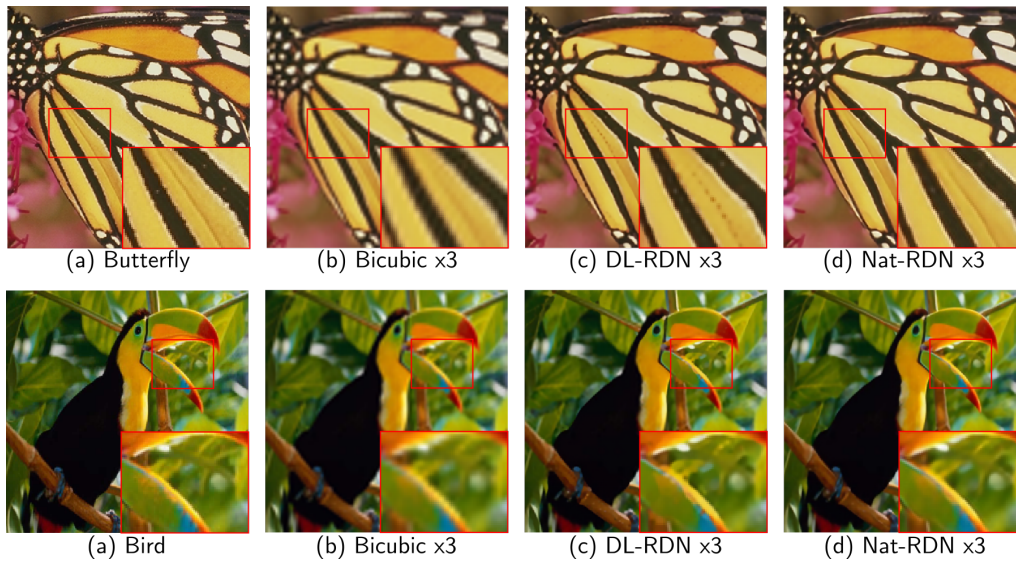


Figure IV.6: Visual comparison between the two differently trained RDN approaches, together with a simple super resolution baseline, for a zoom factor of 3. From left to right: High resolution image, Bicubic interpolation, RDN trained on dead leaves images, and RDN trained on natural images.

images. When mixing natural and synthetic images in the training, the results reach performances on par with the model trained on natural images only. Both results indicate that the dead leaves model with scaling properties is a good candidate to replace natural images for training, with only a few parameters. Indeed, the geometry of the model only depends on three parameters: α, r_{min}, r_{max} . Even though the color parameters are still relatively numerous, we investigated, in chapter VI, simpler color sampling procedures based on a statistical study of the colors in natural images. Another perspective would be to complement our dataset with sinusoidal patches to better restore oscillating patches and straight lines. Eventually we believe, as already explained, that a synthetic dataset can be a simple way to avoid retraining new imaging devices with relatively heavy acquisition campaigns [36] and we plan to investigate this ability further.



Fully synthetic training for real world image denoising

V.1 Introduction

In the previous chapter, we experimentally showed that training an image restoration neural network on synthetic dead leaves images leads to performances nearly as good as if training on natural images. These results confirmed that a straightforward synthetic model could encapsulate the key ingredients for successfully training an image restoration neural network. However, the experiences were limited to simplified tasks such as additive white Gaussian noise removal and single image super-resolution. Such distortion models over-simplify real distortions. Moreover, as explained in Section III.1.5, neural networks trained with these distortion models fail to generalize to real-world conditions.

On the other hand, deep learning models for real-world conditions commonly require real data. Forming a dataset of pairs of clean/distorted images is cumbersome, and computing a ground truth image requires heavy processing [2, 149]. In order to circumvent the burden of the acquisition process, recent works propose to either accurately model the distortion process with physically based models [190] or trained neural distortion generators [1], allowing the generation of an arbitrary amount of distorted data. Going further, Brooks et al. [24] propose synthesizing RAW data from existing sRGB databases and synthesizing distortions in an artificial RAW domain. Yet, all the denoising models trained with these distortion models use existing datasets of clean natural images, which need to be acquired.

In order to ease data generation, we address in this chapter the problem of real-world image restoration by training a CNN with a fully synthetic model. This model is two-fold: first, we generate a large quantity of ground-truth RAW images with a new version of our dead leaves model, adaptable to different imaging devices. Second, we propose a synthetic yet accurate distortion model for two restoration tasks in order to generate our distorted data. These tasks are Smartphone RAW image denoising [2] and low-light image enhancement, inspired by the paper *Learning to See in the Dark* [36].

Our experiments show that a fully synthetic training of image restoration neural networks achieve performances on par with the models trained with real data. In some cases, the visual quality of the obtained results is even better since our ground truth data is truly undistorted, unlike the ones of real datasets, which are only approximately noiseless. We believe this work opens the way for lighter training schemes since it completely circumvents the need for real data acquisition.

This chapter is organized as follows. In Section V.2, we first present an adaptation of the dead leaves model for RAW images, which are much more complex than sRGB images. Second, we explain in Section V.3 the distortion models for each task and how we estimated the corresponding noise parameters. We then present the results of our synthetic training in comparison with real training in Section V.4 before drawing some conclusions.

V.2 Synthetic RAW dead leaves images

In the present section, we present how to generate ground truth synthetic RAW images for both restoration tasks. The algorithm to generate synthetic Dead Leaves RAW images builds on Algorithm 1. We propose some modifications concerning the color sampling procedure, as well as an additional mosaicking step.

Coloring procedure. In order to color the leaves of one synthetic image, we sample them in the colors of a randomly picked RAW image in the original database. We remind the reader that sampling colors uniformly at random in the RGB cube leads to unnatural colors, resulting in poor image restoration performances for the trained network.

In an sRGB image, each pixel contains an RGB triplet. This is not the case for RAW images, for which each pixel either corresponds to a red, a green, or a blue value. A common transform of a RAW image allows for a representation in a four-channeled image with two green components, one red, and one blue. The obtained representation allows us to sample a color quadruplet to color the leaves of our dead leaves images. This representation is computed thanks to an operation that we call *packing*. Its procedure is described in Fig. V.1a

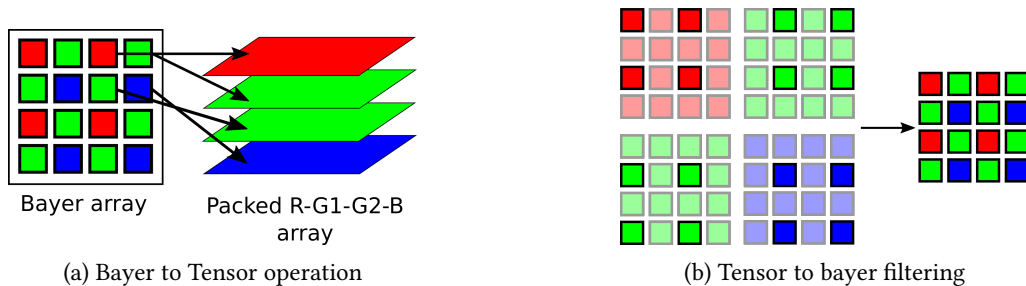


Figure V.1: Bayer frame manipulations

Following this color sampling procedure, we synthesize a dead leaves image of size $(4H, 4W, 4)$ with the same algorithm as for RGB images. It is worth mentioning that the green components of a single color should be equal. Otherwise, there would be checkerboard

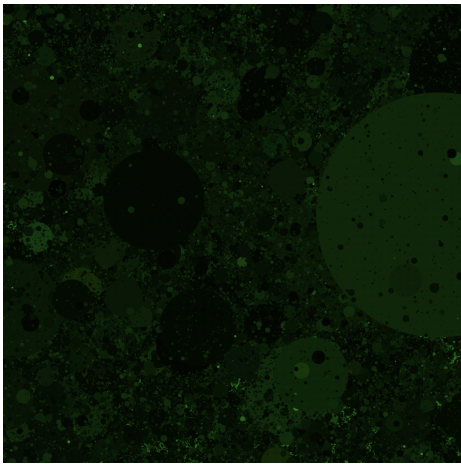
artifacts on the final developed sRGB image. In order to eliminate this behavior, we simply replace both green components with their average.

Mosaicking step. As for sRGB dead leaves images, we apply blur and downscaling on the generated image. These operations are included to remove artificially sharp boundaries and to include sub-pixel-sized objects, which we often encounter in natural images. We also apply blur for antialiasing reasons. The resulting dead leaves tensor has the following dimension $(H, W, 4)$. In order to go back to the RAW domain, which has only one channel, we filter the created tensor following the color filter array (CFA) of the camera of the source image, as shown in Fig. V.1b. That is, at each position (x, y) in the array, we only keep the value of the color at the same position in the Bayer frame. The size of the final RAW dead leaves image is now $(H, W, 1)$, where each pixel value corresponds to a single color information, as in the Bayer Frame.

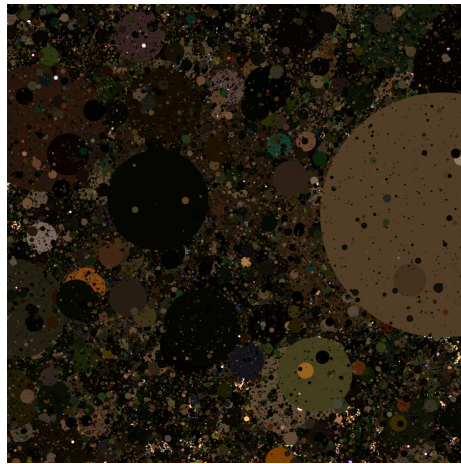
Tasks specifics. For RAW smartphone images, the resulting dead leaves image is the final ground truth image. In order to learn a RAW denoising network, we will add synthetic RAW noise to that image to form noisy/clean image pairs. However, for low-light image enhancement, the goal is to retrieve a properly developed sRGB image. Therefore, we still need to develop our generated RAW image in an sRGB image. To do so, we encapsulated our simulated RAW image into a valid RAW file and preprocessed it with the DCRAW program. To generate distorted images, we will use the proposed distortion process to the synthesized RAW images used to generate the ground truth sRGB image. Fig. V.2 shows an example of such images.

Dead leaves image generation parameters. To account for textures, homogeneous zones, and blurry areas, we generated half the images with $r_{min} = 4$, and the other half with $r_{min} = 100$. r_{max} is always set to 2000 and $\alpha = 3$ to have a scale-invariant dead leaves model. The reader may notice that these values are higher than for sRGB images. We are indeed dealing with higher-resolution images than sRGB images, which contain less fine-grained details than smaller-resolution images. Because of the Bayer frame structure, it is indeed a common practice for camera manufacturers to include a low pass filter before the sampling stage, to prevent aliasing.

This is why we chose to increase the r_{min} parameter. We applied a Gaussian blur to one third of the images with a kernel standard deviation uniformly sampled in $[1, 3]$, to better restore blurred images. The reader might notice that this blur model is an oversimplification of camera blur. We believe it is still better than not including any blur, and we hope this could be improved in future works.



(a) RAW ground truth dead leaves image



(b) sRGB image developed with DCRAW



(c) source image developed in an sRGB image

Figure V.2: Examples of a dead leaves images and its source color image for the synthetic set for low light image enhancement

V.3 Distortion Modeling

V.3.1 RAW smartphone images noise model

V.3.1.1 Noise formula

As we mentioned in Section III.1.5, image-denoising CNNs trained with AWGN fail to denoise real noisy images. In the RGB domain, noise is indeed signal-dependent and spatially correlated. When considering RAW images, the noise is (mostly) i.i.d. but signal-dependent, making the AWGN model of limited use. For such images, the noise can be modeled with a signal-dependent component (the shot noise), which accounts for the quantum behavior of photons, and a signal-invariant component (the read noise), which accounts for all the electronic noises that mostly occur when measuring the electrons produced by the sensor. Having a theoretically clean RAW image X , a real observation Y can be written as:

$$Y = X + N_s(X) + N_r,$$

where N_s stands for the shot noise, and N_r for the read noise. A common practice to model the shot noise is to use a Poisson law. While most papers use the Gaussian-Poisson approximation, [189] showed that it does not stand in low-light conditions.

At the sensor level, let us denote by I_t the matrix (image) of the expected number of photons. Due to the Poisson behavior of photons, the number of photons actually measured is $I \sim \mathcal{P}(I_t)$. Note that, contrary to I_t , I is an integer, which is crucial in the case of very low-light conditions.¹ In order to retrieve a RAW image from the photon count, I is multiplied by a digital gain K that depends on the ISO at shooting time. The RAW noisy image is therefore $Y = KI + N_r$. Following these equations, we can generate a noisy image Y from a noiseless image X by retrieving the initial photon count $I = X/K$. We can obtain the noisy simulation of the shot noise with the following equation:

$$X + N_s(X) \sim K \times \mathcal{P}(X/K),$$

where K depends on the ISO.

The read noise N_r accounts for all electronic noise sources. To generate it, we chose the Gaussian approximation, the most common and reliable model:

$$N_r \sim \mathcal{N}(0, \sigma^2),$$

where the standard deviation σ also depends on the ISO.

In order to generate our synthetic image dataset, the last step is to estimate the noise model parameters, which are device-dependent. In the experimental section, we will consider the problem of denoising images from the SIDD dataset [2]. This database comprises RAW images from 5 phones at different ISOs, in controlled lighting conditions. In order to estimate the parameters for each device, we use the ground truth images from this database. These images are obtained by carefully averaging 400 shots of the same scene in identical conditions. The authors provide noise parameters along each picture but for a Gaussian-Poisson approximation of the digital noise. Unfortunately, the parameters provided were also incorrect: for example, the read noise was supposed to be null for some cameras.

V.3.1.2 SIDD dataset noise parameters estimation

The SIDD dataset comprises RAW images taken with five phones (iPhone 7, Samsung S6, Google Pixel 3, LG 4, Nexus 6) at different ISOs in different lighting conditions. Each image comes with a set of noise parameters, but as previously mentioned, these parameters are inaccurate. Therefore, we estimate the noise parameters for each existing (phone, ISO) pair in the original dataset.

For each (phone, ISO) pair (P, I) , we collect all the corresponding images in the dataset in a subset $D_{P,I}$. For each image in $D_{P,I}$ we estimate two noise parameters (K, σ) per channel. Our estimation method is based on simple properties of the Poisson law. Let us consider X a clean reference and Y its noisy counterpart. Following our noise modeling, $Y \sim K \times \mathcal{P}(X/K) + \mathcal{N}(0, \sigma^2)$. Therefore $\mathbb{E}[Y] = X$ and $\text{Var}(Y) = KX + \sigma^2$, so that there is an affine relationship between the variance and the expectation of the noisy

¹Actually, one has no access to the measure I but to the proportional measure, which is the number of electrons produced for each digital value.

observation. A regression model is then used to estimate those parameters. Now, given a clean and homogeneous patch \mathbf{X} of grey level u , we can compute the empirical variance v of the corresponding noisy patch \mathbf{Y} . The pair (u, v) should follow the affine relationship. This reasoning motivates our estimation procedure, which we describe in the following pseudo-code.

Algorithm 2: Noise parameter estimation for a (P,I) pair

```

Input :  $D_{P,I}$ 
Output:  $\hat{K}, \hat{\sigma}^2$ 
 $K, \sigma^2 = \text{list}(), \text{list}()$ 
for  $(X, Y) \in D_{P,I}$  do
     $p\_clean, p\_noisy = \text{EXTRACT\_PATCHES}(X, Y, 11)$ 
     $m\_clean, v\_clean = \text{STATS}(p\_clean)$ 
     $m\_noisy, v\_noisy = \text{STATS}(p\_noisy)$ 
     $ind = \text{argsort}(v\_clean)[0 : N_5]$ 
     $U, V = m\_clean(ind), v\_noisy(ind)$ 
     $model = \text{Theil\_Sen\_Regressor}.fit(U, V)$ 
    if  $model.score() > 0.7$  then
         $K.append(model.slope)$ 
         $\sigma^2.append(model.intercept)$ 
    end
end
 $\hat{K} = \text{median}(K)$ 
 $\hat{\sigma}^2 = \text{median}(\sigma^2)$ 

```

First, we extract all the patches of size $(11, 11)$ of a clean image \mathbf{X} and a noisy image \mathbf{Y} . For each patch, we compute its mean and variance. We select the 5% of the clean patches with the lowest variance in order to select the ones which are the most homogeneous. We form a first vector U containing the mean of those patches in the clean image. Then, we form a second vector V containing the variance of the same patches in the noisy image. Thanks to the pair (U, V) , we can fit a Theil–Sen regressor to find the slope K and the intercept σ^2 of the affine line. We chose this regression method because it is more robust to outliers than the linear regressor. We discard this parameter estimation if the regression fails to reach a satisfactory score of 0.7. In Fig. V.3, we show an example of the graph (U, V) and its regression for an image at ISO 3200 for the Samsung S6.

This method is applied for each image of the $D_{P,I}$, yielding many estimates for K and σ^2 . To give a unique estimate of these parameters, we select the median of all the estimates for each parameter. In Fig. V.4, we plot the estimated parameters in a logarithmic scale. Interestingly, the parameters seem to follow an affine rule, as was noted in [24].

To generate a noisy image from a clean RAW image, a small but important detail needs to be clarified. All images in the SIDD dataset are normalized between 0 and 1. However, Poisson noise is defined for integer values. The clean images indeed have an artificial float precision since they are obtained by averaging 400 noisy pictures. We first quantize the clean RAW image on N bits to generate Poisson noise, where N depends on the camera. We then add Poisson noise before normalizing and adding the read noise. The simulated noise is very close to the real one, as we can see in Fig. V.5.

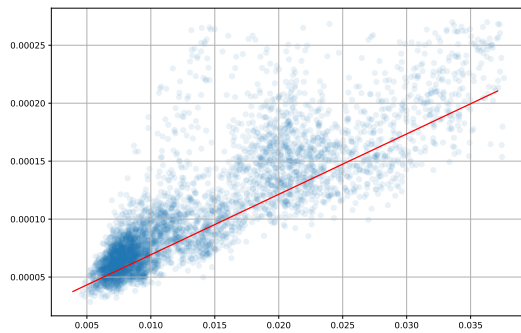


Figure V.3: In blue, a scattering plot of the local variance against the local mean. In red, the Theil-Sen regression of these points.

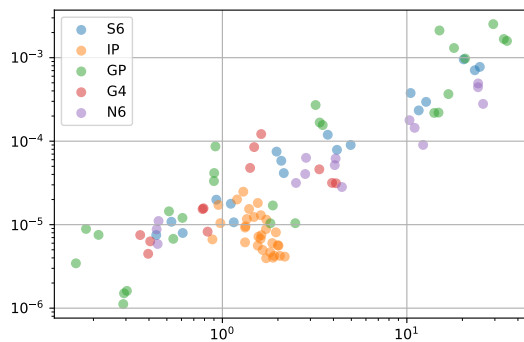


Figure V.4: Noise parameters of the SIDD dataset. We plot the read noise parameter σ^2 against the shot noise parameter K for each phone at different ISOs. The different colors represent different phones.

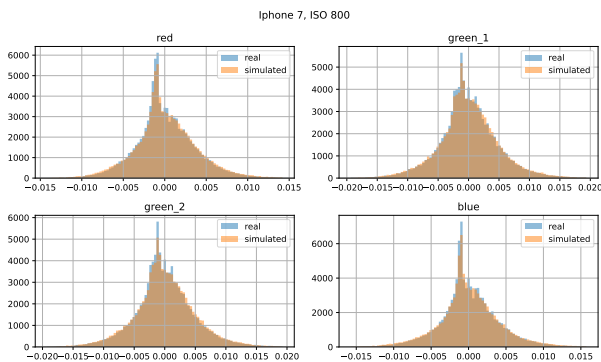


Figure V.5: Histograms per channel of a comparison of the difference between a clean and a noisy image, whether it is a simulation or a real noisy image. The example image was shot at ISO 3200 with the Samsung S6.

V.3.2 Low-light image enhancement

V.3.2.1 Distortion Model

In extreme low-light conditions, the noise model becomes more complex. In [189], the authors provide an extensive study of noise in such conditions for the Sony $\alpha 7s2$ camera, which was used to collect the See-in-the-Dark dataset [36]. According to them, the shot noise can still be modeled by a Poisson noise, but the read noise can not be modeled as a zero-mean Gaussian noise. They model the read noise as a Tukey-Lambda distribution, and estimate its parameters in a complex protocol. A key ingredient of their success was to add an estimated offset μ to their previous zero-mean noise. Since the authors do not provide these parameters, we chose, for simplicity, an uncentered Gaussian approximation of the read noise:

$$N_r \sim \mathcal{N}(\mu, \sigma^2),$$

where μ and σ both depend on the ISO. These parameters were estimated with a simple protocol that we describe next. In low-light conditions, random banding patterns are visible due to a readout electronic noise. It is modeled by a random offset added to each row of the clean RAW image. This banding noise N_b follows a centered Gaussian distribution $\mathcal{N}(0, \sigma_r^2)$, where σ_r also depends on the ISO. Finally, the image is quantized, inducing quantization noise $N_q \sim \mathcal{U}([-1/2, 1/2])$. Having X a ground-truth long exposure noiseless image, our short exposure noisy RAW observation Y follows the following formula:

$$Y = \frac{X}{\gamma} + N_s \left(\frac{X}{\gamma} \right) + N_r + N_b + N_q,$$

where γ corresponds to the ratio of exposure times. In the SID dataset, this ratio is either 100, 250, or 300, depending on the scene. The same factor γ multiplies the short exposure image at the entrance of the network so that the ground truth image and the input image have a similar dynamic. That is why the slightest error in the noise model is very costly and why we need to carefully fit the noise parameters, as we shall see next.

V.3.2.2 See-in-the-Dark noise parameters

To estimate the noise parameters for this task, we adapted the algorithm presented for Smartphone RAW noise parameters estimation. In order to validate our estimation, we could also compare our estimations for some parameters to the one made on the "Photons to Photo" website ² [**photons2photo**]. It provided an estimate for the read noise variance σ^2 at different ISOs and the for the shot noise parameter K only for ISO 100. Assuming a linear relationship between the parameter K and the ISO, this value can be extrapolated to other ISOs. In the following, we explain how to estimate all parameters. These parameters are then compared to those available on the "Photons to Photo" when available and to the linear extrapolation for K .

The estimation algorithm of the shot noise parameters is very similar to the one we used for the SIDD images. However, the specific format of the See-in-the-Dark images and their dynamic range's particularity should be considered. These RAW images are indeed coded in

²<https://www.photonstophotos.net/>

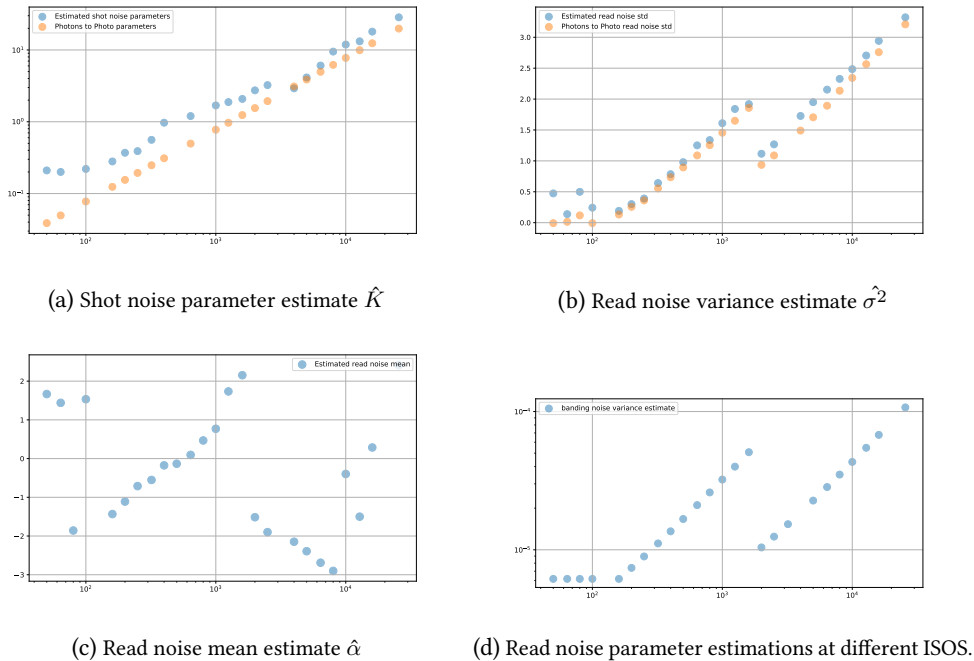


Figure V.6: Estimation of our shot noise and read noise parameters at different ISO. Comparison of our estimates against Photons to Photo's

14 bits integers for the Sony $\alpha 7s2$ camera where the black level is set to 512 and the highest value to 16384. We first subtract the black level from the reference and the noisy image. Then, we divide the reference image by the exposure time ratio to bring it to the dynamic of the short exposure image. We then run our estimation algorithm for both the digital gain K and the read noise variance σ^2 , as presented in the previous section.

We report in Fig. V.6a and Fig. V.6b the estimates of the shot noise gain K and the read noise variance σ^2 . As far as the read noise parameter σ^2 is concerned, we see that our estimates are very close to the one provided by the "photons to photo" website [[photons2photo](#)], with a small discrepancy at lower ISOs. For the shot noise parameter, K , our estimates differ a little from the extrapolated values of the website.

In Fig. V.6c, we report our estimates of the read noise mean. For each pair of noisy and clean images at a fixed ISO, we compute the difference between the noisy and the rescaled clean images. We then compute the mean of this error which gives an estimate of the read noise mean at each ISO.

To estimate the banding noise parameter for each ISO, we perform similarly as for the read and shot noise parameter estimation. We select (50,50) patches of low variance in the clean and noisy images with the lowest possible intensity. Then we compute the mean for each row. This value corresponds to the mean value of the signal plus the offset added by the banding noise. Therefore, we compute the mean's estimations variance in order to give an estimate of the banding noise variance. We report our estimates of these parameters at different ISOs in Fig. V.6d.

V.4 Experiments

V.4.1 Noise removal on Smartphone RAW images

Network and Training. We consider here the problem of denoising smartphone RAW images from the SIDD dataset [2]. To denoise RAW images, we adapted the U-Net architecture to our problem. First, we modified the input layer to transform a Bayer frame into an RGGB tensor, as shown in Fig. V.1a. This tensor is passed to the convolutional network. Rather than using transposed convolutions in the decoder part, we preferred a bilinear upsampling followed by a convolutional layer. This is done to avoid checkerboard artifacts which were thoroughly investigated in [145]. We also chose a residual approach, meaning that the network outputs a prediction of the noise rather than the clean image. This technique helps to improve the convergence speed and was also used for FFDNet.

To train our network, we generated a dataset of 16000 RAW dead leaves patches of size (256,256). We also trained this network with the same number of patches taken from natural images of an 80% portion of the SIDD-Small dataset to compare natural and synthetic data. The other 20 % were used for our test set. To assess the truthfulness of our noise model, presented in Section V.3.1, we compare two trainings of the model on natural images: one using synthetic noise and the other with real noisy data. We optimize the L_1 loss during 1000 epochs with the Adam optimizer. The learning rate is 10^{-4} for the first 500 epochs, 10^{-5} for the following 300 epochs, and 10^{-6} for the last 200.

Results. To evaluate our models, we tested them on the remaining 20% of the SIDD small dataset. Since the network produces a RAW output, we process it with a simple pipeline proposed by the authors of the SIDD dataset. The proposed pipeline is made of a sequence of operations: white balance using metadata of the sensor, edge-aware demosaicing, a color space transform from RAW colors to sRGB, a gamma correction, and a simple tone-mapping function. Given this pipeline, we can measure PSNR in both the RAW and RGB domains.

The numerical results presented in Table V.1 show that training with dead leaves images and a synthetic noise model (DL-Unet) is as good as training with natural images and a synthetic noise model (RS-Unet). This further validates the hypothesis that synthetic data can be used to successfully train a complex image restoration network. More precisely: when testing on RAW images, the loss implied by using dead leaves instead of real images is only 0.25 dB. When evaluating the result on the developed RGB images, the dead leaves training outperforms the training on natural images, all other things being equal.

Further, we compare the performance of DL-Unet, the network trained on fully synthetic data, to the performance of the network trained with real noisy images (RR-Unet). The performance loss, in this case, is 0.8 dB in the RAW domain and 0.27 dB in the RGB domain. This shows that the full synthetic training is successful even though slightly less efficient than when using the ground truth. This loss in performance is of the same order as the loss when comparing RR-Unet and RS-Unet, showing that the loss is primarily due to the noise model and not the image model.

Now, we argue further that *the synthetic training can, in fact, outperform the natural training*. First, this behavior is observed in the RGB results just presented. Moreover, if we look at Fig. V.7, we notice that noise is still present in the "ground truth" image for high ISOs. In both examples, the details show that averaging 400 images is not sufficient to suppress

Table V.1: Numerical evaluation of the denoising of smartphone RAW images. We report the PSNR of our adapted Unet trained either on the dead leaves dataset or on the SIDD-small dataset with real or synthetic noise.

Dataset	SIDD-test RAW	SIDD-test RGB
DL images + synthetic noise (DL-Unet)	49.60 dB	38.05 dB
Nat. images + synthetic noise (RS-Unet)	49.85 dB	37.95 dB
Nat. images + real noise (RR-Unet)	50.40 dB	38.32 dB

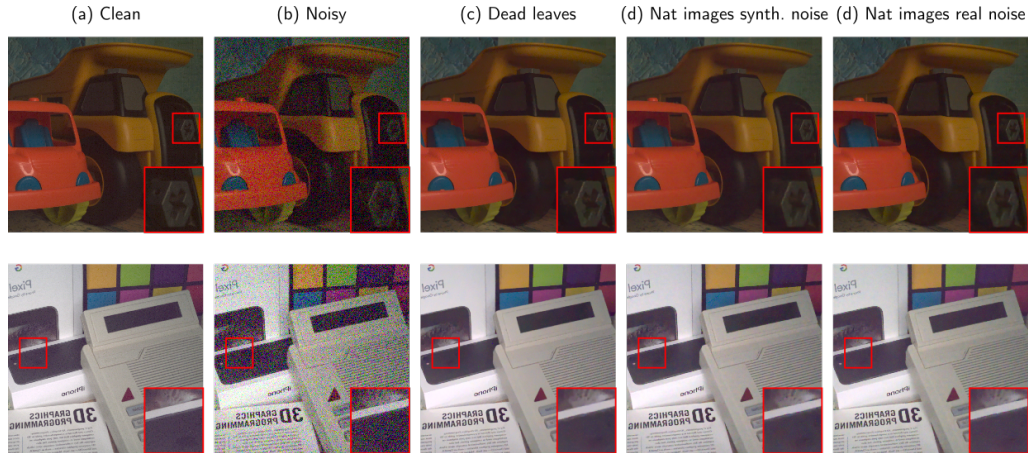


Figure V.7: Real denoising results.

noise entirely. The images denoised with DL-Unet appear cleaner in flat areas than the other denoised images, which still contain some noise. Indeed, the simulated dead leaves contain noiseless flat areas. Moreover, the highlighted detail of the first image also shows some structures that appear sharper in the images denoised with DL-Unet. Even if the quality of these images seems superior, this is not fully reflected by the quantitative comparison, mostly because our synthetic noise model is not perfect and also possibly because of the limitations of the PSNR to quantify image quality.

V.4.2 Low-light image enhancement

Network and Training. Eventually, we consider the task of denoising images taken in extreme low light. We considered the See-in-the-Dark dataset and chose to train an adapted version of a U-Net, similar to the one we used for the smartphone database. The RAW input is packed in an R-G1-G2-B tensor, and passed through the UNet, which uses bilinear up-sampling and convolutions rather than transposed convolutions in the decoder part of the network. To train our network DL-UNet, we generated 10000 RAW clean dead leaves images of size (256,256) and the corresponding noisy images using the synthetic noise model presented in Section V.3.2. The training algorithm is the exact same that was used for smartphone image denoising. To sample the noise parameters, we use the frequency of each ISO in the Sony database, given that each set of parameters corresponds to a unique ISO. We expect that by sampling the noise parameters this way, the network will be able to restore images properly at varying ISOs and to generalize well to the test set. For simplicity, the exposure time ratios are uniformly sampled between (100, 250, 300). To assess the validity of our noise model, we train the network also on clean natural images to which we add

our synthetic noise model(RS-UNet). We then compared these models to the original model trained on real noisy and clean image pairs (RR-UNet).

Results. We tested our trained models on the See-in-the-Dark test set, made of 94 RAW images at different ISOs and different time exposure ratios, with indoor and outdoor scenes. We report in Table V.2 the quantitative results of our tests.

Table V.2: Numerical evaluation of our low-light enhancement networks. We report the PSNR of our three different models at different time exposure ratios. In last three lines, we also report a corrected PSNR where we prescribe the real average of each channel to the output image.

ratio	100	250	300	Global
DL-UNet	28.90	26.86	26.62	27.41
RS-UNet	29.74	27.28	26.80	27.87
RR-UNet	29.81	28.55	28.21	28.83
DL-UNet (corrected mean) [*]	30.08 [*]	28.28 [*]	27.15 [*]	28.42 [*]
RS-UNet (corrected mean) [*]	30.18 [*]	27.97 [*]	27.32 [*]	28.42 [*]
RR-UNet (corrected mean) [*]	29.81 [*]	28.55 [*]	28.21 [*]	28.83 [*]

Regarding PSNR, we notice that the DL-UNet has a 1.4 dB difference from the RR-UNet, which is a significant margin. However, we notice that even RS-UNet has almost a 1dB gap with RR-UNet. This shows that our main limitation resides not in the image generation algorithm but in the distortion model. The noise model is very complex and difficult to model for high ISOs and extremely low-light indoor photographs. We also observed that the models trained with a synthetic degradation model sometimes fail to produce a properly white-balanced image, resulting in low PSNR scores. That is why we also report a corrected PSNR, where we prescribe the mean of each color channel of the output image to the mean of its clean long exposure counterpart. After applying this post-processing, we see that DL-UNet performs on par with RS-UNet and only 0.4 dB behind RR-UNet, which is a small margin. We believe this metric allows for a better assessment of the denoising quality of our model.

Visually the results of the DL-UNet are close and sometimes better than RS-UNet’s and RR-UNet’s. Looking at Fig. V.8a, we see that the details of the bike’s basket are sharper in the DL-UNet image than in the other restorations. In Fig. V.8b, we see that DL-UNet removes banding noise more effectively than RS-UNet in homogeneous areas while preserving sharp details in the vegetation area. In more extreme low-light conditions (see Fig. V.8c), we see that even if the result is close to the one of RR-UNet, DL-UNet tends to produce more color artifacts and does not preserve straight lines as RR-UNet or RS-UNet do. Our degradation model fails to mimic the sensor’s behavior in such conditions. Indeed, we notice in some images that the lens produces ring-like artifacts and that the bottom of the sensor is biased. Since we do not model these phenomena, our model fails to properly restore the image while creating artifacts.

V.5 Conclusion and Perspectives

In this chapter, we extended the use of dead leaves images to real-world image restoration for two applications: Smartphone RAW image denoising and low-light RAW image enhancement. To do so, we first adapted the dead leaves image generation algorithm to RAW images to generate ground truth synthetic images. We then proposed a carefully designed distortion

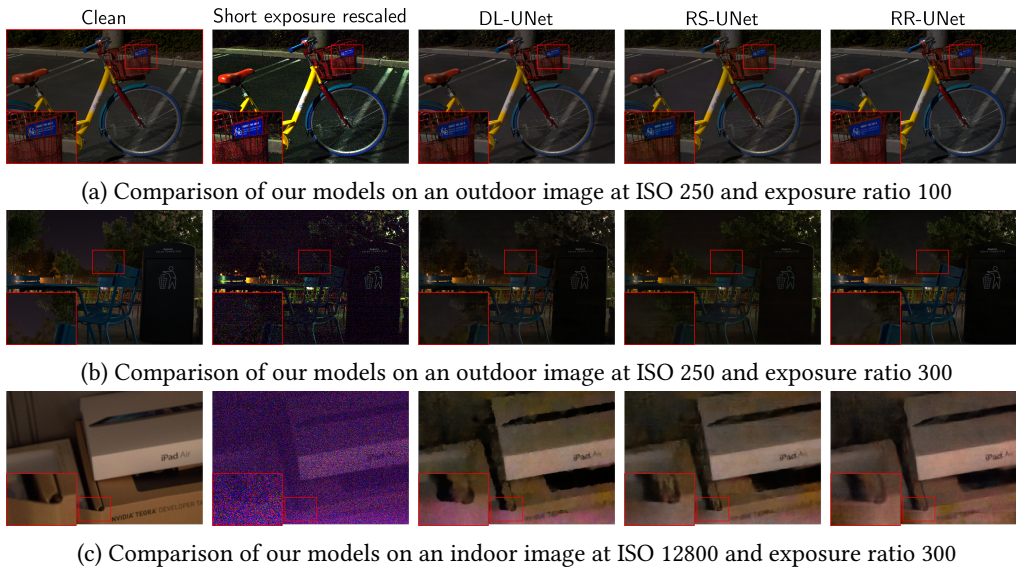
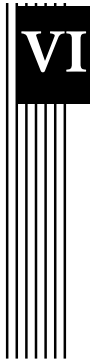


Figure V.8: See-in-the-Dark test examples

model equipped with a robust noise parameter estimation procedure. Given this synthetic image model and this synthetic distortion model, we could form an arbitrary amount of clean/distorted image pairs. With this dataset, we successfully trained an image restoration neural network for both tasks. For smartphone images, we perform on par with training with real data, confirming the validity of the distortion and image models. For low-light image enhancement, we saw that our model was sufficient for outdoor photography but limited for indoor extreme low-light photography. In that case, the number of photons is very limited, and the accuracy of the distortion model is crucial. We believe that our distortion model is not precise enough for these extreme conditions. This opens the door to improvements, even if the problem seems very hard.

To the best of our knowledge, this work is the first to propose a fully synthetic training for complex RAW image restoration tasks. The good performances we obtain mitigate the common idea that one needs a large dataset of paired data to train a RAW denoiser or an ISP CNN. In that sense, we believe that we can avoid heavy data acquisition campaigns saving time and effort. Moreover, synthetic data has the advantage of being truly noiseless, unlike processed ground truth images.

We can, however, mitigate our claim since we still use natural images to color the disks. According to the experiments of the previous chapter in Section IV.3, the color space is a crucial ingredient of the success of this model. Indeed, sampling the colors uniformly at random in the RGB cube leads to poor performances and color artifacts. In the following chapter, we address this problem by investigating how to summarize the distribution of color in natural images with a parametrized probabilistic model.



VI A study on the color distribution in natural images and an application to synthetic training of networks

VI.1 Introduction

Since the early studies of the human perception of colors led by the CIE in 1931, color has been at the center of attention in imaging sciences. With the popularisation of digital images, the focus has been even stronger. Digital sensors need to acquire color information and transform it into colors perceptually close to the scene photographed. This is done with complex operations described in Section II.1.1, including color filtering, color space transforms, and white balancing. All these operations were carefully fitted to human perception. The resulting colors are represented as a triplet of coordinates in what we will call *the sRGB cube*.

Interestingly, the colors of an image tend to be distributed along structured clusters which sparsely fill the sRGB cube. As a first attempt to study these structures, Omer and Werman first presented the Color Lines model [146]. This model assumes that monochromatic objects emit light following a curved line in the sRGB cube. With this model, the authors can accurately cluster the sRGB cube in pixels belonging to the same object, achieving much better results than clustering in other color spaces. The numerous applications include color compression, clustering, recolorization, etc. Building on this work, Lisani et al. argued that colors from a single object are distributed not on a line but rather on a 2D curved manifold [121, 27]. Even though both papers give us a sound model for the distribution of colors in a single image, they do not provide any statistical information about it.

We believe that a study of the first-order statistics of colors in natural images could complement our understanding of the distribution of colors made possible by the aforementioned body of literature. That is why we propose, in the present chapter to summarize the distribution of colors in natural images with a simple probabilistic model. In addition to providing insights about this distribution, this model allows us to efficiently sample natural colors without accessing natural images. As pointed out in the previous chapters, color

sampling is critical when training CNNs on artificial data. Therefore, we propose to generate dead leaves images based on this statistical model to train a RAW Smartphone Image denoiser. Since each camera sensor defines a unique RAW color-space, we propose a mapping operation to a quasi-universal RGB color space. This camera-specific operation allows to estimate the distribution on a database acquired with different cameras, and to generate dead leaves RAW images for cameras absent from the database. The denoising results show that the model trained with this simple color model performs almost on par with the model trained from real colors.

The present chapter is organized as follows: we first give a brief overview of the literature covering color spaces and the distribution of colors. In Section VI.3, we thoroughly explain our color distribution models, the estimation of their parameters, and the proposed sampling techniques. We then present in Section VI.4 the results of our main application: training an image restoration network from synthetic dead leaves images colored with our sampling procedure.

VI.2 Background

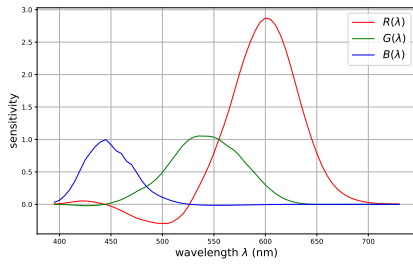
VI.2.1 Some color spaces

The perception of color by the human visual system is one of the most crucial aspects of digital imaging. The acquisition and the display of colors are indeed tuned to fit the human perception of a scene. We will first describe some color spaces that will be used hereafter, and how we can approach most visible colors with a simple screen. Then we will see how a camera acquires color information, and how it is transformed in a viewable color that looks natural.

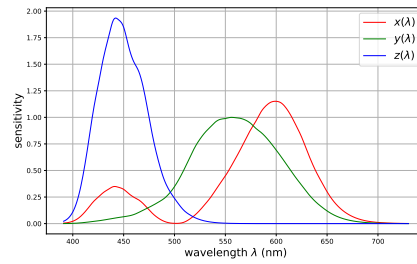
According to Grassmann laws, every pure color of the visible spectrum, i.e. with a single wavelength, can be obtained by a linear combination of three pure and independent light sources, called primaries. In practice, the primaries are chosen to be red (700nm), green (546nm) and blue (435nm). Based on this tristimulus color theory, the CIE (commission internationale de l'éclairage) attempted to understand which combination of primaries were needed to approximate the whole visible spectrum. To do so, Wright and Guild led tests on a total of 17 standard observers. For each pure color, each observer had to match the linear combination of primaries with the pure color, by adjusting the intensity of the primaries. The found coefficients were averaged to create the color matching functions (CMF) we can observe in Fig. VI.1a. For some wavelengths, no linear combination matched the perceived color. In these cases, one of the primaries was added to the tested color, which resulted in some negative values. Based on this data, the CIE proposed in 1931 a canonical representation of colors, called the CIE-XYZ color space. In particular, the desired properties were:

- the white point should be represented by the point $(1/3, 1/3, 1/3)$,
- the Y value should represent the luminosity,
- all coordinates should be positive.

To respect these properties, the CIE proposed a linear transform of the CIE-RGB done by a 3 by 3 matrix multiplication. The transformed color matching functions are reported in Fig. VI.1b.



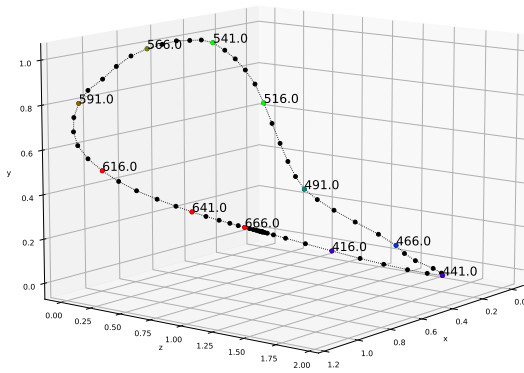
(a) CIE-RGB color matching function



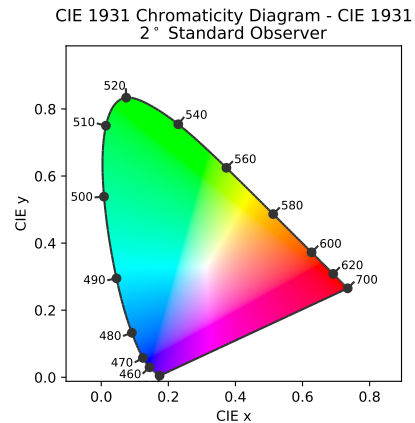
(b) CIE-XYZ color matching function

Figure VI.1: Color matching functions from the CIE experiments of 1931.

We can now represent each visible color in a 3D space, delimited by the envelope of pure colors. It is represented in Fig. VI.2a. Since this shape is not very informative on the perceivable colors, a 2D visualization is preferred. To do so, each point of the XYZ color space is first mapped on the plane $x + y + z = 1$, by dividing each point by the sum of its coordinate. Then we discard the z component, to obtain the CIE-xyY chromaticity diagram presented in Fig. VI.2b. This diagram describes the whole range of perceivable color for the human eye. The CIE XYZ color space is therefore a universal color space, and does not depend on a particular imaging or emitting device.



(a) 3D representation of the CIE-XYZ color space



(b) XyY color space of perceivable colors

Figure VI.2: Chromaticity diagram from the CIE-XYZ color space

Based on this color space and on the tristimulus color theory, the standard RGB (sRGB) space was developed by the CIE as a standard format to store and display colors. In the sRGB color space, each color is a linear combination of 3 primary colors in the chromaticity

diagram: a red (612nm), a green (547nm) and a blue (464nm). Therefore, the sRGB color space can only partially represent all perceivable colors. The visible colors with the sRGB color space is represented as a triangle in the chromaticity diagram. The triangle covers roughly 40 % of the diagram, discarding many perceivable colors. However, the sRGB space was designed so that the discarded colors are colors for which the human eye is not very sensitive to differences. It also focuses on colors for which the human eye is very sensitive to differences. Finally, a Gamma transform is applied to colors so that the colors fit the human perception of contrast.

When it comes to acquiring color information, camera color sensors mimic the behavior of the human cone cells of the retina. Most sensors are made with a Bayer color filter array, which we showed in Chapter II. Each photo-site of the sensor is covered with a color filter, only allowing a specific range of visible wavelength. The sensitivity functions of these color filters are camera-dependent, even if they are all quite close to the sensitivity of the cone cells. Therefore, each camera sensor defines a unique RAW-*RGB* color space. In order to visualize these colors, they undergo many transforms so that they become close to the perceived scene's colors. The first transform is a white balance operation. This operation estimates the temperature of the illuminant, based on a prior such as the white-patch or the grey-world. Based on this estimation, each channel is multiplied by a computed factor. Then the colors are mapped to the CIE-*XYZ* color space. This is done by a $(3, 3)$ matrix multiplication of all colors. This matrix depends on the sensor and the estimated illuminant. The camera manufacturer computes this matrix for a warm and a cold illuminant in a controlled setting. Based on the estimated illuminant, the color transform matrix is an interpolation between these two pre-computed matrices. Therefore, the RAW-*RGB* to CIE-*XYZ* conversion depends both on the camera, and on the illuminant value found during the white balance. Once the colors are mapped to the CIE-*XYZ* color space, they can be visualized on a screen with a *XYZ*-to-sRGB transform. This transform amounts to a $(3, 3)$ matrix multiplication and a gamma correction.

VI.2.2 Modeling the distribution of colors in natural images

As mentioned previously, relatively few works have investigated the distribution of colors of natural images. Early works [[ohta1980color](#)] have shown that a principal component analysis on natural images yields opponent-like color spaces, a hypothesis that was later theoretically investigated in [[buchsbaum1983trichromacy](#)]. Still dealing with first order statistics, empirical histograms of relatively large databases have been collected in [[mazin2014methodes](#)]. Spatial dependency between colors have been investigated by [[ruderman1998statistics](#)] and the corresponding oscillatory patterns of opponent colors have been theoretically justified in [[provenzi2016second](#)]. Several color statistics have been investigated in [[golz2002influence](#)] in view of a better understanding of the color constancy ability of the human visual system. In 2004, [146] introduced the color line prior, an attempt to model the distribution of colors from a single monochromatic object. The authors explain that the colors of a single Lambertian object (i.e. an object which surface diffuses light rays uniformly in all directions), are distributed along a straight line in a RAW-

*RGB cube*¹ starting at $(0, 0, 0)$. The sensor of a camera has indeed a linear response to the number of photons. While RAW data is more suitable to analyze the physical world, it is not always accessible and it also does not correlate with the human perception. For these reasons, colors are usually represented and analyzed in the sRGB cube. The RAW-to-sRGB transform is nonlinear, modifying straight lines into curved ones. Other operations in the processing pipeline of a RAW image are non linear, such as tone mapping for instance. Moreover, the physical properties of the object also distort its color representation. First, objects can be specular. Because of this, the sensor's response to these object is saturated and therefore nonlinear. This implies a T-shaped color cluster. Second, rough objects are far from Lambertian as they emit light in a non uniform manner. This results in spread color clusters, which is a more similar to a surface than a line. The resulting color lines in the sRGB cube are therefore curved (see Fig. VI.3).

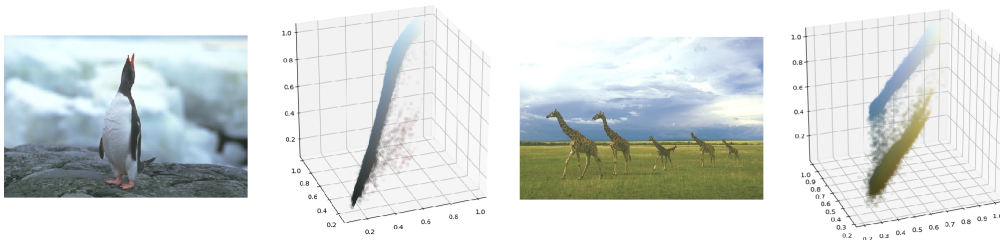


Figure VI.3: Images and their corresponding color point cloud in the sRGB cube

Based on the hypothesis that single objects follow curved lines in the sRGB cube, the authors derive a clustering algorithm for the pixels of an image. In short, they compute spherical slices of the sRGB cube. By detecting the local maxima of the histogram in these slices, they can extract points of different color lines. The same procedure is repeated for successive slices, and points are then connected from one slice to the next, forming color lines. The colors of the images are then appointed to the cluster they are the closest to. The authors show that this clustering is more efficient than clustering in any other color space, whether it is CIE LAB or HSV. This color representation opens the door for better compression of color images, easy color editing of objects, and new clustering algorithms.

The Color Line method was further used for dehazing [56], outperforming the at-the-time state-of-the-art Dark Channel Prior method by He et al. [76], based on Physical modeling of haze and guided filtering. The authors observe that color lines on hazy images are always shifted from the origin and propose an algorithm to correct the angle of the found color line.

Later, Buades et al. argued that the colors from a single object were distributed not along a 1D curved line but on a 2D structure [27]. The argument is motivated by an analysis of cases for which the 1D approximation is insufficient. For Lambertian objects, the hypothesis still holds. However, most objects are specular to some extent. Moreover, a single object with multiple orientations reflects light with a different intensity for each facet. If a single object

¹The RAW-*RGB* cube can be defined as the cube where each point coordinates corresponds to the associated *RGB* values in the RAW image.

has two facets with different orientations, it will have two line clusters in the sRGB cube, assuming each facet is Lambertian. When the orientation changes are smooth, the color cluster corresponding to this object will be distributed along a surface. The same reason explains why the more a material is rough, the more its color distribution will be spread as a 2D surface in the sRGB cube. Roughness can indeed be thought of as the amount of change in the orientation of micro-facets at the surface of an object. Following this idea, the authors developed a dimensionality reduction algorithm for the representation of an image in the sRGB cube, for both 1D and 2D structures. The experiments show that the 2D version of their algorithm leads to a much better reconstruction of the color clusters and a better preservation of the original colors. Lisani et al. then proposed an improvement of this method, allowing for better visualization of the surfaces in the color cube [121].

Even though the 2D surface assumption is more precise for the approximation of the color distribution of natural images, it is much more difficult to extract first order statistics corresponding to this model. Conversely, the Color Line model is much simpler, and the directions of the clusters are easy to retrieve. One of the color sampling algorithm proposed in the foregoing sections will be built on this prior.

VI.3 A density estimation of the natural image color distribution

Following the observations in [146], we chose to estimate the distribution of colors on RAW images. The nonlinearities introduced in the development of sRGB images make the distribution of colors in the sRGB cube more complex. These operations are also camera-specific, and estimating the distribution on sRGB images would necessarily introduce a bias for each camera. Therefore, we chose to use the RAISE dataset [45] for our estimation, which is a large set of RAW images from different cameras, containing both indoor and outdoor photographs, as we can see in Fig. VI.4.



Figure VI.4: Examples of photographs from the RAISE dataset

Even though the developing operations are camera specific, so are the sensitivity functions of the color filters for each camera sensor. Without modification, each camera would have its own identifiable modes in the empirical distribution. To counter this, we propose to map the colors to a quasi universal color-space. To do so, we first extract each camera's daylight white balance multipliers. Then, we multiply the color channels of each image by the

multipliers corresponding to the camera used for this image. These multipliers are indeed all estimated by camera manufacturers to fit a scene with the standard D65 illuminant.

More precisely, we have access to a dataset of RAW images $X_{i,c}$ from C different cameras $\mathcal{D} = \bigcup_{c \in [0, C]} D_c$ where $D_c = \{X_{i,c}\}_{i \in [0, N_c]}$. Each image is first packed in an RGGB tensor $T_{i,c}$, as explained in Fig. V.1a in the previous chapter. Since there are two channels for the green component, they are simply averaged. The resulting tensor $\tilde{T}_{i,c}$ is multiplied by the daylight camera multipliers d_c , thus obtaining a quasi universal representation $T_{i,c}^* = d_c^T \cdot \tilde{T}_{i,c}$. We will present next two different models for the distribution of colors of natural images, estimated thanks to the normalized data:

$$\mathcal{D}^* = \bigcup_{c \in [0, C]} \{T_{i,c}^*, c\}_{i \in [0, N_c]}.$$

VI.3.1 A 3D Gaussian mixture model

The pixel's colors in the set of images \mathcal{D}^* all live in a universal space which we will call the *universal-RGB cube*. To better understand the color distribution, we can fit a 3D Gaussian mixture model on the point cloud of colors in \mathcal{D}^* . Gaussian Mixture Models (GMM) can indeed approximate very complex distributions as long as they comprise sufficient Gaussian components. For each point $x \in \mathbb{R}^3$, the density function of this model is expressed as the following sum:

$$f(x) = \sum_{k=1}^K \frac{\alpha_k}{(2\pi)^{3/2} \det(\Sigma_k)^{1/2}} \exp \left[-\frac{1}{2} (x - \mu_k)^T \cdot \Sigma_k^{-1} \cdot (x - \mu_k) \right],$$

where K is the number of Gaussian components. Each of them is defined with a weighting parameter α_k and a mean and covariance of the Gaussian (μ_k, Σ_k) . All these parameters are fitted to the distribution thanks to the Expectation Maximisation algorithm (EM). This algorithm aims at finding the parameters that maximize the likelihood of the model. The algorithm stops when it has converged, i.e., the likelihood of the model does not increase anymore.

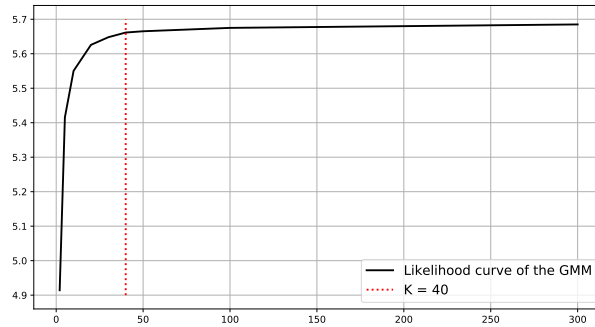


Figure VI.5: Graph of the likelihood of the 3D GMM against the number of components

Theoretically, the more components there are, the better the model will fit the data. However, the relative improvements brought by adding more components can be marginal. We indeed notice, in Fig. VI.5, that the likelihood of the model reaches a plateau at $K = 40$.

Our 3D GMM model will therefore have 40 components, being a good trade-off between the simplicity and correctness of the model. Each Gaussian component of the GMM has ten parameters, bringing the total number of parameters to 400 for the whole color model. Visually, the 3D point cloud obtained by sampling this model is indistinguishable from the actual distribution of the colors in the universal-RGB cube(see Fig. VI.6).

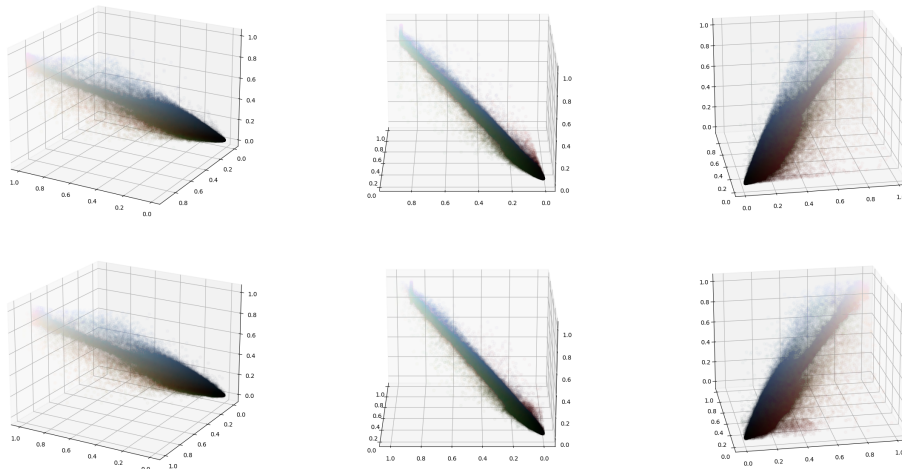


Figure VI.6: Different views of the point cloud of natural colors in the universal-RGB cube. The top row is the real distribution, the bottom row is the GMM approximated distribution

In order to color a dead leaves image from this model, we will simply sample the optimized 3D-GMM. The color obtained will necessarily be natural colors, which will lead to much better results than sampling the universal-RGB cube uniformly at random, based on the experiments shown in Section IV.3.1. However, this model only accounts for the distribution of the colors in the whole image dataset. Now, recall that the synthetic dead leaves images used to trained CNNs in the previous sections are colored using colors sampled from a single natural image, different for each synthesized image. This results in synthetic images having much more coherent colors and contrast that when sampling from the complete database. Next, we present a color model which allows us to sample the colors more appropriately by trying to mimic the single-image-based sampling.

VI.3.2 A decoupled color/luminance model

VI.3.2.1 2D color distribution

Recall that in the *Color Line* paper [146], the authors argue that the emitted surface color of a Lambertian object is a multiplication of the surface albedo with the illumination. Given this assumption, the ratio between color components should be constant. Therefore, the pixels representing a single Lambertian object in a RAW image all live on a single line starting from $(0, 0, 0)$ in the universal-RGB cube. The direction of this line contains the color information of that object.

Using this property, we can study the distribution of all possible directions in the dataset of RAW images. For each color triplet $(r, g, b) \in \mathcal{D}^*$ we apply the simple transform $\tilde{r}, \tilde{g}, \tilde{b} =$

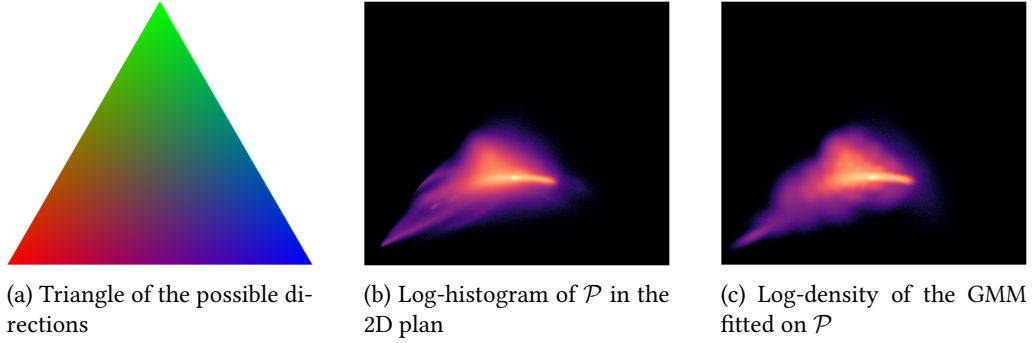


Figure VI.7: Distribution of natural colors in the 2D representation

$\frac{1}{r+g+b}(r, g, b)$. The transformed values all live in the intersection of the universal-RGB cube and the plan $x + y + z = 1$. This 2D surface is a triangle whose vertices are the pure color components. This triangle defines a range of colors shown in Fig. VI.7a).

Since this is a 2D surface, it is reasonable to define the plan's orthonormal basis to work with 2D values. We chose the following orthonormal vectors, $e_1 = \frac{1}{\sqrt{2}}(-1, 0, 1)$ and $e_2 = \frac{1}{\sqrt{6}}(-1, 2, -1)$ to create this basis. Then, we project every normalized triplet on the orthonormal space defined by this basis. We are left with the dataset of 2D points:

$$\mathcal{P} = \left\{ (x_i, y_i) \mid x_i = e_1^T z_i, y_i = e_2^T z_i, z_i \in \mathcal{D}^* \right\}.$$

We report in Fig. VI.7b, the 2D log-histogram of \mathcal{P} . The distribution indeed peaks on its central mode, corresponding to grey colors. To better visualize the weakest values of the distribution, we display the logarithm of the histogram. The histogram is an accurate statistic to approximate the distribution, but it is still over-parametrized, having more than a million bins. To reduce this number, we use a 2D GMM, similar to what was done for the 3D distribution estimation.

Similar to the previous experiments, we show that the GMM's likelihood reaches a plateau as we add more and more components. This behavior is shown in Fig. VI.8. We found that $K = 40$ is the optimal number of components. For each of them, there are six parameters, resulting in a color model with only 240 parameters. In Fig. VI.7c, we report the log density of the fitted GMM. It is visually very close to the actual histogram of the data \mathcal{P} , confirming that the GMM can be used to summarize this distribution.

Given this model of the color distribution, our goal is to create a color sampling algorithm based on the properties presented in the *Color Line* paper. Unlike the previous 3D model, the luminance information is lost in the 2D model. We assume the luminance depends on the object's chromaticity, which, in our framework, is given by the 2D coordinates. Therefore, in the following section, we study the luminance distribution conditioned to the object's chromaticity.

VI.3.2.2 A color sampling algorithm

Data Analysis. In order to understand the luminance distribution conditioned on chromaticity, we analyzed the RAW images of the RAISE dataset. Our strategy is to analyze the

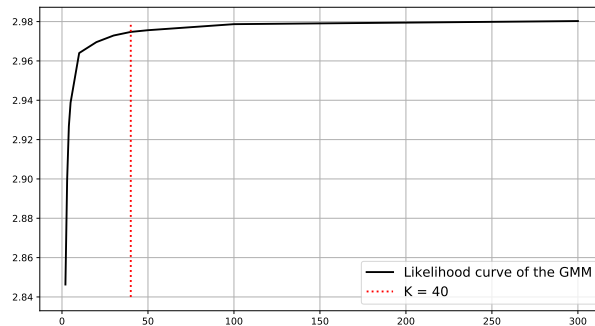


Figure VI.8: Graph of the likelihood of the 2D GMM against the number of component

distribution of the grey levels in patches of the dataset whose chromaticity is homogeneous. The underlying assumption is that patches with a homogeneous chromaticity belong to a single object. Therefore, we first start by extracting all the disjoint $(50,50)$ patches of the set \mathcal{D}^* . For each patch, we computed its average grey level \bar{z} , its average chromaticity (\bar{x}, \bar{y}) , and the chromaticity's covariance $\Sigma_{x,y}$. In order to only analyze the distribution of grey levels for almost monochromatic objects, we filter out the patches with an excessive covariance $\Sigma_{x,y}$, by simply thresholding the determinant. Then, we binned the grey level mean values \bar{z} by similar (\bar{x}, \bar{y}) value. After visualizing the histogram of the grey level mean in each bin, we noticed it had a heavy-tailed distribution, which we could model with a Gamma distribution. We report in Fig. VI.9, some examples of the distribution of the grey level mean depending on the (\bar{x}, \bar{y}) position.

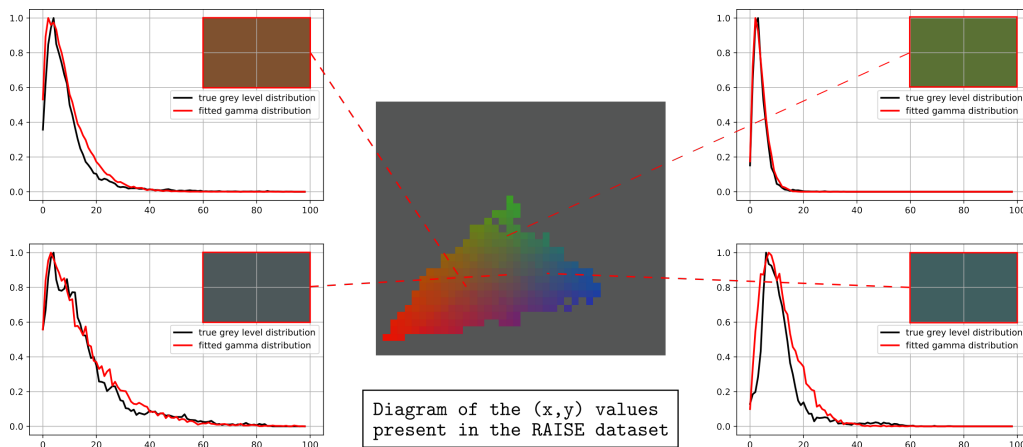


Figure VI.9: Distribution of the average grey level knowing the position (x, y) in the 2D color representation.

In the given examples, we see that the parameters of the Gamma distribution depend on (\bar{x}, \bar{y}) . We notice that the closer we are to a grey color, the more the distribution is spread. In fact, the further the chromaticity (x, y) is from the central "grey" direction, the smaller the maximal grey level value is. As a simple approximation, we consider that the distribution of the average grey level can be expressed as:

$$z|(x, y) \sim \lambda(x, y)\Gamma(\theta_{\text{grey}}),$$

where θ_{grey} are the Gamma distribution’s parameters found for the central bin of the triangle, which corresponds to grey colors. $\lambda(x, y)$ corresponds to the maximal grey level attainable for the chromaticity (x, y) . Given (x, y) , we can easily retrieve a color triplet $(r, g, b)_{x,y}$ in the plane defined by $u + v + w = 1$. $\lambda(x, y)$ is then empirically defined by the following formula:

$$\lambda(x, y) = \frac{1}{3\max((r, g, b)_{x,y})}.$$

Color Sampling algorithm. We can now easily derive a point cloud generation algorithm for a single object, thanks to our models of $p(x, y)$ and $p(\bar{z}|x, y)$. The complete procedure is given in Algorithm 3. Following the *Color Line* prior, the colors of a single object are distributed along a line in the universal-RGB cube, defined by a single chromaticity value (x, y) . Therefore, we can sample an average chromaticity $\mu = (\bar{x}, \bar{y})$ using our 2D GMM for a single point cloud. However, having only a single chromaticity would be unrealistic. An object is indeed not precisely monochromatic. That is why we sample each point’s chromaticity following a 2D-Gaussian distribution centered on the previously sampled μ . This sampling results in the set $C = \{(x_i, y_i) \sim \mathcal{N}(\mu, \sigma^2.I)\}_{i \in [1, \dots, N]}$, where N is the desired number of points. The standard deviation σ follows a uniform distribution $\sigma \sim \mathcal{U}([5.e^{-5}, 5.e^{-4}])$, to increase diversity.

We can now sample the luminance average \bar{z} conditioned on μ , with the Gamma law defined above. Many parametric grey-level distribution models are possible given \bar{z} . For simplicity, we propose to simply model it by a Gaussian distribution $\mathcal{N}(\bar{z}, \beta^2)$, where $\beta \sim \mathcal{U}([0.05, 0.3])$.

Algorithm 3: Sampling colors for a single object.

```

Input :  $N$ 
Output:  $(r_i, g_i, b_i)_{i \in [1, \dots, N]}$ 
 $\hat{x}, \hat{y} \sim GMM$ 
 $\bar{z} \sim \lambda(\hat{x}, \hat{y})\Gamma(\theta_{\text{grey}})$ 
 $\beta \sim \mathcal{U}([0.05, 0.3]), \sigma \sim \mathcal{U}([5.e^{-5}, 5.e^{-4}])$ 
for  $i \in [1, \dots, N]$  do
     $x_i, y_i \sim \mathcal{N}(\mu, \sigma^2.I)$ 
     $z_i \sim \mathcal{N}(\bar{z}, \beta^2)$ 
     $r_i, g_i, b_i = 2D\_TO\_RGB(x_i, y_i)$ 
     $r_i, g_i, b_i = \frac{z_i}{r_i + g_i + b_i}(r_i, g_i, b_i)$ 
end
    
```

The described algorithm allows us to sample colors for a single monochromatic object by following the Color Line Prior, defined by Omer et al. [146]. As we saw in the experimental section of Chapter IV, it was critical to sample colors in a single natural image for each image synthesis. The dead leaves model equipped with this coloring algorithm allowed for the apparition of transitions between objects of different chromaticity as well as monochromatic and poly-chromatic textures. These properties were the key ingredient for the good restoration of colors. We wish to approximate them with our sampling algorithm. First, in order to simply mimic transitions between objects of different chromaticity, we decided to color each dead leaves image from two different color clusters. By generating a large amount of images, we hope that the dataset shows enough variety in the color transitions and poly-chromatic

textures. In Fig. VI.10, we give examples of the obtained point clouds. Second, in order to mimic monochromatic textures, we chose to synthesize another set of dead leaves images with only one color cluster. We believe this simple color sampling method is sufficient to model the interactions between objects and monochromatic textures.

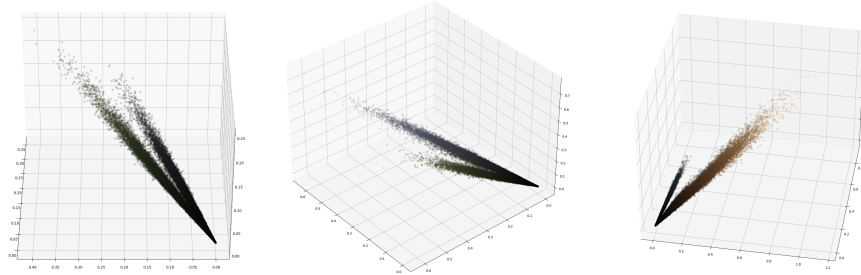


Figure VI.10: 3 different color points cloud for dead leaves image generation

VI.4 Training a denoising network for smartphone RAW images

Given the previously described color models, we now use them to synthesize dead leaves images in order to train a RAW image denoiser. More precisely, here are the different color sampling strategies (for each single image generation):

1. sampling colors from one single real image,
2. sampling colors from the 3D histogram approximated by a Gaussian mixture model,
3. sampling colors from a single color line to model monochromatic texture images,
4. sampling colors from two different color lines in order to model interactions between objects of different chromaticity.

We will focus on Smartphone RAW image Denoising on the SIDD benchmark dataset [2], comprising noisy images from 5 different smartphones. A few adaptations are nevertheless needed to generate RAW images from the given model, which we describe next. We then report the obtained denoising results after training.

VI.4.1 Dead leaves image generation specifics

Dead Leaves datasets. In order to have as much diversity as possible in the color profiles of the generated images, we generate much more images than in the previous chapter, for which we had 16000 (512,512) dead leaves RAW images. Instead we chose to generate 60000 (256,256) dead leaves RAW images. With the same memory footprint we can have much more diversity in the colors generated.

To compare the different color models, we generate four datasets with the same geometry hyper-parameters:

- **3DM-DL**: this dataset is generated with colors coming from our 3D GMM model (strategy 2), fitted on the colors of the RAISE set (see Fig. VI.11a),
- **2DM-DL**: for this set, the color model is the decoupled grey-scale/color model following the color line prior (see Fig. VI.11b). In this case, both strategies 3 and 4 are used (evenly split).
- **Nat-DL**: here, the colors are sampled in the colors of the SIDD dataset (see Fig. VI.11c), following the first sampling strategy.
- **RAISE-DL**: here, the colors are sampled in the universal-RGB colors of the RAISE dataset (see Fig. VI.11c), following the first sampling strategy. We included this dataset to assess if the limitation came from our approximation of color lines, or from the discrepancy between the two different datasets.

The other hyper-parameters are the same as in the previous chapter, that is $\alpha = 3$, $r_{max} = 2000$, $r_{min} = 4$ for half the images and $r_{min} = 100$ for the other half. We apply Gaussian blur to 1/3 of the images with a standard deviation sampled in $[1, 3]$.

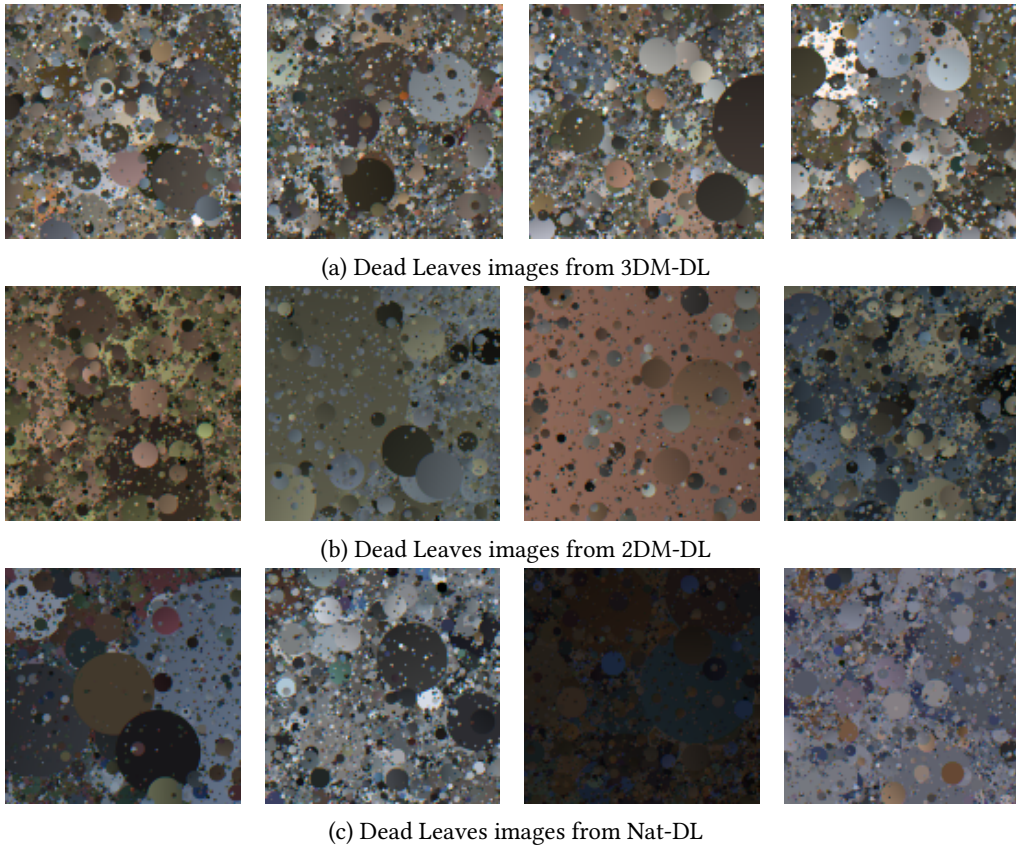


Figure VI.11: RGB visualization of generated RAW Dead Leaves patches with different color models.

Universal RGB to RAW color inversion. Whether it is for images generated with colors from the 3D GMM or the decoupled model, the generated colors live in a universal color representation space, agnostic of the camera, where red green and blue values are more or less balanced. These colors are not at all representative of the RAW colors for specific smartphone cameras. In order to transform the generated colors to the space of RAW colors of a given camera, we need to invert the daylight white balance multiplication with the camera-specific parameters. Since these parameters were not available in the SIDD database for each camera, we decided to approximate them with random values sampled around realistic average estimates, based on the white balance parameters available in the dataset. Specifically, we chose to model the red and blue gain with uniform distributions, $g_r \sim \mathcal{U}([1.9, 2.4])$ and $g_b \sim \mathcal{U}([1.5, 1.9])$.

Given these parameters, we can simply divide the red channel by the sampled gain g_r and the blue channel by g_b . With this technique, all the obtained images would necessarily miss values over $1/g_r$ or $1/g_b$ in the red and blue channels, which would be unrealistic. To preserve some of these higher values, we perform the color gain inversion technique presented by Brooks et al. [24] used to unprocess RGB images in RAW images. This transform is linear for $x < t$ where $t = 0.9$ and cubic for $x \in [t, 1]$, therefore preserving some higher values. More precisely,

$$\alpha(x) = \left(\frac{\max(x - t, 0)}{1 - t} \right)^2 \quad (\text{VI.1})$$

$$f(x, g) = \max\left(\frac{x}{g}, (1 - \alpha(x))\frac{x}{g} + \alpha(x)x\right) \quad (\text{VI.2})$$

We give in Fig. VI.12 some examples of these gain inversion functions for different gain values.

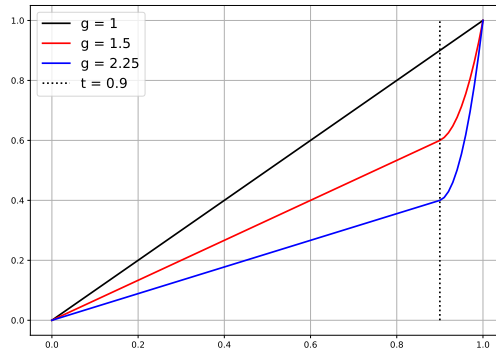


Figure VI.12: Color gain inversion functions for different gain values

VI.4.2 Denoising result.

As in Section V.4.1, we train for each dataset a modified version of the U-net for RAW image restoration with residual learning and bilinear upsampling. The training procedure is a bit

modified, since the datasets have much more images. We train the network for 500 epochs, with an initial learning rate of 10^{-4} for the first 250 epochs, a 10^{-5} for the 150 following epochs, and a 10^{-6} learning rate for the final 100 epochs. We use the ADAM optimizer with a batch size of 64. The testing procedure, on the other hand, remains unchanged. Looking at the numerical results reported in Table VI.1, we observe that for all scores, 2DM-DL outperforms 3DM-DL. This result confirms that our color sampling algorithm based on the Color Line prior is more realistic than sampling from a GMM fitted on the distribution of colors in the RGB cube of the whole dataset.

Table VI.1: Numerical evaluation of the denoising of smartphone RAW images. We report the PSNR and SSIM the denoising models trained on dead leaves images obtained with different coloring strategies.

Dataset	SIDD-test RAW	SIDD-test RGB
Nat-DL images	49.94dB / 0.9854	37.85dB / 0.9283
2DM-DL images	49.62dB / 0.9836	37.36 dB/ 0.9234
3DM-DL images	49.30dB / 0.9827	37.24 dB/ 0.9228
RAISE-DL images	49.68dB / 0.9841	37.56dB / 0.9312

Looking closer at visual results, we can see in Fig. VI.13, that the 2DM-DL model achieves much better results in low-light conditions than the 3DM-DL model. For the first image, the texture is completely blurred out by the 3DM-DL model, whereas the 2DM-DL model restores it as precisely as the Nat-DL model. We can explain this result by the fact that images generated with the 3D-GMM colors always have the same contrast properties. Each image indeed has roughly the same color distribution, with the same amount of high-lights and low-lights. Conversely, the images obtained with our decoupled sampling technique are much more diverse. Some of these images have indeed a very low contrast, being made of only one line cluster. When dealing with well-lit images, the performances, are similar with a slight advantage to 2DM-DL for low contrast textures.

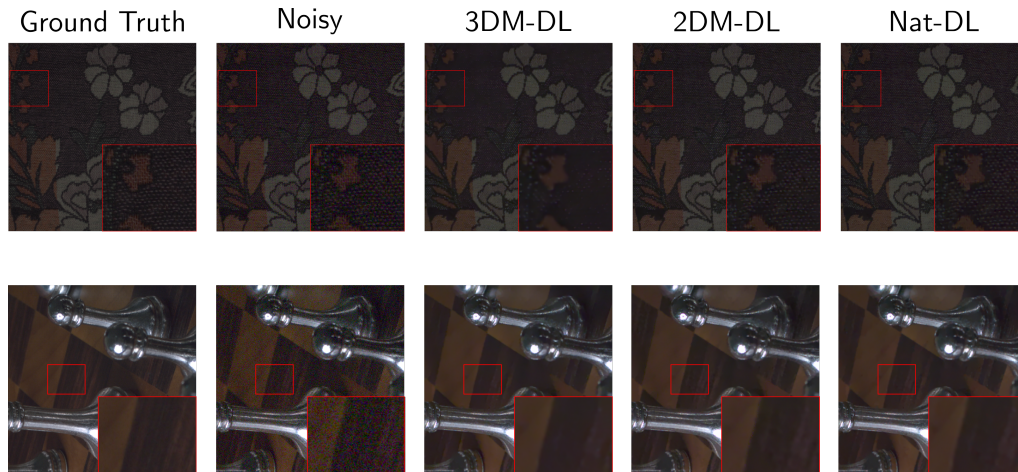


Figure VI.13: Comparison of the different denoising models. From left to right: ground truth image, noisy image, 3DM-DL denoising, 2DM-DL denoising, Nat-DL denoising.

Sampling colors in natural images still produces better numerical results. This can be explained by multiple reasons. First, the color distribution is estimated on a dataset with very different color content to the test set. The RAISE dataset indeed contains a majority of

outdoors color images, with unsaturated colors and similar illuminants. On the other hand, the SIDD dataset is a set of indoor images with a lot of unusual color palettes and a wide variety of illumination conditions. We can see in Fig. VI.11c that the generated images look like they have a single illuminant with the same temperature and intensity.

Moreover, the dataset is obtained from a camera that is different from the test cameras. We addressed this problem by mapping the colors of the RAISE dataset in an approximated universal color space, by simply multiplying the colors by the daylight multipliers, before estimating the distribution. We supposed that this distribution was roughly the same for every camera. The color filter response functions differ from one sensor to another, and approximating the relationship between these functions by a linear transform may be too much of an approximation. In addition to this, we did not have access to the daylight white balance parameters for the testing cameras, so we had to model them by sampling around plausible values.

In order to assess the loss in performances caused by these discrepancy between the RAISE and the SIDD datasets, we trained the denoising network on the RAISE-DL dataset presented above. We report, in Table VI.1, a loss of 0.26 dB in the RAW domain and 0.29 dB in the RGB domain for the PSNR on the test set. The RAISE-DL results constitute an upper-bound for the results of the approximated sampling algorithms, since the colors of the leaves are sampled with real point-clouds.

Surprisingly, the gap in performances between the RAISE-DL method and the 2DM-DL method is relatively small, with less than 0.1 dB in the RAW domain and 0.2 dB in the RGB domain. We remind the reader that the network was not trained in the RGB domain. This results confirms that our 2DM color sampling technique is valid, and that the performance gaps are mainly due to the differences of the dataset’s distribution.

Despite these limitations, the gap between Nat-DL and 2DM-DL is not large: 0.3 dB in the RAW domain and 0.5 dB in the RGB domain (which the network was not trained for). Visually, the results are very close, as we can see in Fig. VI.13. We considerably reduced the amount of color parameters of our model while preserving satisfactory denoising performances.

VI.5 Conclusion

In the present chapter, we first conducted a study of the first-order statistics of the colors in natural images. We observed that the natural colors only sparsely occupy the universal-*RGB* cube, which we defined in Section VI.2.2. This distribution can indeed be approximated accurately with a 3D Gaussian mixture model. Our initial goal was to provide a color model for synthetic images. Based on the experiments of Chapter IV, we know that a realistic color profile is a crucial ingredient for the success of deep image restoration methods. Since we couldn’t generate point clouds similar to those of a single image with the 3D model, we decided to study a decoupled version of the color distribution with a chromaticity and a luminance component. Based this decoupled color model and the color line prior [146], we created a color sampling algorithm which generates realistic color point clouds in the universal-*RGB* cube. Given these new color sampling strategies, we were able to generate different datasets and train a denoising network on them. The results obtained confirm the

validity of our color sampling algorithm based on the color line prior. In terms of PSNR, our model is only 0.3 dB behind the previous data-dependant version, in the RAW domain. This performance gap is mainly due to the discrepancy between the dataset on which we estimated the distribution and the test-dataset.

In order to reduce this gap, an interesting strategy could be to refine the transform operation from universal colors to camera-specific RAW colors. As a first approximation, we multiplied each color by a diagonal matrix, which coefficient were sampled in a plausible interval. A potentially better way to do this would be to estimate a full color correspondence matrix between two cameras.

Overall, the present chapter shows that we can efficiently train an image restoration neural network without using any information about a dataset of real images. Moreover, we drastically reduced the number of parameters for the color sampling technique to around 200 parameters, making our synthetic image model simpler, and universal. In the next chapter, we will present a new way to better exploit dead leaves images, by enforcing a good frequential response during training. This should, hopefully, improve the perceptual quality of restored images.



A perceptual loss to improve texture preservation

VII.1 Introduction

Image quality is one of the most important criteria for camera manufacturers and consumers. In order to quantify image quality fairly, independent evaluation agencies perform a wide range of standard tests. Thanks to these tests, one can rank cameras regarding a specific criteria. Among them, some evaluate the noise reduction capacity, the demosaicking quality, the amount of chromatic aberration, etc. These tests are quite informative, for the consumer and the manufacturer. First, the consumer can decide which camera to buy, having a precise idea of the actual image quality reachable with each device. Second, the manufacturer has a clear insight of what camera features need to be improved in order to have a better ranking.

Several test procedures are indeed available in the form of an ISO norm. Among them, the ISO/TS 19567-2:2019 norm presents a test procedure that evaluates the capacity of a camera to properly restore micro-textures in all conditions, thanks to a dead leaves image target. Dead leaves images indeed exhibit statistical properties close to those of natural images, as explained in Section IV.2. The test consists of photographing a dead leaves image target as the one in Fig. VII.1, under controlled conditions. Then, one can compute a restoration score by comparing the power spectrum of the obtained image and a computed ground truth spectrum. This metric, called *acutance*, was first presented by Cao et al. [33] in 2010.

In previous chapters, we observed that training image-denoising networks on dead leaves images led to very good restoration performances, on synthetic and natural images,

Moreover, we showed in Section IV.3 that training an image denoiser on a mix of natural images and dead leaves images led to:

- denoising performances on par with training on natural images,
- much better denoising performances on a test set of synthetic images.

Inspired by these results, we show, in the present chapter, that it is possible to optimize the acutance of an image-denoising network without degrading the performances on natural

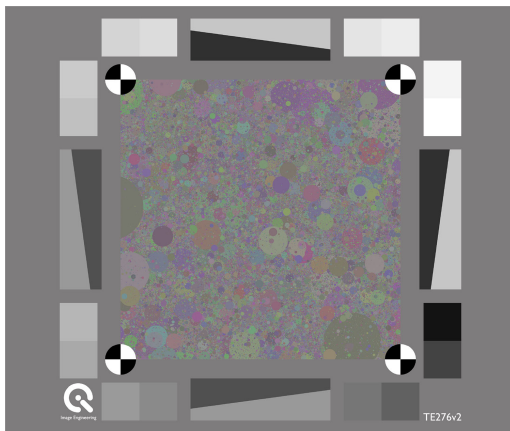


Figure VII.1: Unofficial dead leaves image target used for camera evaluation

images. To do so, we train a denoising CNN on a mixed database of synthetic and natural images to minimize a new loss function. In short, this loss combines a data fidelity loss (in our case the \mathcal{L}_2 distance) with a frequential loss, adapted from the acutance metric.

This chapter is organized as follows. In Section VII.2, we first introduce the acutance metric following the papers of Cao et al. [32, 33] and the paper of Artmann [14]. Then, we present how we adapted it to form a loss function, which can be used to train an image denoising network. In the present chapter, we will only focus on AWGN removal, by training the FFDNet network presented by Zhang et al [202]. In Section VII.3, we report our experiments. We first investigate the role of the penalization term as well as the impact of the acutance loss on the restored spectrum. Additionally, we show that our method performs on par with a natural-image trained FFDNet for natural image denoising, while improving both the acutance and the PSNR for dead leaves image denoising. We also show in visual examples that the network trained with synthetic images reduces the network hallucination phenomenon (i.e., the artificial creation of details by classical image-to-image neural networks).

This chapter not only shows that one can increase its performances on a standard test by training an imaging pipeline to improve its response to specific images. We believe this work is an additional argument to support the replacement of some steps of the camera ISP by learnable neural networks, which was already suggested in the following references [85, 36, 64].

VII.2 A frequential loss for dead leaves images

VII.2.1 Background

Dead leaves images were first used for camera evaluation in 2009 by Cao et al [32]. The paper aims at creating a standard test for cameras to measure texture sharpness. The idea is to measure the response of a camera to a specific image target. This target should have the following properties:

- **Rotation and shift invariance:** Texture can indeed appear in all directions, and over the whole image field.
- **Scale invariance:** the test must be led regardless of the definition of the sensor. Therefore the distance between the camera and the target should not affect the test.
- **Contrast invariance:** Texture images can indeed have many different contrast profiles. The target image should contain all possible contrasts
- **Natural properties:** The target image should look like a texture or at least exhibit some properties of natural images, such as occlusions, edges, similar statistical properties etc.

Since the dead leaves image model perfectly fits these criteria, it was chosen by the author for the target generation. In particular, the scale invariance property is only achievable when the disks radii follows a power law with $\alpha = 3$. The dead leaves target is therefore generated with this parameter. Note that to ensure the convergence of the algorithm, bounding parameters r_{min}, r_{max} are required, preventing full scale invariance. In the paper, the authors evaluate the response of a camera to the dead leaves target by computing the ratio of the power spectra, resulting in a Modulation Transfer Function (MTF). At each position (m, n) for an (N, N) image:

$$\text{MTF}_{2D}(m, n) = \frac{|\hat{Y}(m, n)|}{|\hat{X}(m, n)|},$$

where \hat{Y} is the spectrum of the obtained image, \hat{X} is the ground truth spectrum. In all that follows, we compute the image spectra on a greyscale version of the color image, obtained by the standard linear combination $Grey = 0.2126R + 0.7152G + 0.0722B$. Let us explain why this function is relevant to assess the spatial frequency response of a camera. Supposing that the obtained image corresponds to the convolution of a blur kernel B with the true signal, this translates in the Fourier domain as $\hat{Y} = \hat{B} \cdot \hat{X}$. Therefore the ratio of the power spectra corresponds to the Fourier coefficient of the blur kernel induced by the camera. Of course, this explanation excludes non linear transforms which are frequent in the acquisition of an image. It is nonetheless a good indicator of the cameras spatial frequency response. This quantity was already evaluated in another ISO norm (ISO 12233:2017), which measured the exact same ratio but for a different target image with slanted edges and oscillating details.

Since the dead leaves target is rotationally invariant, so is its spectrum. Therefore we can express the MTF as a 1D function. We can indeed obtain a 1D signal by averaging MTF_{2D} on concentric rings of width 1. The MTF becomes:

$$\text{MTF}_{1D}(k) = \frac{\mathbb{1}_{C_k}}{\#C_k} \odot \text{MTF}_{2D},$$

where $C_k = \{(i, j) \in [-N/2, N/2]^2 | (k-1)^2 \leq |i^2 + j^2| < k^2\}$ corresponds to a ring of radius k . In a follow-up paper [33], Cao et al. propose to compute the ground truth spectrum based on an explicit formula, rather than computing it from the original digital file. The spectrum of a dead leaves image indeed follows a power law distribution (empirically). Though this is true for most frequencies, the approximation fails for very low and very high

frequencies as we can see in Fig. VII.2. In practice, the authors claim that this does not matter, since very high spatial frequencies are almost imperceptible by a camera. With this approximation, the slope of the 1D power spectrum η corresponds to the exponent of the power law, and was estimated at $\eta = 1.93$ by Cao et al. They propose the following formula:

$$|\hat{X}(m, n)| = \frac{c(N)}{(m^2 + n^2)^{\eta/2}},$$

where $c(N)$ is a constant which depends on the size of the dead leaves target.

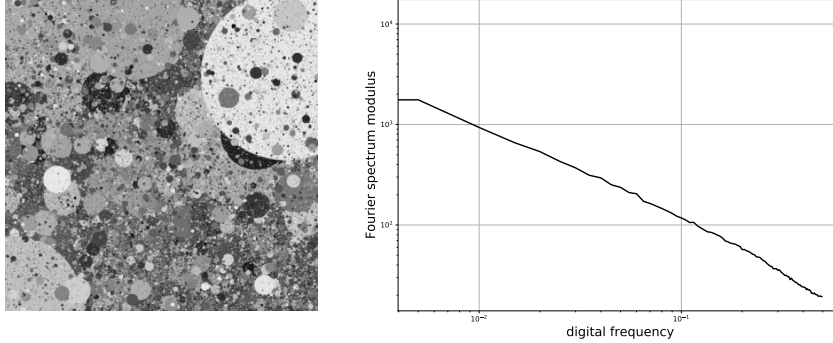


Figure VII.2: Grey level dead leaves image and its associated digital spectrum in logarithmic scales. The theoretical value is a straight line.

Though the full MTF_{1D} is a good indicator of the camera's capacity to render textures, it is more helpful to compute a single score. To that end, Cao et al. define the *texture acutance* as a weighted sum of the MTF_{1D} , with weights defined by a contrast sensitivity function (CSF), inspired by the slanted edge SFR. Our visual system is indeed more sensitive to some frequencies than others. In that regard, the CSF models the sensitivity of the visual system to spatial frequencies expressed in cycle/degree. Based on the physiological analysis of the contrast sensitivity of infants and monkeys led by Movshon and Kiorpes [137], Cao et al. used the following formula to model the CSF:

$$CSF(\nu) = a.\nu^c.e^{-b\nu},$$

where ν is a spatial frequency expressed in cycles/degree, the parameters $b = 0.2$, $c = 0.8$, and a is a normalizing parameter so that $\int_0^{\text{Nyquist}} CSF(\nu) d\nu = 1$. We show in Fig. VII.3, the profile of the CSF.

Given this formula, the texture acutance score can be written as:

$$A = \int_0^{\text{Nyquist}} CSF(\nu).MTF_{1D}(\nu) d\nu.$$

The perfect MTF corresponds to a constant function equal to 1, meaning that the frequential content has been perfectly restored by the camera for every frequency. This leads to an acutance $A = 1$. An acutance greater than 1 indicates that frequential content was added to the image probably because of noise or sharpening. An acutance lower than 1 indicates that some frequencies have been lost.

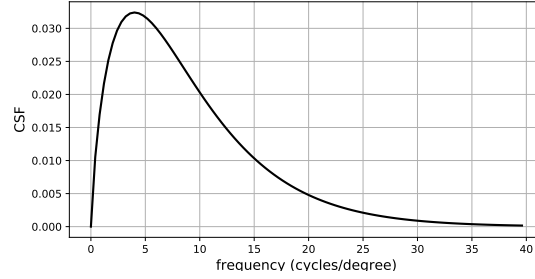


Figure VII.3: Profile of the CSF of the human visual system for spatial frequencies expressed in cycles/degree.

Unfortunately, an acutance equal to 1 does not necessarily mean that the frequencies are perfectly restored. Because of the acutance formula, an imaging device that adds high frequencies and removes low frequencies could also have an acutance equal to 1 without properly restoring the whole spectrum of the target. Moreover, the spectrum of the obtained image can be heavily degraded by noise, leading to an incorrect estimation of the MTF. In 2015, Artmann proposed to better assess the MTF by subtracting the spectrum of the noise [14]. He proposes to estimate the spectrum of the noise by taking a photograph of a uniform grey surface and computing its power spectrum $|\hat{N}|$. The MTF becomes

$$\text{MTF}_{\text{direct}}(m, n) = \sqrt{\frac{|\hat{Y}(m, n)|^2 - |\hat{N}(m, n)|^2}{|\hat{X}(m, n)|^2}}.$$

Even though this method allows to mitigate the effect of noise on the MTF estimation, it assumes an additive and signal independent noise model, which is far from reality. Moreover, according to Artmann, there are many non linear functions to reduce noise and process the image, which also alter the MTF estimation. In the same paper [14], Artmann proposes a new computation trying to correct these issues. Here, we consider the complex spectrum of a reference digital dead leaves target \hat{X} , rather than the estimate of the power spectrum $|\hat{X}|$ in the spatial domain. In the previous version, the phase information was lost. Only having access to the amplitude of the spectrum meant that we could not differentiate frequencies which were already in the target and information that was added by the imaging device. Therefore noise and non linear functions had an impact in previous computations.

The proposed method, which we call $\text{MTF}_{\text{cross}}$ uses the cross power density between the target and the obtained image $\phi_{XY}(m, n)$, and the auto power density $\phi_{XX}(m, n)$. More precisely,

$$\phi_{XY}(m, n) = \hat{Y}(m, n)\hat{X}^*(m, n) \quad \text{and} \quad \phi_{XX}(m, n) = \hat{X}(m, n)\hat{X}^*(m, n).$$

The difficulty here resides in making a digital reference target X that can be easily matchable with the photograph Y . Artmann proposes a precise protocol to match markers and to counter lense distortions, which we will not detail here. The photograph is also linearized, by inverting the camera tone mapping operations, which are assumed to be known for every camera. The ground truth image is linear by construction. Having perfectly matched linear

images, the MTF becomes:

$$\text{MTF}_{cross}(m, n) = \left| \frac{\phi_{XY}(m, n)}{\phi_{XX}(m, n)} \right|.$$

We can replace the initial MTF_{2D} with MTF_{cross} which is a more accurate estimate of the SFR, and compute a more reliable acutance score. The procedure proposed by Artmann has also been used in the standard ISO/TS 19567-2:2019 norm for texture sharpness estimation. We describe in the following section how we derived a texture loss function based on the presented acutance score.

VII.2.2 Acutance loss for image restoration CNNs

Previous experiments (see Section IV.3) showed that models trained on mixed databases perform on par with models trained on natural images only, while improving results on dead leaves image targets. We believe we can further improve the frequential response of models trained on mixed sets, by adapting the acutance score in a loss function.

In the context of AWGN removal for color RGB images, the noisy image corresponds to $Y = X + n$ where X is a ground truth dead leaves image of size $(N, N, 3)$. The denoising network f_θ produces an estimate of the clean image $Z = f_\theta(Y)$. For our restoration problem, we can consider that the denoising network is analogous to the camera which acquires the dead leaves target. We can compute the MTF_{cross} of the denoising network with the same formula $\text{MTF}_{cross}(m, n) = \left| \frac{\phi_{XZ}(m, n)}{\phi_{XX}(m, n)} \right|$, based on the computation of the digital spectrum of both X and Z . Before, computing the spectra, we first convert each color image in a grey level image and linearize it by applying an inverse gamma transform $\Gamma^{-1}(x) = x^{2.2}$ to be as close as possible with the camera evaluation protocol.

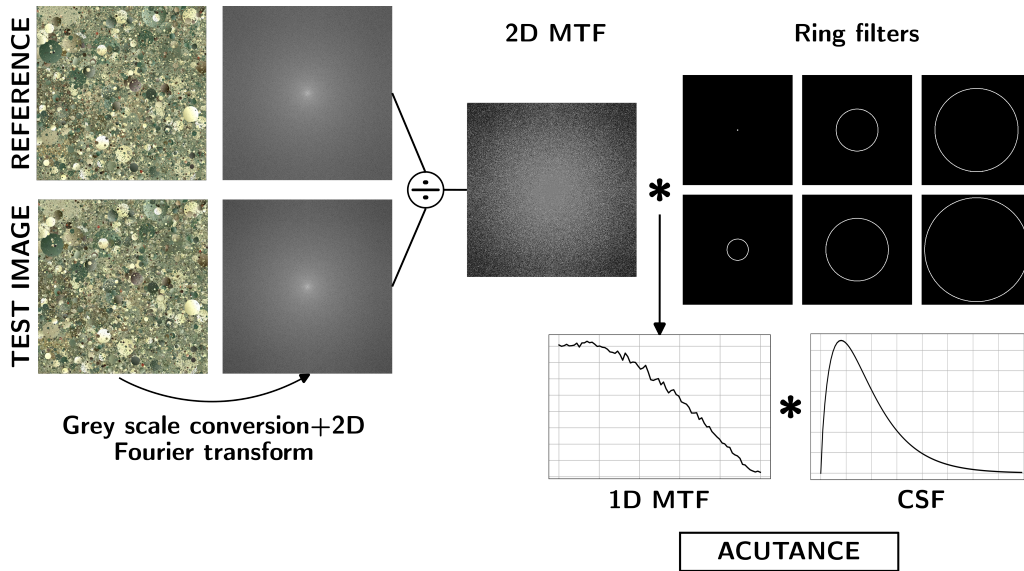


Figure VII.4: Diagram explaining the computation of the acutance metric

The obtained MTF_{cross} is turned into a 1D signal as described above. For faster computation, concentric ring masks are stored in GPU so that the MTF_{1D} can be accelerated

with parallel computing. The acutance score can then be derived with the exact same formula $A(Z, X) = \int_0^{\text{Nyquist}} \text{CSF}(\nu) \cdot \text{MTF}_{1D}(\nu) d\nu$. Since the best acutance is 1, we define our acutance loss function as:

$$\mathcal{L}_{acutance}(Y, X) = |1 - A(f_\theta(Y), X)|,$$

which penalizes both adding or removing frequential information.

Training an image denoising network only with this loss is impossible since it loses spatial information. We therefore keep our initial data fidelity term, which is the \mathcal{L}_2 loss. When training on dead leaves images our loss is a weighted combination of both losses $\mathcal{L} = \mathcal{L}_2 + \lambda \cdot \mathcal{L}_{acutance}$.

Since we train the image denoiser on both natural images and dead leaves images, we compute the acutance loss only on the dead leaves images in a minibatch D of size K and the \mathcal{L}_2 for all images. The formation of minibatches during training indeed randomly samples images from the mixed set. Thus, the loss in a batch becomes:

$$\mathcal{L}_{batch} = \frac{1}{K} \sum_{i=0}^K \|x_i - f_\theta(x_i + n_i)\|_2^2 + \frac{\lambda}{m^T \mathbf{1}} \sum_{i=0}^K m_i \cdot \mathcal{L}_{acutance}(f_\theta(x_i + n_i), x_i),$$

where m is a masking vector of size K such that $m_i = 1$ if x_i is a dead leaves image, or $m_i = 0$ otherwise. In order to count the number of dead leaves images we sum this masking vector which is given by $m^T \mathbf{1}$.

VII.3 Image denoising results with FFDNet

We chose to train the FFDNet network to illustrate the impact of the perceptual loss we presented in the previous section¹. However, we adapted the training scheme of the network to the present problem. First, we increased the size of the training patches from (50, 50, 3) to (100, 100, 3). The reason for this is that the estimation of the 1D-MTF on a small patch is not sufficiently accurate. Keeping the same rings' width would result in fewer estimates for the 1D-MTF. On the other hand, decreasing the rings' width would lead to noisier estimates. Therefore, we decided to train with larger patches. Second, we had to reduce the batch size from 64 to 32 during training. Since training patches are wider, the memory footprint is also larger, which forces us to create smaller mini-batches. For similar reasons, we reduced the number of training samples from 300000 to 150000, with 100000 natural image patches and 50000 synthesized dead leaves patches. The other training hyper-parameters remain unchanged, such as the number of epochs or the learning rate decaying schedule.

In this experimental section, we will first study the impact of the weighting parameter λ on the numerical results. We will then show how the acutance metric optimization affects the restored image spectrum. Finally, we will compare the visual results obtained with and without this loss.

VII.3.1 Finding the optimal perceptual loss parameter

Jointly training on natural images and synthetic images leads to performances on par with training with natural images. But is it still the case when we add the acutance loss? We can

¹We remind the reader that this network was presented in Section III.1.1.

also ask ourselves if increasing the weighting parameter λ increases the acutance evaluation without impairing performances on natural images.

To answer these questions, we trained the same FFDNet network with the acutance loss with different values of λ , following a logarithmic grid. More precisely we trained FFDNet with $\lambda \in [1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]$.

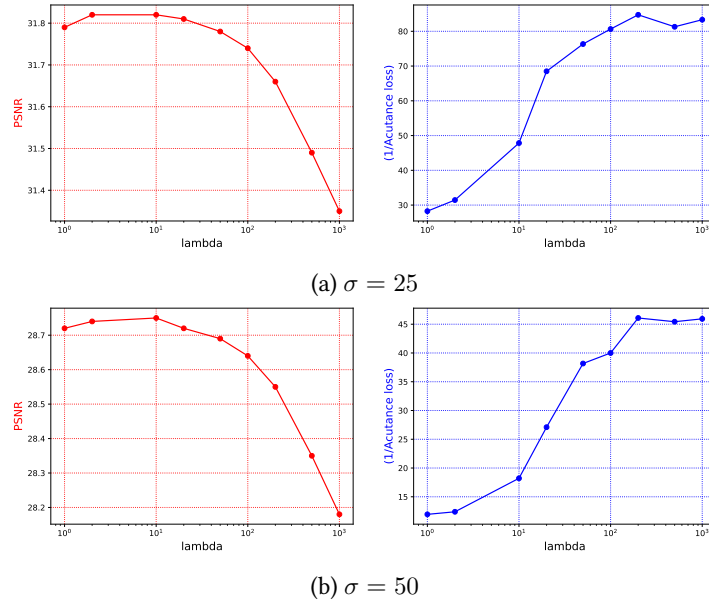


Figure VII.5: Graph of the PSNR and the inverse of the acutance loss depending on the penalizing parameter λ for different noise standard deviation σ (for both, the higher the curve the better).

We then tested our models numerically on two datasets. First, we evaluated the data fidelity by computing the PSNR on the Kodak24 dataset, a benchmark test set of 24 natural images. Second, we evaluated the acutance metric on a test set of synthesized dead leaves images which we already used in Chapter IV. We report, in Fig. VII.5, the graphs of the PSNR and the inverse of the acutance loss depending on the weighting parameter λ for different noise level values. In both cases, the higher a point is, the better the evaluation.

The profile of the graphs is the same for $\sigma = 25$ or $\sigma = 50$. We observe that the data fidelity is almost constant up to $\lambda = 50$, and rapidly decreases for higher values. The acutance loss behavior is almost the opposite: it increases rapidly up to $\lambda = 100$ and reaches a plateau afterward. This observation indicates that a good parameter for data fidelity and acutance lies between 50 and 100. These graphs also provide an answer to our initial question. They indeed show that we can optimize the acutance loss without impairing classic denoising evaluation, which is an interesting result for camera manufacturers.

VII.3.2 Spectral preservation

For mixed trainings of FFDNet, the acutance loss is greatly improved by adding the acutance loss. However, the acutance loss does not give us any idea of the MTF of the trained network.

In order to understand the impact of the acutance loss on the spectral preservation of the network, we compute its MTF as described next.

We compute the 1D-MTF from the denoised image and the original image for each dead leaves image of the synthetic test set. Since the 1D-MTF depends on the image’s content, which differs from image to image, we average the obtained MTF over the whole dataset. This results in a more accurate estimate of the MTF In Fig. VII.6, we report the MTF of

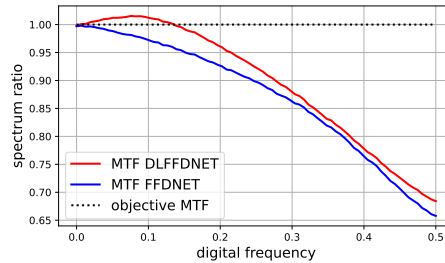


Figure VII.6: Comparison of the MTF evaluated with FFDNet trained on a mixed database with or without the acutance loss, on the whole dead leaves image test set.

FFDNet trained with and without the acutance loss (with $\lambda = 50$) for a noise level $\sigma = 25$. We can make the following observations:

- **low frequencies:** the MTF of the model trained with acutance loss is close to or above 1. Recall that a perfect MTF is equal to one for all frequencies. For these frequencies, this means that the network adds frequential information. This is one of the disadvantages of the acutance metric, for which the optimal value is one. However, a system can achieve an acutance of 1 by adding and removing frequencies.
- **low to medium frequencies:** in this segment, the gap between the two MTF is significant, confirming that the acutance loss allows having a better frequential response.
- **higher frequencies:** for these frequencies, the MTF for the model trained with the acutance is still above the other one, but the gap is smaller. Note that the CSF is very small for these frequencies, which can explain this behavior.

In general, the gap between the obtained MTF follows the profile of the CSF. It is large when the CSF is large and small when the CSF is small. Overall, the obtained MTF behaves as expected. However, the artificial addition of low frequencies is problematic. We believe that some alternatives to the acutance loss could be created to penalize the addition of frequential content more.

VII.3.3 Visual analysis

In Chapter IV, we saw that FFDNet trained on a mixed database could reach performances on par with FFDNet trained on natural images only. Numerically, the results were very similar. We also did not notice strong visual differences in the denoised images. However, we can note some subtle visual improvements when using the acutance loss during training.

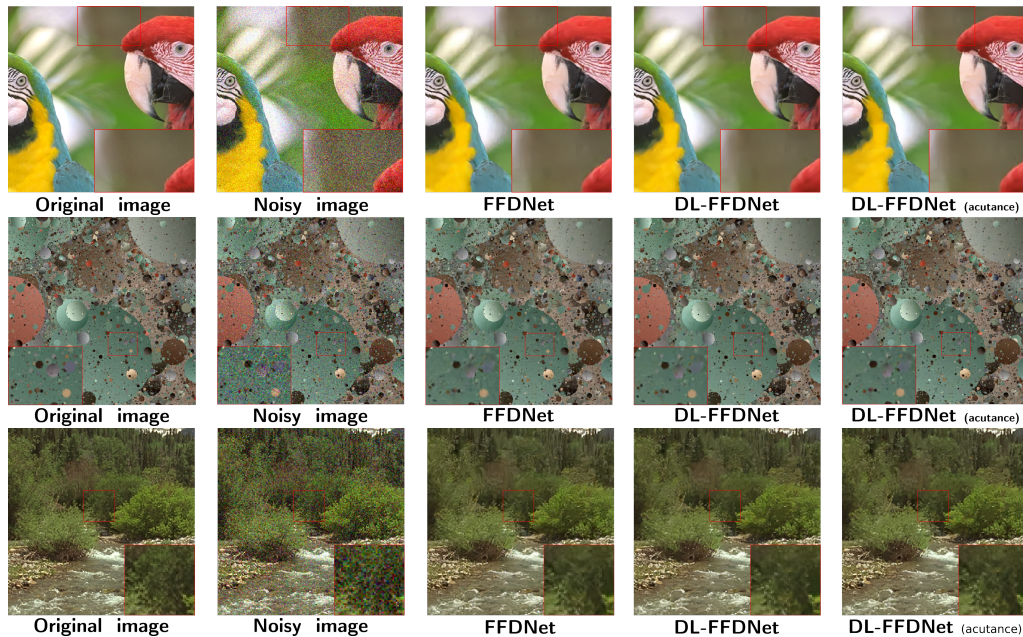


Figure VII.7: Comparison of FFDNet results on two natural images and on a dead leaves image. From left to right: original image, noisy image, image denoised with standard FFDNet, image denoised with FFDNet trained on a mixed database without the acutance loss, and finally with the acutance loss.

First, we note that the denoised images remove some artifacts created by the denoising network. As we can see in the first row of Fig. VII.7, there are some oscillation-like artifacts in the background of the image, which is homogeneous in the original image (please consult the electronic version of the document). The same artifact appears for the model trained on a mixed database. However, this artifact is almost gone for the model trained with the acutance loss. Since it is a frequencial artifact, we can assume that having a frequencial constraint tends to reduce it. We can do a similar observation for a light ringing artifact at the border of the red area, which is corrected with the model trained with the acutance loss. Observing the results of the second row, we clearly see that the classic FFDNet network adds frequencial information which was not there before. Even though we do not observe this in the global MTF, oscillations are created, which did not exist in the original image. If we compare the results of the models trained on a mixed set, we see that the results are very similar. However, we notice that the model trained with the acutance loss better retrieves tiny disks. When it comes to texture images, the performances are almost equivalent for all models, as we can see in the third row. Since texture images usually contain a lot of high frequencies, we can question the validity of the acutance weighting function for texture preservation. The CSF indeed puts much more weight on lower frequencies than on higher frequencies, as we can see in Fig. VII.3. Other weighting functions could possibly increase the preservation of textures by adding more weight on higher frequencies. Such a re-weighting can be incorporated easily to our framework.

VII.4 Conclusion

Texture preservation is a crucial image quality criterion for cameras. To evaluate it, Cao et al.[33] first proposed a test procedure based on the frequential response of an imaging device to a dead leaves image target. In the present chapter, we adapted the acutance metric presented by Artmann [14] in a perceptual loss function for a denoising problem. Following the results of Chapter IV, which showed that training on a mixed set of synthetic and natural images did not impair performances on classic benchmarks, we decided to train the same denoiser with the acutance loss. The results showed that we could increase the acutance loss without degrading data fidelity metrics like the PSNR. Moreover, we observed that the acutance metric substantially impacted the preservation of the spectrum of the restored images. Finally, we observed that in various cases, the acutance loss allowed to remove some artifacts and to improve the preservation of small details.

Despite these promising results, the use of the acutance metric could benefit from improvements. First, the acutance loss optimization does not prevent the addition of frequential content. We indeed observed that the global MTF exceeded one for some frequencies. Another loss than the acutance might be more relevant to prevent this behavior. Second, we noticed that the acutance loss did not improve the performances in texture areas. This is probably due to the CSF weighting, which is centered on lower frequencies. For all these reasons, we believe there is still room for improvement in this metric and that dead leaves images can greatly help in this respect.



Conclusions and perspectives

VIII.1 Conclusions

In order to train a neural network for image restoration, the standard practice is to collect large datasets of pairs of distorted and clean images of the same scene. While this may be easy for simple or artificial restoration tasks, it is much more complicated for real-world problems. In these cases, acquiring paired images is a cumbersome and time-consuming effort, which can also require complex post-processing operations to create ground-truth data. This is a severe limitation of deep learning-based image restoration methods since trained networks often fail to generalize to new modalities of acquisition and distortion. Our main goal was to address this problem by synthesizing images with the necessary properties to replace real images in a training set. We hoped that an easy generation algorithm could replace the tedious image acquisition procedure, thus saving time and effort while maintaining excellent restoration performances.

Our first challenge was to create an image-generation algorithm to achieve this goal. The algorithm had to be simple enough to be easily tuned to various restoration tasks and agnostic of the test-data distribution. However, it also had to model some properties of natural images with enough accuracy to ensure good restoration performances. A crucial question arose: which are these desired properties?

In chapter IV, we showed that training an image restoration neural network on images generated with an adapted version of the dead leaves model led to surprisingly good restoration performances. Dead leaves images, first presented by Matheron [132], indeed exhibit statistical properties close to those of natural images (e.g. similar distributions of the gradient and spectrum), as well as clear edges and occlusions, which are frequent in natural images. Moreover, it is easily tunable to different modalities: the geometry of a dead leaves image is indeed dictated by only three parameters $(r_{min}, r_{max}, \alpha)$. Therefore the dead leaves image model was a natural candidate for image generation.

In Section IV.3, we also analyzed the importance of different image properties. To do so, we generated different dead leaves image datasets by changing generation parameters and removing some steps of the algorithm. We also compared our model with other synthetic image models, such as Gaussian random fields. The most important properties were:

- **Natural colors:** sampling colors in natural images greatly improved performances, in comparison with training with dead leaves images whose colors were sampled uniformly at random.
- **Variety of content:** A model trained with fine-grained textures and homogeneous areas was the better tradeoff.
- **Modeling the acquisition:** we modeled camera acquisition by adding blur and sub-sampling to our generated images. Without these steps, the trained model performed poorly.
- **Non-Gaussianity:** the model trained on Gaussian random fields performed poorly on images with clear edges, unlike the model trained on dead leaves images, which are non-Gaussian.

In addition to this study, we showed that a model trained on synthetic and natural images performed on par with a model trained on natural images only. Therefore, training with additional synthetic data did not impair performances, confirming that the dead leaves model was a good model for natural images.

Among the previously listed properties, the most important was the color sampling choice. For this particular task, our initial algorithm sampled the colors of a randomly picked natural image for each synthetic image generation. This technique ensures that the colors in a dead leaves image are coherent and natural. However, it also requires having access to a dataset of natural images, making the whole algorithm data-dependent. This dependency is one of this algorithm’s major drawbacks since our goal was to rely on a data-agnostic model. In order to bypass this dependency, an alternative consists of sampling colors with a simulated color distribution that is similar to a natural image’s color distribution. This brings a new challenge: how can we generate such plausible color point clouds?

In chapter VI, we proposed a data-agnostic and universal color sampling technique based on the color line prior introduced by Omer et al. [146]. This work shows that, in many cases, the colors of a single natural image are distributed along elongated line clusters in the sRGB cube, each line corresponding to a single object. Our first contribution in this chapter was a statistical study of the first-order statistics of the colors in a large dataset of RAW images from different cameras. We showed that we could approximate the whole distribution of colors in a universal- RGB cube with a simple 3D Gaussian mixture model. However, our goal was to simulate color point clouds similar to those of natural images. The 3D model only allows to sample colors from the distribution estimated on the whole dataset. Therefore, the obtained color profiles of the synthesized images would be almost identical from one generation to the other. We then showed that we could split the color distribution model into a decoupled one with a chromaticity and a luminance component. The chromaticity was approximated with a 2D Gaussian mixture model, while the conditional luminance was approximated with a Gamma distribution. The decoupled model was indeed more adequate

for simulating color line clusters. With the sampling procedure described in Section VI.3, we were able to generate dead leaves images with realistic color profiles, and train an image restoration neural network with them. Its performances were close to those obtained by training on our previously generated dead leaves images. This result confirmed the validity of our coloring procedure. Moreover, it also made our image generation algorithm completely data-agnostic while preserving excellent restoration performances. With this result, we achieved the first objective of this PhD work, which was to implement a data-agnostic image generation method for deep learning-based image restoration methods and to study the necessary properties of synthetic models to reach good performances.

Our second objective was to make the proposed method tunable for real-world restoration problems. To achieve this objective, our first challenge was to synthesize real data. Since we focused on two restoration tasks that used RAW images, we had to simulate RAW-like images. Therefore, in chapter V, we proposed an adaptation of the dead leaves image algorithm, which produced artificial RAW images. The main modification concerned the color sampling algorithm and the post-processing of the dead leaves image. We could generate a large dataset of undistorted RAW images with this algorithm. However, training an image restoration neural network supposes to have access to distorted and undistorted image pairs. In order to create such a dataset, we had to simulate the distortion process. This was very challenging since we had to model the distortion process and estimate the model's parameters on real RAW data without having access to the corresponding cameras. We proposed a sound noise model for the two tasks we studied, i.e., Smartphone RAW image denoising and low-light RAW image enhancement. The model depended on a set of parameters that varied with the sensitivity (ISO) during the shot. Based on these distortion models, we proposed a simple method to estimate the corresponding noise parameters on the datasets we had access to. Other estimation methods existed, but they required having access to the corresponding camera. Nonetheless, we could accurately model these distortions with good estimates of the noise parameters. Having pairs of clean and distorted RAW dead leaves images, we were able to train a neural network for the two image restoration tasks at hand. The trained network produced competitive results with the network trained on real RAW images. For Smartphone RAW image denoising, our model performed on par with its classic alternative, producing even sharper images in some cases. For low-light image enhancement, the results were also very good when the images were not too dark. However, our model could not properly restore the images in extreme cases, such as very high ISO and indoor night photography. In these conditions, complex electronic processing are involved, making the distortion model even more complex. Therefore, modeling such extreme low-light distortions is an arduous and time-consuming task.

Overall, we successfully extended the use of dead leaves images for real-world applications. We were able to adapt the generation algorithm and model complex noise processes, leading to competitive restoration performances.

The present dissertation focused on exploiting the potential of dead leaves images as a simple image generation tool in the context of image restoration. Before our work, dead leaves images were already used for a practical application: camera evaluation. In order to assess the capacity of cameras to render textures properly, one can analyze the frequential

response of a camera. To do so, a standard test consists of taking a picture of a dead leaves image target and comparing the photograph's spectrum with a ground truth one. The dead leaves model was chosen for its multiple invariance properties and its similarity with natural images [32, 33, 14].

Inspired by this body of literature and by our results in chapter IV, we decided to transform the acutance metric used for camera evaluation into a perceptual loss. We observed that we could indeed have excellent image restoration results on natural images while improving the results on synthetic images by simply adding dead leaves images to the natural training set. Following this result, we showed, in chapter VII, that we could improve the acutance metric of a denoising network without impairing the data fidelity evaluation on natural images. Adding the acutance loss improved the frequency preservation of the dead leaves images. It also reduced the appearance of hallucination artifacts, which were frequent for the model trained on natural images only. The work presented in chapter VII better exploits the full potential of dead leaves images, bridging the gap between image restoration and camera evaluation. It is also an additional argument to support the idea that learned methods could efficiently replace some steps of the Image Signal Processing pipeline.

Overall, the experiments presented in this manuscript contradict the common assumption that large databases of real paired images are required to train an image restoration neural network for good performance. We circumvent the burden of data acquisition by synthesizing images, opening the way for lighter training schemes, and saving time and effort.

Even though we obtained good restoration performances, our model trained on dead leaves images struggled to restore some structures. In the following section, we will present the prospects of improvement for our methods and other perspectives opened by our works.

VIII.2 Perspectives

VIII.2.1 Improving the realism of the synthetic dataset

One of the key ingredients of the success of the proposed method was to match, as much as possible, the properties of the synthetic model with those of natural images. That is the underlying reason for most of our implementational choices:

- the geometry hyperparameters were chosen to create either homogeneous areas and micro-textures, which are frequent in natural images,
- the colors were sampled in a point cloud as similar as possible to the one of a real image,
- the post-processing operation was a simple approximation of a photograph acquisition, with blur and downsampling.

Even though we achieve good restoration performances with this simple model, improving the realism of the synthetic dataset could lead to performance improvements.

Geometrical modifications. We noticed that some structures were difficult to retrieve with our approach: straight borders, fine lines, and oscillating patches, for instance. The initial idea is to add more variety in the shape of the structuring element of the dead leaves image model. Disks only present curved edges, which might be the cause of the defects we observed. We tried adding rectangular shapes with random orientations to the original dead leaves model. However, this did not lead to any improvements in the restoration of straight borders. We could explain this result by arguing that the receptive field of our restoration network is quite small compared to the maximal radius size. This means that some disk borders are almost straight lines in the receptive field of the network. In that regard, another idea we believe is interesting is adding oscillating patches to the dataset. Simple cosine functions with different amplitude, period, and average color could generate these patches. Going further, large disks in the dead leaves images could present this kind of frequential content. The idea of attributing texture to the disks of the dead leaves images was already introduced in [127], inspired by our paper [6]. In their work, Madhusadana et al. sample the texture of each disk in a dataset of real photographic textures. While this allows having statistical properties closer to those of natural images, this also implies a data dependency which we wanted to avoid. Conversely, we propose to add oscillating functions as simple as a cosine, which does not depend on any real dataset. Another idea would be to add a Gaussian random field texture to large disks, making their aspect closer to the one of natural objects. Hopefully, these geometrical modifications of the dead leaves image model can improve the restoration performances, especially for the structures our model failed to restore. One of our priorities is to find the right balance between the model's simplicity and expressiveness. Therefore, new modifications should not include too many parameters or complex operations.

Realistic acquisition model. In Section V.2, we presented a simple algorithm to create artificial RAW data. The method was based on a Bayerisation of a simulated dead leaves image. However, we pointed out that both the blur kernel applied to the image and the downsampling step oversimplified the acquisition process. Photographic lenses all induce blur to a certain extent, but the blur kernel is much more complex than a Gaussian one. Recent works in image deblurring rely on carefully modeled blur kernels, which depend on the physical properties of the lens used. We believe that our method could benefit from these models. Moreover, the downsampling step was initially introduced to simulate the acquisition of a distant and printed dead leaves target, thus preventing aliasing and artificially sharp edges. However, this approximation assumes that every object in the dead leaves image is on the same plane. In real-world conditions, this absence of depth only corresponds to very specific acquisition conditions (for instance, having long focal length lenses, a small aperture, and distant objects), which are not frequent. It would be really interesting to include and exploit depth information. The first possibility would be to store each disk's arrival time. Having this information, we could try to simulate the depth of field by applying different blur levels depending on the arrival time. An even more direct and potentially more realistic approach would be to simulate a 3D dead leaves image made of spheres of random radiuses and color in a 3D rendering software such as Blender. This approach was already used by Madhusadana

et al. [128], but for a completely different objective than realism. They propose estimating disparity maps for stereo images by training a neural network on a 3D dead leaves image set. Given the exact depth information, generating two different views of a single scene and computing the disparity map between them is easy. An interesting perspective would be to create a 3D dead leaves image scene and to use a realistic simulation of a camera in this software to generate images with limited depth of field and realistic camera blur. An accurate model of the camera may improve the realism of the scenes and subsequently improve image restoration performances.

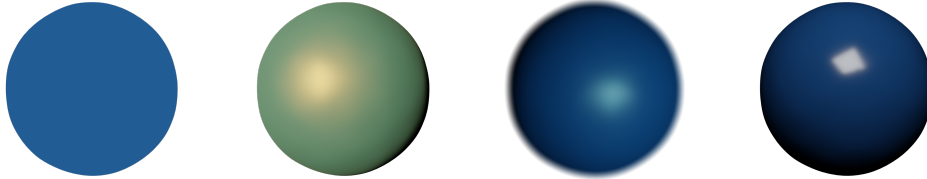


Figure VIII.1: Different views of a single blue sphere under various acquisition conditions generated with Blender.

Using realistic 3D rendering could potentially lead to other improvements. As pointed out in chapter VI, the perceived color of a Lambertian object is the product of the object's albedo with the illuminant. However, in most cases, objects are far from being Lambertian. The perceived color is a complex combination of the object's surface roughness, albedo, metallic, illuminant, etc. While these properties are hard to control on a 2D RGB image, it is much easier to control them in a 3D rendering program. For instance, we could generate leaves as spheres with varying roughness. Moreover, we could also render our image with various illuminants. We could change the temperature of the illuminant, its orientation, and its directionality. We show in Fig. VIII.1 different views of a single sphere under different conditions. We can obtain a variety of appearances by changing a few parameters. By better modeling the physics of the scene, and the interaction between objects and light, we could have a more realistic color distribution in the simulated image, which plays a major role in the success of our method. All these modifications could improve our simulated images' realism and, hopefully, translate into better restoration performances. Nevertheless, this approach would potentially make our data generation much more complex. Again, the algorithm should be balanced between complexity and realism to make it as adaptable as possible.

VIII.2.2 Extending the statistical analysis of the color distribution

The statistical study of the color distribution presented in chapter VI, as well as the color line sampling, led to good restoration performances for restoration neural networks. We could model color point clouds of different objects to model the interaction between objects of different chromaticity. Nevertheless, there is still room for improvement.

Joint chromaticity distribution. In our experiments, we independently sampled the chromaticity of two objects. However, this independence is indeed an approximation. Therefore,

studying the joint distribution of pairs of chromaticities of different objects would be interesting. With this model, we could sample the chromaticity of an object given the chromaticity of another one. It is unclear if this would improve our results, nor that independence is not a valid property. Nonetheless, this study would bring new insights about the distributions of colors in images, which is still an interesting contribution.

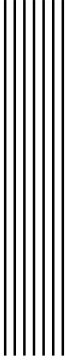
Extend the color model to surfaces. As explained by Lisani et al. [121], the color line prior is often not relevant to summarize the distribution of colors in the sRGB cube. A 2D surface model is often more accurate to model the color clusters from single objects. This is caused by two major components in the image: blur and textures. One could argue that blur can be obtained as post-processing and that we can create textures on large disks, in which case the color line model could be sufficient to indicate the chromaticity and luminance of disks. However, these steps also make our generation algorithm more complex. Therefore, having a color model that could generate realistic surface-like point clouds could lead to more realistic color profiles in the generated images without increasing the complexity. One of the main challenges is understanding the statistical properties of the color surface in an RGB cube. An initial step would consist in using the algorithm proposed by Lisani et al. to first summarize the color point cloud of a single image in different surface clusters. Then, we could estimate the properties of the found color clusters: width, length, curvature, average chromaticity, etc. By doing so over a large set of images, we could fit a reasonable probabilistic model on those properties to sample surfaces rather than lines.

Better model the mapping of RAW-RGB spaces from camera to camera. The performance loss in chapter VI was mainly due to the discrepancies between the dataset on which we estimated the color distribution (RAISE) and the dataset of test images (SIDD). The RAISE dataset was indeed acquired with cameras that did not exist in the SIDD dataset. Since the sensor's color filter array differs from one camera to the other, the color distributions also differ. Therefore, each camera defines its own RAW-RGB color space. In order to counter that, we mapped the colors of the RAISE dataset in an approximated universal RAW-RGB color space by multiplying each color by a diagonal matrix. Its diagonal coefficients were the daylight white-balance parameters, which correspond to fixed parameters estimated by the manufacturer for the D65 illuminant. To map the color back to camera-specific RAW-RGB spaces, we inverted this transform by multiplying the colors with the inverse diagonal matrix, containing the daylight white-balance gains for a specific camera. Since we did not have access to the cameras or these parameters, we approximated them by considering random values around plausible ones. A first improvement would be to obtain these parameters and only apply those rather than sampling random values. Going further, we could find a linear mapping from one color camera to the other in the form of a full $(3, 3)$ matrix. This technique was already studied by Nguyen et al. [nguyen2014raw]. This paper proposes to estimate the parameters of a mapping between two cameras, having access to identical photographs of a color chart for both cameras. The author found that the best mapping was an affine transform fitted on these images. The main difference with our set-up is that we consider that we do not have access to the cameras but unmatched images from different cameras. Therefore, this problem is much more challenging.

VIII.2.3 Investigating alternatives for the acutance loss

The acutance loss presented in chapter VII computes a weighted sum of an imaging device's modularity transfer function (MTF). The weights account for the contrast sensitivity of the human visual system and sum to one. Since the perfect MTF is one for each frequency, the corresponding acutance value is also one. However, the same acutance score could be reached with an infinite number of MTF profiles. For instance, an MTF where the response for high frequencies is much smaller than one and the response for low frequencies is much higher than one can also lead to an acutance of one. We indeed observe that training a denoising network with our acutance loss tends to add frequential information in lower frequencies, which is unwanted behavior. In order to prevent this phenomenon, the first possibility consists of adding a constraint to the restored spectrum. We could penalize the addition of frequential information with an additional loss term. Hopefully, this minor modification could efficiently remove these frequential artifacts.

While the acutance metric was initially developed to assess the cameras' capacity to render textures correctly, we noticed in our experiments that the addition of the acutance loss brought little to no improvements in texture preservation. The main cause is the profile of the contrast sensitivity function (CSF). This weighting function gives much more importance to low and intermediate frequencies than high ones. This is counter-intuitive since microtextures are mostly made of high-frequency content. The first possible explanation is that our transform from spatial frequency to digital frequency was inaccurate. We indeed assumed that the size of a pixel on a screen was $P = 0.2mm$ and that the viewing distance was $D = 1m$. While the pixel size is always the same, the viewing distance might be smaller and closer to $D = 0.5m$. In this case, the spatial to digital frequency transform is different, and the CSF gives more weight to high frequencies than before. Another possibility is to change the CSF with a constant function so that each frequency will have the same impact on the acutance score. The MTF is indeed always poor for high frequencies and good for low frequencies. It is understandable since higher frequencies are more challenging to retrieve from a noisy image than lower frequencies, which correspond to almost homogeneous areas.



Bibliography

- [1] Abdelrahman Abdelhamed, Marcus A Brubaker, and Michael S Brown. “Noise flow: Noise modeling with conditional normalizing flows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3165–3173.
- [2] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. “A high-quality denoising dataset for smartphone cameras”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1692–1700.
- [3] Raphaël Achddou, J Matias Di Martino, and Guillermo Sapiro. “Nested Learning for Multi-Level Classification”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2815–2819.
- [4] Raphaël Achddou, Yann Gousseau, and Saïd Ladjal. “Fully synthetic training for image restoration tasks”. In: *Computer Vision and Image Understanding (2023)*, p. 103723.
- [5] Raphaël Achddou, Yann Gousseau, and Saïd Ladjal. “Hybrid Training of Denoising Networks to Improve the Texture Acutance of Digital Cameras”. In: *International Conference on Scale Space and Variational Methods in Computer Vision*. Springer. 2023, pp. 314–325.
- [6] Raphaël Achddou, Yann Gousseau, and Saïd Ladjal. “Synthetic images as a regularity prior for image restoration neural networks”. In: *International Conference on Scale Space and Variational Methods in Computer Vision*. Springer. 2021, pp. 333–345.
- [7] Mahmoud Afifi and Michael S Brown. “Deep white-balance editing”. In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2020, pp. 1397–1406.
- [8] Eirikur Agustsson and Radu Timofte. “NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. July 2017.
- [9] Byeongyong Ahn and Nam Ik Cho. “Block-matching convolutional neural network for image denoising”. In: *arXiv preprint arXiv:1704.00524 (2017)*.

-
- [10] Luis Alvarez, Yann Gousseau, and Jean-Michel Morel. “The size of objects in natural and artificial images”. In: *Advances in Imaging and Electron Physics*. Vol. 111. Elsevier, 1999, pp. 167–242.
- [11] Matthew Anderson, Ricardo Motta, Srinivasan Chandrasekar, and Michael Stokes. “Proposal for a standard default color space for the internet—srgb”. In: *Color and imaging conference*. Vol. 1996. 1. Society for Imaging Science and Technology. 1996, pp. 238–245.
- [12] Francis J Anscombe. “The transformation of Poisson, binomial and negative-binomial data”. In: *Biometrika* 35.3/4 (1948), pp. 246–254.
- [13] Saeed Anwar and Nick Barnes. “Real image denoising with feature attention”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3155–3164.
- [14] Uwe Artmann. “Image quality assessment using the dead leaves target: experience with the latest approach and further investigations”. In: *Digital Photography XI*. Vol. 9404. SPIE. 2015, pp. 130–144.
- [15] Gilles Aubert, Pierre Kornprobst, and Giles Aubert. *Mathematical problems in image processing: partial differential equations and the calculus of variations*. Vol. 147. Springer, 2006.
- [16] Woong Bae, Jaejun Yoo, and Jong Chul Ye. “Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 145–153.
- [17] Manel Baradad, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. “Learning to see by looking at noise”. In: *Advances in Neural Information Processing Systems* 34 (2021).
- [18] Bryce E Bayer. “Color imaging array”. In: *United States Patent 3,971,065* (1976).
- [19] Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, and Patrick Pérez. “This dataset does not exist: training models from generated images”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 1–5.
- [20] Charles Bordenave, Yann Gousseau, and François Roueff. “The dead leaves model: a general tessellation modeling occlusion”. In: *Advances in applied probability* 38.1 (2006), pp. 31–46.
- [21] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.
- [22] Willard S Boyle and George E Smith. “Charge coupled semiconductor devices”. In: *Bell System Technical Journal* 49.4 (1970), pp. 587–593.
- [23] Andrew Brock, Jeff Donahue, and Karen Simonyan. “Large scale GAN training for high fidelity natural image synthesis”. In: *arXiv preprint arXiv:1809.11096* (2018).

-
- [24] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. “Unprocessing images for learned raw denoising”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11036–11045.
- [25] Antoni Buades, Bartomeu Coll, and J-M Morel. “A non-local algorithm for image denoising”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 2. IEEE. 2005, pp. 60–65.
- [26] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. “A review of image denoising algorithms, with a new one”. In: *Multiscale modeling & simulation* 4.2 (2005), pp. 490–530.
- [27] Antoni Buades, Jose-Luis Lisani, and Jean-Michel Morel. “Dimensionality of color space in natural images”. In: *JOSA A* 28.2 (2011), pp. 203–209.
- [28] Gershon Buchsbaum. “A spatial processor model for object colour perception”. In: *Journal of the Franklin institute* 310.1 (1980), pp. 1–26.
- [29] Harold C Burger, Christian J Schuler, and Stefan Harmeling. “Image denoising: Can plain neural networks compete with BM3D?” In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 2392–2399.
- [30] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. “A naturalistic open source movie for optical flow evaluation”. In: *European conference on computer vision*. Springer. 2012, pp. 611–625.
- [31] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. “Learning Photographic Global Tonal Adjustment with a Database of Input / Output Image Pairs”. In: *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*. 2011.
- [32] Frédéric Cao, Frederic Guichard, and Hervé Hornung. “Measuring texture sharpness of a digital camera”. In: *Digital Photography V*. Vol. 7250. International Society for Optics and Photonics. 2009, 72500H.
- [33] Frédéric Cao, Frédéric Guichard, and Hervé Hornung. “Dead leaves model for measuring texture quality on a digital camera”. In: *Digital Photography VI*. Vol. 7537. International Society for Optics and Photonics. 2010, 75370E.
- [34] Antonin Chambolle. “An algorithm for total variation minimization and applications”. In: *Journal of Mathematical imaging and vision* 20.1 (2004), pp. 89–97.
- [35] Stanley H Chan, Xiran Wang, and Omar A Elgendy. “Plug-and-play ADMM for image restoration: Fixed-point convergence and applications”. In: *IEEE Transactions on Computational Imaging* 3.1 (2016), pp. 84–98.
- [36] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. “Learning to see in the dark”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3291–3300.

- [37] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. “Pre-trained image processing transformer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12299–12310.
- [38] Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. “Image blind denoising with generative adversarial network based noise modeling”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3155–3164.
- [39] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.
- [40] Yunjin Chen and Thomas Pock. “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1256–1272.
- [41] Cristovao Cruz, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. “Nonlocality-reinforced convolutional neural networks for image denoising”. In: *IEEE Signal Processing Letters* 25.8 (2018), pp. 1216–1220.
- [42] Kostadin Dabov, Alessandro Foi, and Karen Egiazarian. “Video denoising by sparse 3D transform-domain collaborative filtering”. In: *European Signal Processing Conference* 16.8 (2007), pp. 145–149. ISSN: 22195491.
- [43] Scott J Daly. “Visible differences predictor: an algorithm for the assessment of image fidelity”. In: *Human Vision, Visual Processing, and Digital Display III*. Vol. 1666. SPIE. 1992, pp. 2–15.
- [44] Yabo Dan, Yong Zhao, Xiang Li, Shaobo Li, Ming Hu, and Jianjun Hu. “Generative adversarial networks (GAN) based efficient sampling of chemical composition space for inverse design of inorganic materials”. In: *npj Computational Materials* 6.1 (2020), pp. 1–7.
- [45] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. “RAISE: A raw images dataset for digital image forensics”. In: *Proceedings of the 6th ACM multimedia systems conference*. 2015, pp. 219–224.
- [46] Aram Danielyan, Vladimir Katkovnik, and Karen Egiazarian. “Image deblurring by augmented Lagrangian with BM3D frame prior”. In: *Workshop on Information Theoretic Methods in Science and Engineering*. Vol. 1. 2010.
- [47] Axel Davy, Thibaud Ehret, Jean-Michel Morel, Pablo Arias, and Gabriele Facciolo. “Non-local video denoising by CNN”. In: *arXiv preprint arXiv:1811.12758* (2018).
- [48] Charles-Alban Deledalle, Loïc Denis, Florence Tupin, Andreas Reigber, and Marc Jäger. “NL-SAR: A unified nonlocal framework for resolution-preserving (Pol)(In) SAR denoising”. In: *IEEE Transactions on Geoscience and Remote Sensing* 53.4 (2014), pp. 2021–2038.
- [49] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

- [50] David Donoho and IAIN M. JOHNSTONE. “Ideal Spatial Adaptation by Wavelet Shrinkage”. In: 81.3 (1994), pp. 425–455.
- [51] David L Donoho and Iain M Johnstone. “Minimax estimation via wavelet shrinkage”. In: *The annals of Statistics* 26.3 (1998), pp. 879–921.
- [52] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).
- [53] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. “Flownet: Learning optical flow with convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.
- [54] Thibaud Ehret, Axel Davy, Pablo Arias, and Gabriele Facciolo. “Joint Demosaicking and Denoising by Fine-Tuning of Bursts of Raw Images”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.
- [55] Thibaud Ehret, Axel Davy, Pablo Arias, and Gabriele Facciolo. “Joint demosaicking and denoising by fine-tuning of bursts of raw images”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8868–8877.
- [56] Raanan Fattal. “Dehazing using color-lines”. In: *ACM transactions on graphics (TOG)* 34.1 (2014), pp. 1–14.
- [57] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. “Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data”. In: *IEEE Transactions on Image Processing* 17.10 (2008), pp. 1737–1754.
- [58] Eric R Fossum and Donald B Hondongwa. “A review of the pinned photodiode for CCD and CMOS image sensors”. In: *IEEE Journal of the electron devices society* (2014).
- [59] David H Foster. “Color constancy”. In: *Vision research* 51.7 (2011), pp. 674–700.
- [60] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. “Micro-texture synthesis by phase randomization”. In: *Image Processing On Line* 1 (2011), pp. 213–237.
- [61] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “Image style transfer using convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2414–2423.
- [62] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “Texture synthesis using convolutional neural networks”. In: *arXiv preprint arXiv:1505.07376* (2015).
- [63] Pascal Getreuer. “Rudin-Osher-Fatemi total variation denoising using split Bregman”. In: *Image Processing On Line* 2 (2012), pp. 74–95.
- [64] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. “Deep joint demosaicking and denoising”. In: *ACM Transactions on Graphics (ToG)* 35.6 (2016), pp. 1–12.
- [65] Bernd Girod. “What’s wrong with mean-squared error?” In: *Digital images and human vision* (1993), pp. 207–220.

- [66] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.
- [67] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.
- [68] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feed-forward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.
- [69] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Russ R Salakhutdinov. “Neighbourhood components analysis”. In: *Advances in neural information processing systems*. 2005, pp. 513–520.
- [70] Tom Goldstein and Stanley Osher. “The split Bregman method for L1-regularized problems”. In: *SIAM journal on imaging sciences* 2.2 (2009), pp. 323–343.
- [71] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [72] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.
- [73] Yann Gousseau and François Roueff. “Modeling occlusion and scaling in natural images”. In: *Multiscale Modeling & Simulation* 6.1 (2007), pp. 105–134.
- [74] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. “Toward convolutional blind denoising of real photographs”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 1712–1722.
- [75] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.
- [76] Kaiming He, Jian Sun, and Xiaoou Tang. “Single image haze removal using dark channel prior”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.12 (2010), pp. 2341–2353.
- [77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.
- [79] Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pająk, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. “Flexisp: A flexible camera image processing framework”. In: *ACM Transactions on Graphics (ToG)* 33.6 (2014), pp. 1–13.

- [80] Mattias P Heinrich, Maik Stille, and Thorsten M Buzug. “Residual U-net convolutional neural network architecture for low-dose CT denoising”. In: *Current Directions in Biomedical Engineering* 4.1 (2018), pp. 297–300.
- [81] Keigo Hirakawa and Thomas W Parks. “Adaptive homogeneity-directed demosaicing algorithm”. In: *Ieee transactions on image processing* 14.3 (2005), pp. 360–369.
- [82] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.
- [83] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.
- [84] Thomas S. Huang, George J. Yang, and Gregory Y. Tang. “A Fast Two-Dimensional Median Filtering Algorithm”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 27.1 (1979), pp. 13–18. issn: 00963518. doi: 10.1109/TASSP.1979.1163188.
- [85] Andrey Ignatov, Luc Van Gool, and Radu Timofte. “Replacing mobile camera isp with a single deep learning model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 536–537.
- [86] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.
- [87] *Photography – Digital cameras – Part 2: Texture analysis using stochastic pattern*. Standard. Geneva, CH: International Organization for Standardization, 2019.
- [88] Ali Jahanian, Xavier Puig, Yonglong Tian, and Phillip Isola. “Generative models as a data source for multiview representation learning”. In: *arXiv preprint arXiv:2106.05258* (2021).
- [89] Viren Jain and Sebastian Seung. “Natural image denoising with convolutional networks”. In: *Advances in neural information processing systems* 21 (2008).
- [90] Geonwoon Jang, Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. “C2n: Practical generative noise modeling for real-world denoising”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2350–2359.
- [91] Younghyun Jo, Sejong Yang, and Seon Joo Kim. “Investigating loss functions for extreme super-resolution”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 424–425.
- [92] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *European conference on computer vision*. Springer. 2016, pp. 694–711.
- [93] Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. “Pre-training without natural images”. In: *Proceedings of the Asian Conference on Computer Vision*. 2020.

- [94] Marian Kazubek. “Wavelet domain image denoising by thresholding and Wiener filtering”. In: *IEEE Signal Processing Letters* 10.11 (2003), pp. 324–326. ISSN: 10709908. doi: 10.1109/LSP.2003.818225.
- [95] Wilfrid S Kendall and Elke Thönnies. “Perfect simulation in stochastic geometry”. In: *Pattern Recognition* 32.9 (1999), pp. 1569–1586.
- [96] Jack Kiefer and Jacob Wolfowitz. “Stochastic estimation of the maximum of a regression function”. In: *The Annals of Mathematical Statistics* (1952), pp. 462–466.
- [97] Dong-Wook Kim, Jae Ryun Chung, and Seung-Won Jung. “Grdn: Grouped residual dense network for real image denoising and gan-based real-world noise modeling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.
- [98] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. “Accurate image super-resolution using very deep convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1646–1654.
- [99] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [100] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).
- [101] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. “Normalizing flows: An introduction and review of current methods”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 3964–3979.
- [102] Shayan Kousha, Ali Maleky, Michael S Brown, and Marcus A Brubaker. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17463–17471.
- [103] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. “Noise2void-learning denoising from single noisy images”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2129–2137.
- [104] Samuli Laine, Tero Karras, Jaakko Lehtinen, and Timo Aila. “High-quality self-supervised deep image denoising”. In: *Advances in Neural Information Processing Systems* 32 (2019).
- [105] Edwin H Land. “The retinex theory of color vision”. In: *Scientific american* 237.6 (1977), pp. 108–129.
- [106] Valero Laparra, Johannes Ballé, Alexander Berardino, and Eero P Simoncelli. “Perceptual image quality assessment using a normalized Laplacian pyramid”. In: *Electronic Imaging* 2016.16 (2016), pp. 1–6.
- [107] Rémi Laumont, Valentin De Bortoli, Andrés Almansa, Julie Delon, Alain Durmus, and Marcelo Pereyra. “On Maximum-a-Posteriori estimation with Plug & Play priors and stochastic gradient descent”. In: *arXiv preprint arXiv:2201.06133* (2022).
- [108] Marc Lebrun. “An Analysis and Implementation of the BM3D Image Denoising Method”. In: *Image Processing On Line* 2 (2012), pp. 175–213. doi: 10.5201/ipo1.2012.1-bm3d.

- [109] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. “A nonlocal Bayesian image denoising algorithm”. In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1665–1688.
- [110] Bruno Lecouat, Jean Ponce, and Julien Mairal. “Revisiting Non Local Sparse Models for Image Restoration”. In: *arXiv preprint arXiv:1912.02456* (2019).
- [111] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.
- [112] Ann B Lee, David Mumford, and Jingtang Huang. “Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model”. In: *International Journal of Computer Vision* 41.1-2 (2001), pp. 35–59.
- [113] Kanggeun Lee and Won-Ki Jeong. “Noise2kernel: Adaptive self-supervised blind denoising using a dilated convolutional kernel architecture”. In: *Sensors* 22.11 (2022), p. 4255.
- [114] Stamatis Lefkimmiatis. “Non-local color image denoising with convolutional neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3587–3596.
- [115] Stamatis Lefkimmiatis. “Universal denoising networks: a novel CNN architecture for image denoising”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3204–3213.
- [116] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. “Noise2noise: Learning image restoration without clean data”. In: *arXiv preprint arXiv:1803.04189* (2018).
- [117] Xin Li, Bahadır Gunturk, and Lei Zhang. “Image demosaicing: A systematic survey”. In: *Visual Communications and Image Processing 2008*. Vol. 6822. SPIE. 2008, pp. 489–503.
- [118] Zun Li and Jin Wu. “Learning deep CNN denoiser priors for depth image inpainting”. In: *Applied Sciences* 9.6 (2019), p. 1103.
- [119] Jingyun Liang, Jiezhong Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. “Swinir: Image restoration using swin transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1833–1844.
- [120] Pierre-Louis Lions and Bertrand Mercier. “Splitting algorithms for the sum of two nonlinear operators”. In: *SIAM Journal on Numerical Analysis* 16.6 (1979), pp. 964–979.
- [121] Jose-Luis Lisani, Antoni Buades, and Jean-Michel Morel. “Image color cube dimensional filtering and visualization”. In: *Image Processing on Line* 1 (2011), pp. 57–69.
- [122] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. “Non-local recurrent network for image restoration”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 1673–1682.

- [123] Yang Liu, Zhenyue Qin, Saeed Anwar, Pan Ji, Dongwoo Kim, Sabrina Caldwell, and Tom Gedeon. “Invertible denoising network: A light solution for real noise removal”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 13365–13374.
- [124] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.
- [125] Aamir Hamid Lone and Arsheen Neda Siddiqui. “Noise models in digital image processing”. In: *Global Sci-Tech* 10.2 (2018), p. 63. ISSN: 0975-9638. DOI: 10 . 5958 / 2455-7110 . 2018 . 00010 . 1.
- [126] Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang. “Waterloo exploration database: New challenges for image quality assessment models”. In: *IEEE Transactions on Image Processing* 26.2 (2016), pp. 1004–1016.
- [127] Pavan C Madhusudana, Neil Birkbeck, Yilin Wang, Balu Adsumilli, and Alan C Bovik. “Image Quality Assessment Using Synthetic Images”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 93–102.
- [128] Pavan C Madhusudana, Seok-Jun Lee, and Hamid Rahim Sheikh. “Revisiting Dead Leaves Model: Training with Synthetic Data”. In: *IEEE Signal Processing Letters* (2021).
- [129] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. “Non-local sparse models for image restoration”. In: *2009 IEEE 12th international conference on computer vision*. IEEE. 2009, pp. 2272–2279.
- [130] Henrique S Malvar, Li-wei He, and Ross Cutler. “High-quality linear interpolation for demosaicing of Bayer-patterned color images”. In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. IEEE. 2004, pp. iii–485.
- [131] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections”. In: *Advances in neural information processing systems* 29 (2016).
- [132] George Matheron. “Random Sets and Integral Geometry”. In: (1975).
- [133] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. “What makes good synthetic training data for learning disparity and optical flow estimation?” In: *International Journal of Computer Vision* 126.9 (2018), pp. 942–960.
- [134] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4040–4048.
- [135] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. “Finding an unsupervised image segmenter in each of your deep generative models”. In: *arXiv preprint arXiv:2105.08127* (2021).

- [136] Antoine Monod, Julie Delon, Matias Tassano, and Andrés Almansa. “Video Restoration with a Deep Plug-and-Play Prior”. In: *arXiv preprint arXiv:2209.02854* (2022).
- [137] J Anthony Movshon and Lynne Kiorpes. “Analysis of the development of spatial contrast sensitivity in monkey and human infants”. In: *JOSA A* 5.12 (1988), pp. 2166–2172.
- [138] David Mumford and Basilis Gidas. “Stochastic models for generic images”. In: *Quarterly of applied mathematics* 59.1 (2001), pp. 85–111.
- [139] Kodai Nakashima, Hirokatsu Kataoka, Asato Matsumoto, Kenji Iwata, Nakamasa Inoue, and Yutaka Satoh. “Can vision transformers learn without natural images?” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2. 2022, pp. 1990–1998.
- [140] Patrenahalli M. Narendra. “A Separable Median Filter for Image Noise Smoothing”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-3.1 (1981), pp. 20–29. ISSN: 01628828. DOI: 10.1109/TPAMI.1981.4767047.
- [141] Michael K Ng, Liqun Qi, Yu-fei Yang, and Yu-Mei Huang. “On semismooth Newton’s methods for total variation minimization”. In: *Journal of Mathematical Imaging and Vision* 27.3 (2007), pp. 265–276.
- [142] Jim Nilsson and Tomas Akenine-Möller. “Understanding ssim”. In: *arXiv preprint arXiv:2006.13846* (2020).
- [143] “Non-Local Means Denoising”. In: *Image Processing On Line* 1 (2011), pp. 208–212. doi: 10.5201/ipo1.2011.bcm_nlm.
- [144] Jens Nußberger, Frederic Boesel, Stefan Lenz, Harald Binder, and Moritz Hess. “Synthetic observations from deep generative models and binary omics data with limited sample size”. In: *Briefings in Bioinformatics* 22.4 (2021), bbaa226.
- [145] Augustus Odena, Vincent Dumoulin, and Chris Olah. “Deconvolution and Checkerboard Artifacts”. In: *Distill* (2016). doi: 10.23915/distill.00003. URL: <http://distill.pub/2016/deconv-checkerboard>.
- [146] Ido Omer and Michael Werman. “Color lines: Image specific color representation”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*. Vol. 2. IEEE. 2004, pp. II–II.
- [147] Arthur Ouaknine. “Review of deep learning algorithms for object detection”. In: *Medium*. February 5 (2018), p. 2018.
- [148] Nicola Pierazzo, Jean Michel Morel, and Gabriele Facciolo. “Multi-scale DCT denoising”. In: *Image Processing On Line* 7 (2017), pp. 288–308. ISSN: 21051232. DOI: 10.5201/ipo1.2017.201.
- [149] Tobias Plotz and Stefan Roth. “Benchmarking denoising algorithms with real photographs”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1586–1595.
- [150] Tobias Plötz and Stefan Roth. “Neural nearest neighbors networks”. In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 1087–1098.

- [151] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. “Adaptive Wiener denoising using a Gaussian scale mixture model in the wavelet domain”. In: *IEEE International Conference on Image Processing 2* (2001), pp. 37–40. doi: 10.1109/icip.2001.958418.
- [152] Ekta Prashnani, Hong Cai, Yasamin Mostofi, and Pradeep Sen. “Pieapp: Perceptual image-error assessment through pairwise preference”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1808–1817.
- [153] William K Pratt. “Generalized Wiener filtering computation techniques”. In: *IEEE Transactions on Computers* 100.7 (1972), pp. 636–641.
- [154] Peng Qiao, Yong Dou, Wensen Feng, Rongchun Li, and Yunjin Chen. “Learning non-local image diffusion for image denoising”. In: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, pp. 1847–1855.
- [155] Mohammad Saeed Rad, Behzad Bozorgtabar, Urs-Viktor Marti, Max Basler, Hazim Kemal Ekenel, and Jean-Philippe Thiran. “Srobb: Targeted perceptual loss for single image super-resolution”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2710–2719.
- [156] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).
- [157] Suman Ravuri and Oriol Vinyals. “Classification accuracy score for conditional generative models”. In: *Advances in neural information processing systems* 32 (2019).
- [158] Erik Reinhard, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.
- [159] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).
- [160] Danilo Rezende and Shakir Mohamed. “Variational inference with normalizing flows”. In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.
- [161] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. “Playing for data: Ground truth from computer games”. In: *European conference on computer vision*. Springer. 2016, pp. 102–118.
- [162] Max J Riedl. *Optical design fundamentals for infrared systems*. Vol. 48. SPIE press, 2001.
- [163] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* (1951), pp. 400–407.
- [164] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

- [165] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3234–3243.
- [166] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.
- [167] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.
- [168] Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. “Enhancenet: Single image super-resolution through automated texture synthesis”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 4491–4500.
- [169] Antoine Salmona, Valentin de Bortoli, Julie Delon, and Agnès Desolneux. “Can Push-forward Generative Models Fit Multimodal Distributions?” In: *arXiv preprint arXiv:2206.14476* (2022).
- [170] Eli Schwartz, Raja Giryes, and Alex M Bronstein. “Deepisp: Toward learning an end-to-end image processing pipeline”. In: *IEEE Transactions on Image Processing* 28.2 (2018), pp. 912–923.
- [171] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. “How good is my GAN?” In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 213–229.
- [172] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [173] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. “The JPEG 2000 still image compression standard”. In: *IEEE Signal processing magazine* 18.5 (2001), pp. 36–58.
- [174] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.
- [175] Matias Tassano, Julie Delon, and Thomas Veit. “An analysis and implementation of the ffdnet image denoising method”. In: *Image Processing On Line* 9 (2019), pp. 1–25.
- [176] Carlo Tomasi and R Manduchi. “Bilateral Filtering for Gray and Color Images”. In: *Proceedings of the 1998 IEEE International Conference on Computer Vision* 93.2 (1998), pp. 177–181. issn: 00220795. doi: 10.11677/joe.0.0930177.
- [177] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Deep image prior”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.
- [178] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. “Deep Graph-Convolutional Image Denoising”. In: *arXiv preprint arXiv:1907.08448* (2019).

- [179] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. “Image denoising with graph-convolutional neural networks”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 2399–2403.
- [180] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [181] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. “Plug-and-play priors for model based reconstruction”. In: *2013 IEEE Global Conference on Signal and Information Processing*. IEEE. 2013, pp. 945–948.
- [182] Gregory K Wallace. “The JPEG still picture compression standard”. In: *Communications of the ACM* 34.4 (1991), pp. 30–44.
- [183] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. “Non-local neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7794–7803.
- [184] Yuzhi Wang, Haibin Huang, Qin Xu, Jiaming Liu, Yiqun Liu, and Jue Wang. “Practical deep raw image denoising on mobile devices”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 1–16.
- [185] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. “Uformer: A general u-shaped transformer for image restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17683–17693.
- [186] Zhou Wang and Alan C Bovik. “Mean squared error: Love it or leave it? A new look at signal fidelity measures”. In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.
- [187] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.
- [188] Andrew B Watson. “DCTune: A technique for visual optimization of DCT quantization matrices for individual images”. In: *Sid International Symposium Digest of Technical Papers*. Vol. 24. Society for information display. 1993, pp. 946–946.
- [189] Kaixuan Wei, Ying Fu, Jiaolong Yang, and Hua Huang. “A physics-based noise formation model for extreme low-light raw denoising”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2758–2767.
- [190] Kaixuan Wei, Ying Fu, Yinqiang Zheng, and Jiaolong Yang. “Physics-based Noise Modeling for Extreme Low-light Photography”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
- [191] Xiangyu Xu, Yongrui Ma, and Wenxiu Sun. “Towards real scene super-resolution with raw images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1723–1731.
- [192] Dong Yang and Jian Sun. “BM3D-Net: A convolutional neural network for transform-domain collaborative filtering”. In: *IEEE Signal Processing Letters* 25.1 (2017), pp. 55–59.

- [193] Leonid P Yaroslavsky. *Digital picture processing: an introduction*. Vol. 9. Springer Science & Business Media, 2012.
- [194] Guoshen Yu and Guillermo Sapiro. “DCT Image Denoising: a Simple and Effective Image Denoising Algorithm”. In: *Image Processing On Line* 1 (2011), pp. 292–296. doi: 10.5201/ipo1.2011.ys-dct.
- [195] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. “Restormer: Efficient transformer for high-resolution image restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5728–5739.
- [196] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. “Cycleisp: Real image restoration via improved data synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2696–2705.
- [197] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. “Multi-stage progressive image restoration”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 14821–14831.
- [198] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. “Plug-and-play image restoration with deep denoiser prior”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
- [199] Kai Zhang, Jingyun Liang, Luc Van Gool, and Radu Timofte. “Designing a practical degradation model for deep blind image super-resolution”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 4791–4800.
- [200] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising”. In: *IEEE transactions on image processing* 26.7 (2017), pp. 3142–3155.
- [201] Kai Zhang, Wangmeng Zuo, and Lei Zhang. “Deep plug-and-play super-resolution for arbitrary blur kernels”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1671–1681.
- [202] Kai Zhang, Wangmeng Zuo, and Lei Zhang. “FFDNet: Toward a fast and flexible solution for CNN-based image denoising”. In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4608–4622.
- [203] Kai Zhang, Wangmeng Zuo, and Lei Zhang. “Learning a single convolutional super-resolution network for multiple degradations”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3262–3271.
- [204] Richard Zhang. “Making convolutional networks shift-invariant again”. In: *International conference on machine learning*. PMLR. 2019, pp. 7324–7334.
- [205] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

-
- [206] Xuaner Zhang, Qifeng Chen, Ren Ng, and Vladlen Koltun. “Zoom to learn, learn to zoom”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3762–3770.
- [207] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. “Residual dense network for image super-resolution”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2472–2481.
- [208] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. “Datasetgan: Efficient labeled data factory with minimal human effort”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10145–10155.
- [209] Yuqian Zhou, Jianbo Jiao, Haibin Huang, Yang Wang, Jue Wang, Honghui Shi, and Thomas Huang. “When awgn-based denoiser meets real noises”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13074–13081.
- [210] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.
- [211] Mingqiang Zhu, Stephen J Wright, and Tony F Chan. “Duality-based algorithms for total-variation-regularized image restoration”. In: *Computational Optimization and Applications* 47.3 (2010), pp. 377–400.

Titre : Apprentissage synthétique pour les méthodes neuronales de restauration d'images

Mots clés : restauration d'images, modélisation statistique, apprentissage profond

Résumé : La photographie est devenue un élément important de notre vie. De plus, les attentes en termes de qualité d'image augmentent alors que la taille des appareils diminue. De ce fait, l'amélioration des algorithmes de traitement d'images est essentielle.

Dans ce manuscrit, nous nous intéressons aux tâches de restauration d'images. L'objectif est de produire une image propre à partir d'une ou plusieurs observations bruitées de la même scène. Pour ces problèmes, les méthodes d'apprentissage profond ont connu une forte croissance dans la dernière décennie, surpassant l'état de l'art pour la plupart des tests traditionnels.

Bien que ces méthodes aient des résultats impressionnants, elles présentent un certain nombre de défauts. Tout d'abord, elles sont difficiles à interpréter en raison de leur fonctionnement en "boîte noire". De plus, elles généralisent assez mal à des modalités absentes de la base de données d'entraînement. Enfin, elles nécessitent de grandes bases de données, qu'il est parfois difficile d'acquérir.

On se propose d'attaquer ces différents problèmes en remplaçant l'acquisition des données par un algorithme simple de génération d'images, basé sur le

modèle feuilles mortes. Bien que ce modèle soit très simple, les images générées ont des propriétés statistiques proches de celles des images naturelles et de nombreuses propriétés d'invariance. L'entraînement d'un réseau de restauration avec ce type d'images nous permet d'identifier les propriétés importantes des images pour le succès des réseaux de restauration. De plus, cette méthode permet de s'affranchir de l'acquisition des données.

Après avoir présenté ce modèle, nous montrons que la méthode proposée permet d'obtenir des performances de restauration très proches des méthodes traditionnelles pour des tâches relativement simples. Après quelques adaptations du modèle, l'apprentissage synthétique permet également de s'attaquer à des problèmes concrets difficiles, tels que le débruitage d'images RAW. On propose ensuite une étude statistique de la distribution des couleurs des images naturelles, permettant d'élaborer un modèle paramétrique réaliste d'échantillonnage des couleurs. Enfin, nous présentons une nouvelle fonction de perte perceptuelle basée sur les protocoles d'évaluation des caméras, en utilisant les images de feuilles mortes.

Title : Synthetic learning for neural image restoration methods

Keywords : Image restoration, statistical modeling, deep learning

Abstract : Photography has become an important part of our lives. In addition, expectations in terms of image quality are increasing while the size of imaging devices is decreasing. In this context, the improvement of image processing algorithms is essential.

In this manuscript, we are particularly interested in image restoration tasks. The goal is to produce a clean image from one or more noisy observations of the same scene. For these problems, deep learning methods have grown dramatically in the last decade, outperforming the state of the art for the vast majority of traditional tests.

While these methods produce impressive results, they have a number of drawbacks. First of all, they are difficult to interpret because of their "black box" operation. Second, they generalize poorly to modalities absent from the training database. Finally, they require large databases, which are sometimes difficult to acquire.

We propose to attack these different problems by replacing the data acquisition by a simple image generation algorithm, based on the dead leaves model. Although this model is very simple, the generated

images have statistical properties close to those of natural images and many invariance properties. Training a restoration network with this kind of image allows us to identify the important properties of the images for the success of the restoration networks. Moreover, this method allows us to get rid of the data acquisition, which can be tedious.

After presenting this model, we show that the proposed method allows to obtain restoration performances very close to traditional methods for simple tasks. After some adaptations of the model, synthetic learning also allows us to tackle difficult concrete problems, such as RAW image denoising. We then propose a statistical study of the color distribution of natural images, allowing to elaborate a realistic parametric model of color sampling for our generation algorithm. Finally, we present a new perceptual loss function based on camera evaluation protocols, using the dead leaf images. The training performed with this function shows that we can jointly optimize the evaluation of the cameras, while keeping identical performances on natural images.