



école —
normale —
supérieure —
paris — saclay —

MASTER MVA : GRAPHS IN MACHINE LEARNING

RAPPORT DE PROJET

Débruitage d'image avec la méthode d'agrégation COBRA

Juliette **RENGOT**

Elève ingénieur

M. Benjamin **GUEDJ**

Superviseur

Lien vers le dépôt git : https://github.com/rengotj/cobra_denoising

De *Novembre 2018* à *Janvier 2019*

Date de rendu : 07 Janvier 2019

Table des matières

I	Introduction	2
II	Les modèles de bruit	2
II.1	Le bruit Gaussien additif	3
II.2	Le bruit poivre-et-sel	3
II.3	Le bruit de Poisson	3
II.4	Le bruit de chatolement (<i>Speckle noise</i>)	3
II.5	La suppression aléatoire de patches	4
III	Méthodes de débruitage d'images	4
III.1	Filtrage linéaire	4
III.2	Filtrage non-linéaire	4
III.3	Méthodes variationnelles	6
III.4	Méthode de moyennes non-locales	6
III.5	Déconvolution de Richardson-Lucy	7
III.6	Incorporation de données manquantes (<i>Inpainting</i>)	7
IV	Stratégie d'agrégation	8
IV.1	Schéma théorique	8
IV.2	Modèle de l'algorithme	8
IV.3	Détails d'implémentation	8
V	Expérimentation de notre approche	8
V.1	Entraînement du modèle	8
V.2	Optimisation des paramètres	10
V.3	L'évaluation des performances	10
V.4	Résultats obtenus avec un bruit Gaussien	11
V.5	Résultats obtenus avec un bruit poivre-et-sel	12
V.6	Résultats obtenus avec un bruit de Poisson	12
V.7	Résultats obtenus avec un bruit de chatolement	14
V.8	Résultats obtenus avec un bruit de suppression aléatoire de patches	14
V.9	Résultats obtenus avec plusieurs types de bruit sur une même image	16
VI	Le paramétrage automatique des filtres	16
VII	Conclusion	19
VIII	Références	20

I Introduction

Le débruitage est une question essentielle dans le traitement d'images. Il s'agit d'améliorer la qualité d'une image en retirant l'information parasite qui s'ajoute aléatoirement aux détails de la scène. Ce bruit peut être dû aux conditions de capture de l'image (manque de lumière, flou de bougé, mauvais réglage de la profondeur de champ...) ou à la caméra elle-même (augmentation de la température du capteur, erreur de transmission des données, approximations effectuées lors de la numérisation...). Le challenge consiste donc à supprimer le bruit de l'image tout en préservant sa structure. De nombreuses méthodes de débruitage existent et donnent déjà de bons résultats. Cependant, les images débruitées ont encore tendance à être trop lisses (des détails sont perdus) et un peu floues (les contours sont moins nets). Chercher à améliorer les performances de ces algorithmes est donc toujours un sujet de recherche très actif.

Ce projet propose une nouvelle approche pour le débruitage d'image. L'idée est d'utiliser les différentes méthodes de débruitage classiques pour obtenir plusieurs prédictions par pixels à débruiter. Ces valeurs doivent ensuite être agrégées, de la meilleure façon possible, pour produire un nouveau débruitage. Chaque méthode de débruitage ayant des avantages et des inconvénients, étant plus ou moins efficaces en fonction du type de bruit ou de la structure de l'image, l'objectif est d'arriver à assembler les points forts de chaque algorithme pour obtenir de meilleures performances.

Pour réaliser cette agrégation, nous allons adapter la méthode proposée par l'algorithme COBRA : COmBined Regression Alternative (Biau et al. [2015], Guedj and Desikan [2018]) au cas spécifique du débruitage d'images.

Dans ce rapport de projet, nous commencerons par décrire cinq modèles de bruit artificiel couramment utiliser. Ensuite, nous présenterons quelques méthodes de débruitage classiques et simples. Nous pourrons alors expliquer la stratégie d'agrégation de l'algorithme COBRA dans notre contexte spécifique de débruitage d'image. Par la suite, nous détaillerons notre méthode d'expérimentation et les résultats obtenus sur différents types de bruit. Enfin, nous analyserons une autre application possible de notre méthode, le paramétrage automatique des filtres, en s'appuyant sur l'exemple du filtre médian.

II Les modèles de bruit

Dans ce projet, nous n'utilisons pas d'images naturellement bruitées. Nous ajoutons artificiellement des perturbations à des images de bonne (i.e. sans bruit). Cela permet de mieux maîtriser les différents paramètres.

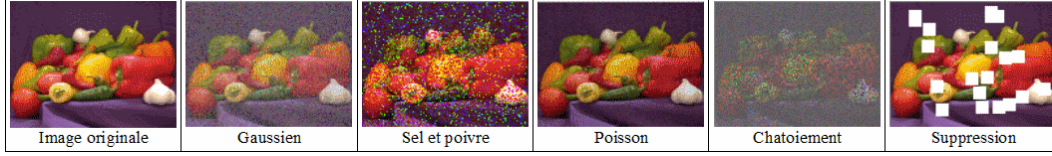


Figure 1: Les différents types de bruit utilisés.

Nous avons choisi arbitrairement cinq types de bruit: le bruit Gaussien, le bruit poivre-et-sel, le bruit de Poisson, le bruit de chatoiement et la suppression aléatoire de patches (Figure 1). D'autres types de bruit peuvent être facilement ajouté à notre modèle en rajoutant des méthodes dans la classe "noisyImage" (définie dans le fichier séparé "noise.py"). Nous allons maintenant présenter les méthodes choisies.

II.1 Le bruit Gaussien additif

On note x un signal discret non bruité, et b une variable aléatoire continue (considérée comme le bruit du signal). Le signal bruité est donné par $y = x + b$. C'est pour cela que l'on parle de bruit "additif".

Le bruit est dit "Gaussien" si sa densité de probabilité suit une loi Gaussienne:

$$\frac{1}{\sigma\sqrt{2\pi}} \cdot e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

II.2 Le bruit poivre-et-sel

Ce bruit modélise le cas où certains pixels de l'image sont saturés. Visuellement, cela se traduit par des points blancs et noirs répartis aléatoirement dans l'image avec une certaine densité. La source la plus fréquente de ce type de bruit est la présence d'un pixel mort dans le capteur.

II.3 Le bruit de Poisson

Il s'agit du bruit électronique généré par la nature quantique de la lumière. Pour une intensité moyenne I (en photon/seconde), le nombre de photons observés sur une durée T (en seconde) est une variable aléatoire distribuée selon une loi de Poisson :

$$\mathbb{P}(k|I, T) = \frac{(IT)^k \cdot e^{-IT}}{k!}$$

Le bruit de Poisson n'est pas additif mais multiplicatif : $y = x \cdot b$.

II.4 Le bruit de chatoiement (*Speckle noise*)

Le bruit de chatoiement est caractéristique des images radars et provient des diffusions multiples des ondes. Il est généralement admis qu'il suit une loi de Rayleigh dont la fonction de densité est donnée par :

$$f(x, \sigma) = \frac{x}{\sigma^2} \cdot e^{-\frac{x^2}{2\sigma^2}} \text{ pour } x \geq 0$$

Il s'agit aussi un bruit multiplicatif.

II.5 La suppression aléatoire de patches

Ici, on cherche à modéliser les pertes d'informations en supprimant aléatoirement certaines valeurs des pixels de l'image. On parle de patches car on supprime des petits voisinages de certains pixels.

III Méthodes de débruitage d'images

Nous avons choisi arbitrairement sept méthodes classiques de débruitage : le filtre Gaussien, le filtre médian, le filtre bilatéral, la méthode de Chambolle, la méthode des moyennes non-locales, la déconvolution de Richardson-Lucy et l'inpainting (Figure 2). Notre approche s'étend facilement à d'autres stratégies de débruitage. Pour cela, il suffit de rajouter des méthodes dans la classe "denoisedImage" (définie dans le fichier séparé "denoise.py"). Nous allons maintenant présenter les méthodes choisies.

III.1 Filtrage linéaire

On considère deux espaces vectoriels \mathcal{X} et \mathcal{Y} munis d'une topologie séquentielle et d'un opérateur \mathcal{A} linéaire continu. A un signal d'entrée $e \in \mathcal{X}$, on associe ainsi un signal de sortie de $s \in \mathcal{Y}$:

$$\mathcal{A} : e \mapsto s$$

\mathcal{A} est un filtre linéaire si et seulement si la relation entre e et s est une convolution. C'est pour cette raison que l'on parle aussi de filtrage par convolution. Les propriétés suivantes sont alors vérifiées :

1. Invariance dans le temps : $\mathcal{T}\mathcal{A} = \mathcal{A}\mathcal{T}$ avec \mathcal{T} l'opérateur de translation.
2. Si $\lim_{k \rightarrow +\infty} e_k(t) = e(t)$ alors $\lim_{k \rightarrow +\infty} s_k(t) = s(t)$

Le filtre Gaussien : Le noyau de convolution est alors la fonction Gaussienne de moyenne nulle suivante :

$$G(x, y) = \frac{1}{2 \cdot \pi \cdot \sigma^2} \cdot e^{-\frac{x^2+y^2}{2 \cdot \sigma^2}}$$

En pratique, on utilisera un filtre Gaussien tel que $\sigma = 0.8$. Le filtre Gaussien donne un effet de flou sur l'image débruitée : les contours en particulier ne sont plus aussi nets.

III.2 Filtrage non-linéaire

Contrairement aux filtres par convolution, les filtres non-linéaires font intervenir les pixels voisins suivant une loi non-linéaire. On cherche à donner plus importance aux pixels qui ressemblent à celui qu'on cherche à débruiter. Le but est de pallier à l'effet de flou sur les bords des objets.

	Bruit Gaussien	Bruit sel et poivre	Bruit de Poisson	Bruit de chatolement	Suppression aléatoire
Image Bruitée					
Filtrage Gaussien					
Filtrage Médian					
Filtrage Bilatéral					
Moyennes non-locales					
Total variation Chambolle					
Richardson Lucy					
Inpainting					

Figure 2: Quelques exemples d'image débruitées avec les méthodes classiques.

Le filtre médian : L'idée est de remplacer la valeur d'un pixel par la valeur médiane de son voisinage :

$$I_d(x, y) = \text{mediane}\{I(k, l) | (k, l) \in \mathcal{V}(x, y)\}$$

avec I_d l'intensité des pixels dans l'image débruitée, I l'intensité des pixels dans l'image bruitée et $\mathcal{V}(x, y)$ le voisinage du pixel de coordonnées (x, y)

Ce filtre est connu pour être particulièrement efficace sur le bruit poivre-et-sel. En pratique, on définira le voisinage comme le carré de taille 3×3 centré sur le pixel considéré.

Le filtre bilatéral : L'image est débruitée en réalisant des moyennes pondérées des pixels proches à la fois en distance et en valeur :

$$I_d(x, y) = \frac{\sum_{(k, l) \in \mathcal{V}(x, y)} I(k, l) w(x, y, k, l)}{\sum_{(k, l) \in \mathcal{V}(x, y)} w(x, y, k, l)}$$

avec $w(x, y, k, l) = e^{-\frac{(x-k)^2 + (y-l)^2}{2 \cdot \sigma_s^2} - \frac{||I(x, y) - I(k, l)||^2}{2 \cdot \sigma_c^2}}$.

Le filtre bilatéral a l'avantage de préserver les bords de l'image. En pratique, on choisira l'écart type d'écart de distance σ_s égal à 1 et l'écart type d'intensité σ_c égal à celui de l'image.

III.3 Méthodes variationnelles

Les signaux ayant trop de détails, possiblement fallacieux, possède une forte variation totale (TV : *total variation*). Cela signifie que l'intégrale de la valeur absolue du gradient du signant est grande. Réduire cette variation totale permet de supprimer les détails indésirables tout en préservant la structure de l'image.

La variation totale de Chambolle : L'algorithme proposé par Chambolle (Chambolle [2005]) permet de minimiser, par rapport à I_d , l'énergie définit par :

$$(TV_1(I_d) + \frac{||I - I_d||^2}{2})$$

avec $TV_1(I_d) = \sum ||\nabla I_d||$ la variation totale d'ordre 1.

III.4 Méthode de moyennes non-locales

Contrairement aux filtres, précédemment évoqués, qui se concentrent sur le voisinage d'un pixel, ce filtre réalise une moyenne de la totalité des valeurs des pixels contenus dans l'image, pondérées en fonction de leur similarité avec le pixel cible:

$$I_d(x, y) = \frac{1}{C(x, y)} \int_{\Omega} I(k, l) f(x, y, k, l) dk dl$$

avec Ω l'ensemble des points de l'image, f une fonction de pondération et C un facteur de normalisation défini par : $C(x, y) = \int_{\Omega} f(x, y, k, l) dk dl$. En pratique on utilise la pondération de Gauss : $f(x, y, k, l) = e^{-\frac{|\mu(x, y) - \mu(k, l)|^2}{h^2}}$ avec h un paramètre de filtrage correspond à l'écart de la Gaussienne et $\mu(p)$ la valeur moyenne des points centrés autour de p .

La méthode des moyennes non-locales est reconnue pour mieux préserver les détails de l'image à débruiter.

III.5 Déconvolution de Richardson-Lucy

Lorsqu'une image est produite en utilisant un système optique, il y a inévitablement un peu de flou. Un point source idéal n'apparaît pas comme un point. Il est étalé dans une plus grande zone appelée "point spread function" et notée P dans la suite. On note P^* l'adjoint de P .

Richardson (Richardson [1972]) et Lucy (Lucy [1974]) ont mis en place un algorithme itératif qui suit la procédure suivante (la multiplication et la division sont réalisées terme à terme) :

$$I_{n+1} = I_n \cdot \frac{I}{I_n \otimes P} \otimes P^*$$

En pratique, on définira la matrice P par :

$$\begin{pmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{pmatrix} \quad (1)$$

III.6 Incorporation de données manquantes (*Inpainting*)

Au fil du temps, une image peut être détériorée et avoir des parties manquantes où l'information est effacée. L'inpainting (Damelin and Hoang [2018], Chuiab and Mhaskar [2010]) désigne le processus de reconstruction des parties perdues. Cette reconstruction s'appuie sur les valeurs des pixels voisins non corrompus.

La première étape de l'algorithme d'inpainting consiste à trouver dans l'image les zones à traiter. En pratique on considère les zones restées totalement blanches. On suppose que la structure autour et à l'intérieur de ces zones est identique. On prolonge donc les lignes de contour à l'intérieur de la zone détériorée. On obtient ainsi différentes régions que l'on remplit avec les couleurs correspondant à celles des voisins.

Formellement, on note D la zone manquante, $S = \delta D$ la frontière de D et u_0 une fonction continue sur toutes les régions contenant D supposée connue sur un voisinage à l'extérieur de D . Le problème d'incorporation de données manquantes consiste à trouver une fonction u sur une région contenant D solution de l'équation bi-harmonique :

$$\begin{cases} \Delta^2 u = 0 \\ \Delta u|_S = \Delta u_0|_S \\ u|_S = u_0|_S \end{cases}$$

IV Stratégie d'agrégation

IV.1 Schéma théorique

Grâce à ces méthodes, pour chaque pixel p de l'image bruitée x , on dispose de M estimateurs différents $(f_1 \dots f_M)$. Dans notre cas, $M = 7$. On agrège ces estimateurs en faisant une moyenne pondérée sur les intensités :

$$f(p) = \frac{\sum_{q \in x} \omega(p, q) x(q)}{\sum_{q \in x} \omega(p, q)}$$

On définit les poids de la manière suivante :

$$\omega(p, q) = \mathbb{1}\left(\sum_{k=1}^M \mathbb{1}(|f_k(p) - f_k(q)| \leq \epsilon) \geq M * \alpha\right)$$

avec ϵ un paramètre de confiance et α un paramètre de proportion.

Ces poids signifient que pour débruiter un pixel p , on réalise la moyenne des intensités des pixels q tels qu'une proportion d'au moins α des estimateurs aient la même valeur en p et en q , avec une confiance ϵ .

IV.2 Modèle de l'algorithme

Cette méthode d'agrégation est implémentée dans la bibliothèque *pycobra* (Guedj and Desikan [2018]). Notre modèle cobra comporte sept machines, chacune réalisant un débruitage selon une méthode classique (Figure 3).

IV.3 Détails d'implémentation

L'implémentation du modèle précédemment décrit est disponible dans le fichier "denoising_cobra.py" et suit le pseudo-code de la figure 4.

L'utilisateur peut contrôler le nombre de features utilisé dans le modèle grâce au paramètre "*patch_size*". Pour chaque pixel p à débruiter, on considère le patch de l'image centré sur p de taille $(2 * patch_size + 1) \times (2 * patch_size + 1)$. En pratique $patch_size = 1$ est une valeur satisfaisant. Pour chaque pixel, on a ainsi un vecteur de neuf features.

V Expérimentation de notre approche

L'expérimentation de notre approche est menée sur des images en nuances de gris mais elle se généralise facilement à des images en couleurs.

V.1 Entraînement du modèle

On dispose de 25 images $(y_1 \dots y_{25})$, supposées sans bruit, que l'on utilise comme "vérité terrain". On ajoute artificiellement du bruit. Si on utilise tous les types de bruit disponibles,

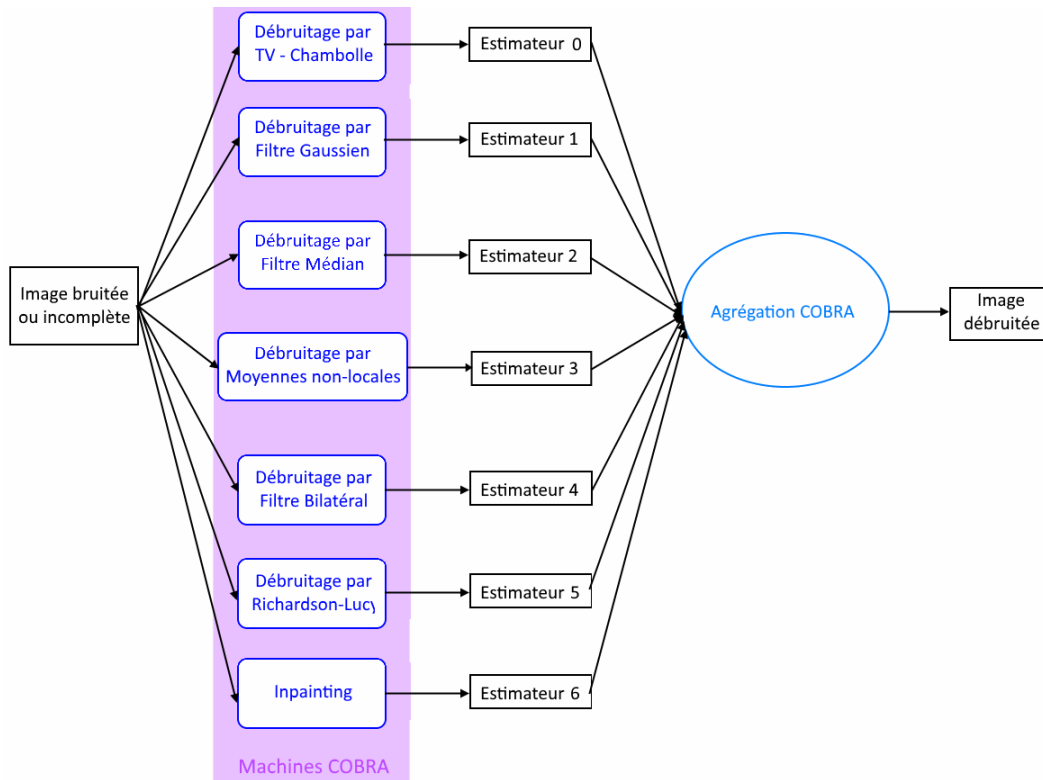


Figure 3: Schéma de notre modèle de débruitage

Debruitage_avec_cobra :

INPUT :

im_noise : l'image bruitée

p : la taille du patch de pixels à considérer

train_path : chemin d'accès aux images d'entraînement

PROCEDURE :

#Définition du modèle

Xtrain ← images artificiellement bruitées d'entraînement

Ytrain ← images originales d'entraînement

cobra ← modèle Cobra initial

cobra ← ajuster les paramètres avec les données (Xtrain, Ytrain)

cobra ← charger les 7 machines : 'bilatéral', 'nlmeans', 'gauss', 'médian', 'Tvchambolle', 'richardson_lucy' et 'inpainting'

cobra ← agréger les prédictions

#Débruitage

Xtest ← extraction des features de im_noise dans un vecteur de taille (nb_pixels, (2*p+1)²).

Y ← prédiction de Xtest par le modèle cobra

Y ← Ajout à Y des valeurs de im_noise perdues au bords à cause du travail par patch

OUTPUT :

Y : l'image débruitée

Figure 4: Pseudo-code.

on obtient ainsi 125 images bruitées ($x_1 \dots x_{125}$). Pour un modèle cobra, on doit à la fois entraîner les machines et l'agrégation. Soit on laisse le module *pycobra* séparer lui-même les images bruitées disponibles en deux groupes. Soit on crée deux observations indépendantes à partir d'une image bruitée en y ajoutant un bruit Gaussien de moyenne nulle et de variance égale à 1. Cela permet d'éviter le sur-apprentissage. Le modèle s'adapte donc mieux aux nouvelles images.

V.2 Optimisation des paramètres

Lors de la phase d'entraînement, on utilise des paramètres par défaut : $\alpha = 1$ et $\epsilon = 0.1$. Cela signifie qu'on impose que toutes les machines s'accordent sur des pixels dont la différence entre la valeur de leur estimateur et celle du pixel à débruiter ne dépasse pas 0.1. Cette stratégie n'est pas forcément la meilleure. On doit s'adapter aux données. Après l'entraînement, on utilise donc la bibliothèque "Diagnostic" de *pycobra* pour estimer les paramètres optimaux. On recommence ensuite l'entraînement avec ce nouveau paramétrage.

En pratique on peut ainsi fixer : $\alpha = 4/7$ et $\epsilon = 0.2$.

V.3 L'évaluation des performances

Pour évaluer la qualité de l'image débruitée I_d par rapport à l'image originale I_o , on utilise deux mesures :

- La racine carrée de l'erreur quadratique moyenne (RMSE : *Mean Square Error*) est définie comme la moyenne arithmétique des carrés des écarts entre les prévisions et les observations :

$$\sqrt{\frac{\sum_{x=1}^N \sum_{y=1}^M (I_d(x,y) - I_o(x,y))^2}{N \times M}}$$

On cherche à minimiser cette valeur.

- Le PSNR (*Peak Signal to Noise Ratio*) est une mesure de distorsion définie par :

$$10 \cdot \log_{10}\left(\frac{d^2}{RMSE^2}\right)$$

avec d la dynamique du signal, c'est-à-dire la valeur maximum possible pour un pixel (ici, $d = 255$).

On cherche à maximiser cette valeur.

fonctions sont implémentées dans le fichier séparé "*evaluation.py*". Il est donc facile d'ajouter d'autres types d'évaluation de la qualité de débruitage.

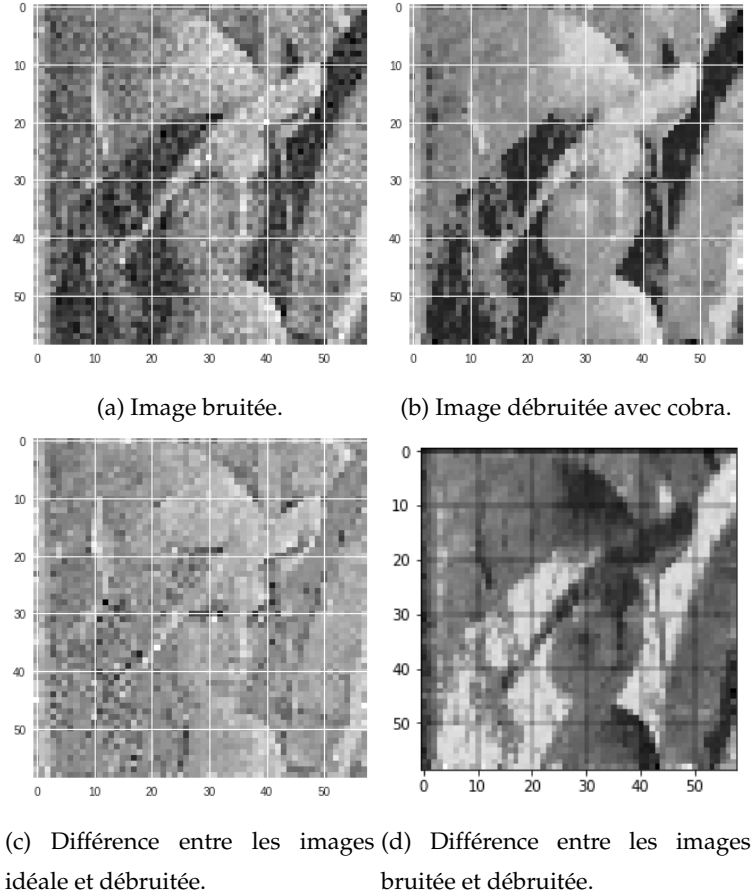


Figure 5: Résultat obtenu pour un bruit Gaussien

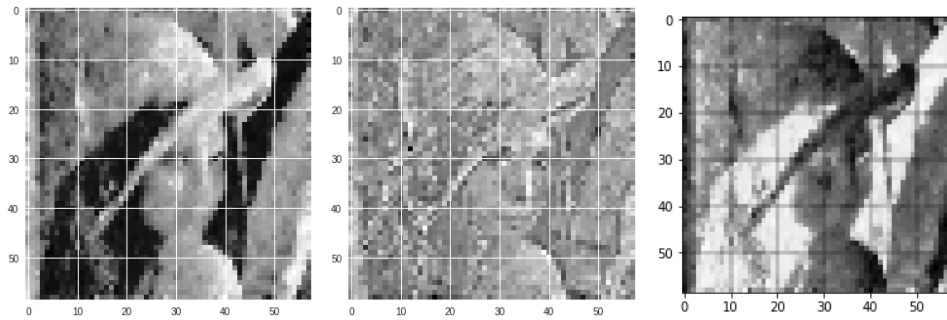
V.4 Résultats obtenus avec un bruit Gaussien

On utilise l'image de référence "lena" à laquelle on ajoute un bruit Gaussien de moyenne $\mu = 0.5$ et de variance $\sigma = 0.1$ (Figure 5a). On entraîne le modèle sur d'autres images avec tous les types de bruit disponibles. La figure 6 montre que les performances de débruitage ne sont malheureusement pas améliorées avec notre approche. Le filtre bilatéral reste le meilleur avec un PSNR supérieur d'environ $0.59dB$. On surpasse tout de même les performances de la méthode des moyennes non-locales, du filtre médian, de la variation totale de Chambolle et de la déconvolution de Richardson-Lucy. L'image obtenue (Figure 5b) est moins bruitée que l'originale. On discerne mieux les cheveux, les yeux, le nez et la bouche de Lena alors que dans l'image bruitée ces détails étaient pratiquement effacés. On perd un peu de la structure de l'image. En effet, lorsqu'on affiche la différence entre l'image idéale et l'image débruitée (Figure 5d), on ne devrait voir que le bruit. Les contours de l'image sont considérés à tort comme du bruit.

On a entraîné notre modèle sans utiliser la connaissance que l'on avait sur le type de bruit. Pour essayer d'améliorer les résultats, on recommence l'expérience en n'entraînant le

Type de débruitage		PSNR	RMSE
Cobra	Bruit inconnu	68,1863	0,0994
	Bruit connu	70,0055	0,0806
Filtre Bilatéral		68,7748	0,0929
Moyennes Non-locales		66,115	0,1261
Filtre Gaussien		68,2181	0,099
Filtre médian		64,1304	0,1585
TV-Chambolle		67,2624	0,1105
Richardson-Lucy		62,3874	0,1937
Inpainting		68,218	0,099

Figure 6: Scores obtenus avec un bruit Gaussien



(a) Image débruitée avec co- (b) Différence entre les images (c) Différence entre les images
bra. idéale et débruitée. bruitée et débruitée.

Figure 7: Résultat obtenu pour un bruit Gaussien supposé connu.

modèle que sur des images bruitées avec un bruit Gaussien (Figure 7). Le PSNR augmente d'environ $1.82dB$ (Figure 6). Ainsi, on obtient les meilleurs résultats.

V.5 Résultats obtenus avec un bruit poivre-et-sel

Ici, on utilise un bruit poivre-et-sel tel que la proportion de pixels blancs par rapport à celle de pixels noirs soit égale à $sp_ratio = 0.2$ et tel que la proportion des pixels de l'image à remplacer soit égale à $sp_amount = 0.3$. On obtient les meilleures performances de débruitage (Figure 9). Avec notre modèle, le PSNR vaut 66.3977 et la RMSE 0.1221. Auparavant, le meilleur modèle était le filtre médian ($PSNR = 63.741$ et $RMSE = 0.1658$). L'image obtenue (Figure 8b) est moins bruitée mais reste perturbée par des pixels gris indésirables.

V.6 Résultats obtenus avec un bruit de Poisson

Avec un bruit de Poisson aussi, la méthode *pycobra* améliore la qualité du débruitage (Figure 11). On obtient un PSNR de 70,0695 et une RMSE de 0.08. On dépasse donc les performances des autres méthodes, la meilleure d'entre elles étant le filtre Gaussien

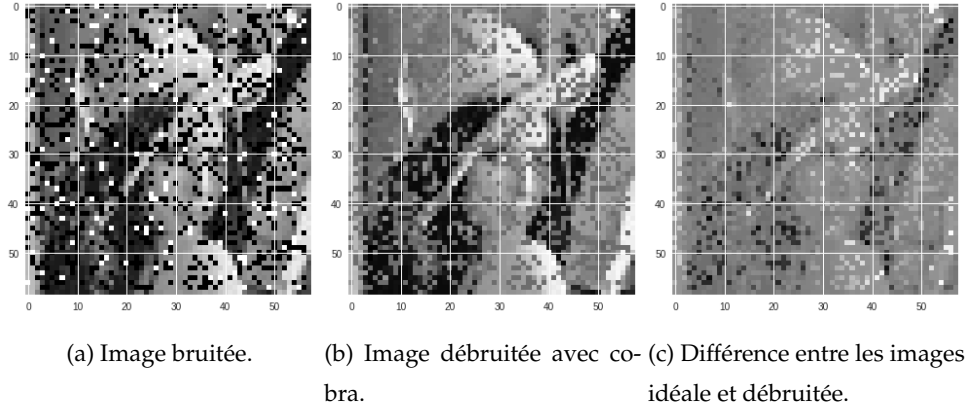


Figure 8: Résultat obtenu pour un bruit "poivre-et-sel"

Type de débruitage	PSNR	RMSE
Cobra	66,3977	0,1221
Filtre Bilateral	61,3672	0,2179
Moyennes Non-locales	59,8005	0,2609
Filtre Gaussien	59,6558	0,2653
Filtre médian	63,741	0,1658
TV-Chambolle	63,2861	0,1747
Richardson-Lucy	60,8724	0,2306
Inpainting	60,4111	0,2432

Figure 9: Scores obtenus avec un bruit poivre-et-sel

($PSNR = 68.8177$ et $RMSE = 0.0924$). L'image obtenue (Figure 10b) est beaucoup plus nette.

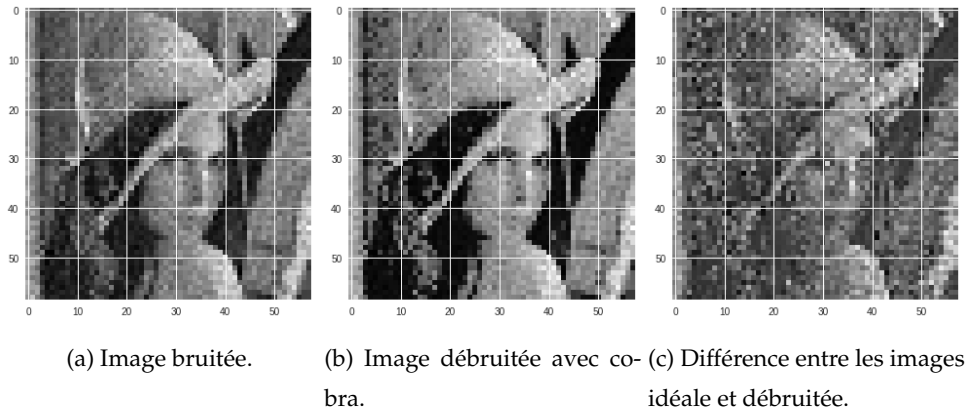


Figure 10: Résultat obtenu pour un bruit de Poisson

Type de débruitage	PSNR	RMSE
Cobra	70,0695	0,08
Filtre Bilateral	68,6073	0,0947
Moyennes Non-locales	66,7181	0,1177
Filtre Gaussien	68,8177	0,0924
Filtre médian	63,9126	0,1625
TV-Chambolle	67,3367	0,1096
Richardson-Lucy	62,3064	0,1955
Inpainting	68,8065	0,0925

Figure 11: Scores obtenus avec un bruit de poisson

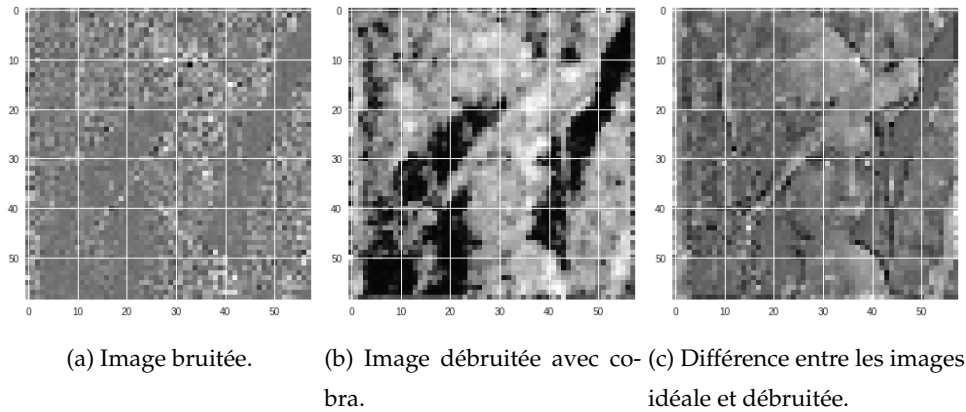


Figure 12: Résultat obtenu pour un bruit de chatolement

V.7 Résultats obtenus avec un bruit de chatolement

Pour un bruit de chatolement (Figure 12a), la figure 13 montre que les performances de débruitage sont nettement aussi améliorées avec notre approche. On a un PSNR de $65.1268dB$ alors que celui du filtre bilatéral, qui était la meilleure des méthodes classiques, ne valait que $61.8937dB$. L'image obtenue (Figure 12b) est moins bruitée mais reste d'assez mauvaise qualité. Ce bruit est particulièrement difficile. En effet, sur l'image bruitée, on ne perçoit presque plus la structure de l'image.

V.8 Résultats obtenus avec un bruit de suppression aléatoire de patches

On supprime aléatoirement 20 patches, de taille (4×4) , de l'image originale. Ces pixels deviennent blancs (Figure 14a). Dans ce cas, notre méthode n'est pas la plus performante. L'inpainting classique atteint un PSNR de 77.8956 alors que notre méthode garde un PSNR de $74,1915$ (Figure 15). Les autres approches sont beaucoup moins efficaces car elles n'ont pas été conçues pour cette situation. L'image obtenue (Figure 14b) ne présente plus de carré blanc. L'information a été restaurée sur la totalité de l'image. On perçoit tout de même des zones plus claires dans l'image débruitée qui ne devraient pas exister.

Type de débruitage	PSNR	RMSE
Cobra	65,1268	0,1413
Filtre Bilatéral	61,8937	0,205
Moyennes Non-locales	61,3119	0,2193
Filtre Gaussien	61,661	0,2106
Filtre médian	61,6672	0,2105
TV-Chambolle	61,761	0,2082
Richardson-Lucy	59,9156	0,2575
Inpainting	61,6607	0,2106

Figure 13: Scores obtenus avec un bruit de chatolement

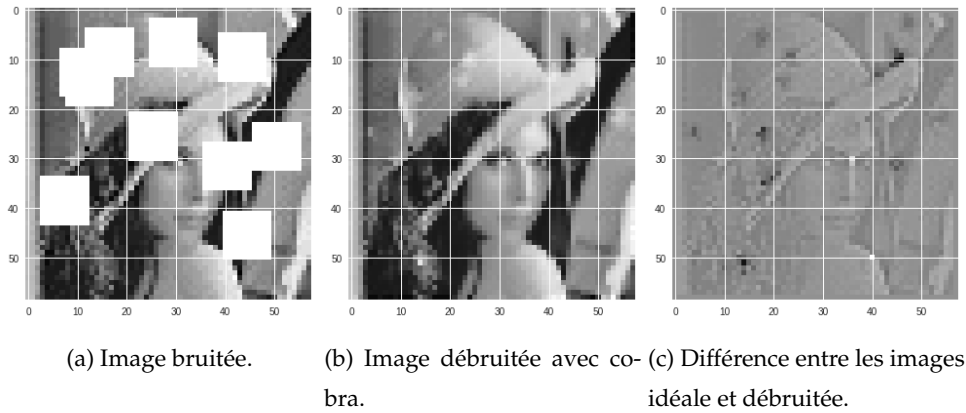


Figure 14: Résultat obtenu avec une suppression aléatoire de patches.

Type de débruitage	PSNR	RMSE
Cobra	74,1915	0,0498
Filtre Bilateral	63,6138	0,1682
Moyennes Non-locales	63,5052	0,1703
Filtre Gaussien	63,5326	0,1698
Filtre médian	64,0806	0,1594
TV-Chambolle	64,1667	0,1578
Richardson-Lucy	60,9729	0,228
Inpainting	77,8956	0,0325

Figure 15: Scores obtenus avec une suppressions aléatoire de patches

V.9 Résultats obtenus avec plusieurs types de bruit sur une même image

On peut se demander comment se comporte l'algorithme lorsqu'une seule image présente plusieurs types de bruit. On définit alors un "bruit multiple" (Figure 16a). On applique un bruit Gaussien dans le coin supérieur droit, un bruit poivre-et-sel dans le coin supérieur gauche, un bruit de poisson dans le coin inférieur gauche et un bruit de chatolement dans le coin inférieur droit. De plus, on supprime aléatoirement des petits patches dans toutes l'images.

Les méthodes classiques ne proposent pas un débruitage efficace. Les images débruitées (Figure 16) sont très éloignées de la réalité. On ne retrouve plus vraiment la structure de l'image. Les quatre zones distinctes sont encore souvent facilement discernables.

Notre méthode améliore nettement la qualité du débruitage. Le PSNR atteint $65.4311dB$ (Figure 18). On a gagné environ $1.7dB$ par rapport à TV-Chambolle qui était le meilleur débruitage pour cette situation. On obtient une image débruitée (Figure 17a) dans laquelle on reconnaît mieux le visage de Lena. Il n'y a plus de carré blanc sans information. Les bruits Gaussien, de Poisson et poivre-et-sel ont été efficacement compensés et ces parties de l'images sont homogènes. Seul le coin inférieur droit se démarque encore. Le bruit de chatolement est le moins bien traité. On note aussi que les contours et les textures sont particulièrement difficiles à débruiter.

Si l'on entraîne notre modèle seulement sur des images avec du bruit multiple, on améliore entre la qualité du débruitage ($PSNR = 66.51$ et $RMSE = 0.1205$). La figure 19 permet de visualiser l'image obtenue.

VI Le paramétrage automatique des filtres

Au cours de notre expérimentation, nous avons pu constater que les méthodes classiques pouvaient être efficaces sur certains types de bruit. Par exemple, le filtre médian est particulièrement adapté au bruit poivre-et-sel mais la taille du filtre choisi influence beaucoup la qualité de l'image débruitée. Plus le paramètre "*sp_amount*" est élevé, plus il est judicieux de prendre une grande taille de filtre. Notre méthode peut être utilisée pour paramétrer automatiquement les méthodes classiques. On entraîne notre modèle avec un seul type de méthode mais pour plusieurs valeurs du paramètre à régler. Par exemple, on choisit comme machines le filtre médian avec trois tailles différentes : 3, 5 ou 10. Peu importe la quantité de bruit présent dans l'image ($sp_amount = 0.1$, $sp_amount = 0.3$ ou $sp_amount = 0.5$) la méthode de débruitage cobra est la plus performante (Figure 20). On a donc plus besoin de connaître à l'avance les propriétés du bruit pour débruiter de manière efficace.

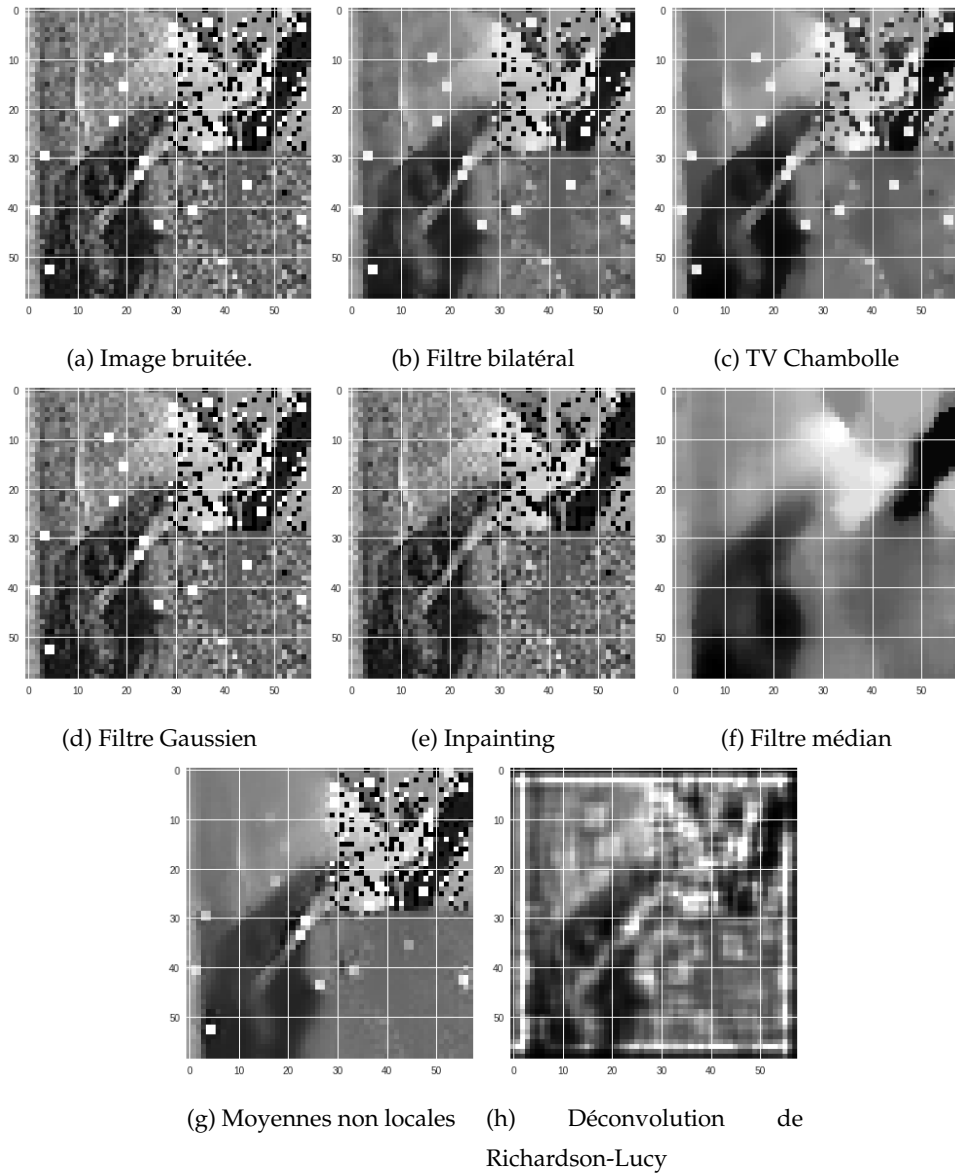
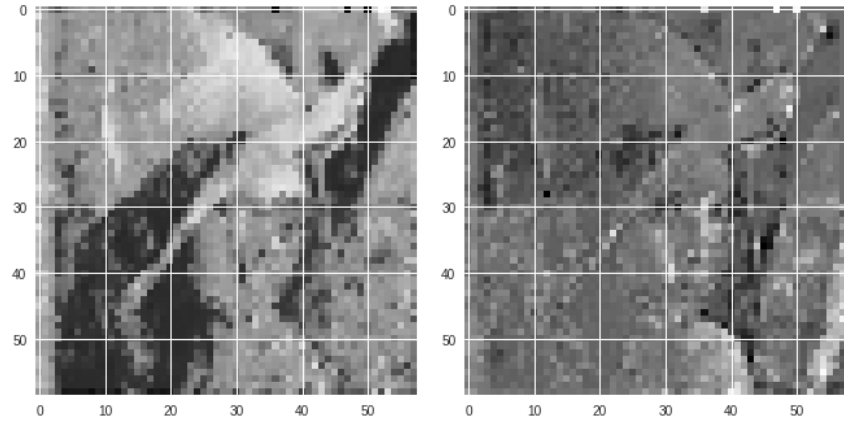


Figure 16: Débruitage d'image avec un bruit multiple par les méthodes classiques

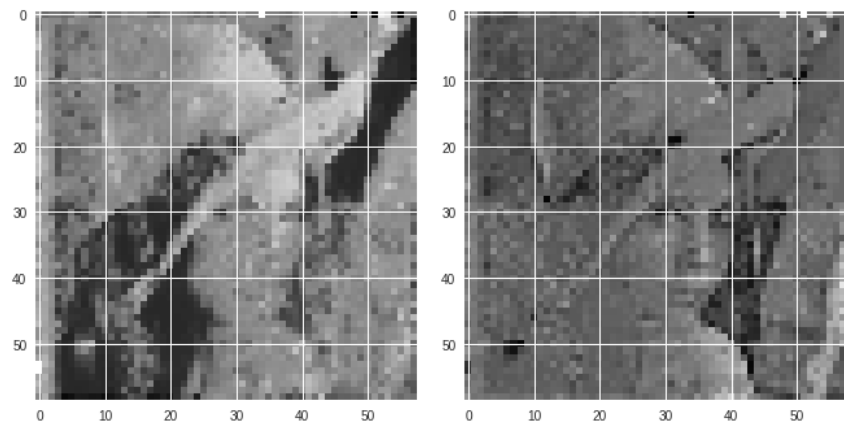


(a) Image débruitée avec cobra. (b) Différence entre les images idéale et débruitée.

Figure 17: Résultat obtenu avec notre méthode pour un bruit multiple.

Type de débruitage		PSNR	RMSE
Cobra	Bruit inconnu	65,431	0,1365
	Bruit connu	66,51	0,1205
<i>Filtre Bilatéral</i>		62,417	0,1931
<i>Moyennes Non-locales</i>		61,825	0,2067
<i>Filtre Gaussien</i>		61,793	0,2074
<i>Filtre médian</i>		63,393	0,1725
<i>TV-Chambolle</i>		63,735	0,1659
<i>Richardson-Lucy</i>		61,029	0,2265
<i>Inpainting</i>		62,71	0,1867

Figure 18: Scores obtenus avec un bruit multiple.



(a) Image débruitée avec cobra. (b) Différence entre les images idéale et débruitée.

Figure 19: Résultat obtenu avec un bruit multiple connu.



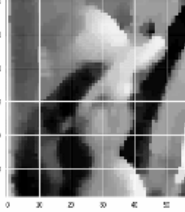
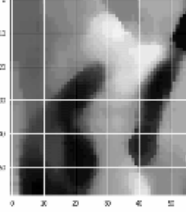

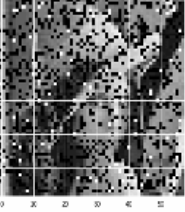
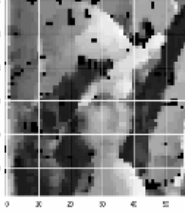
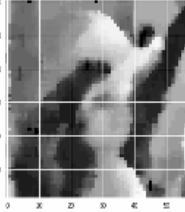
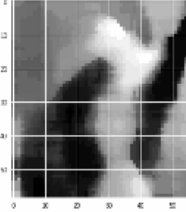
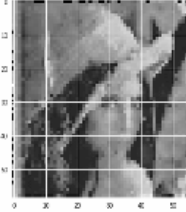
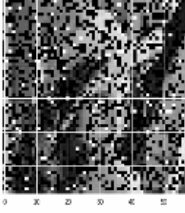
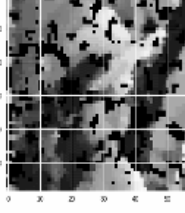
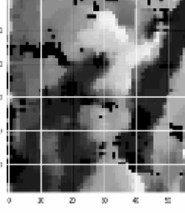
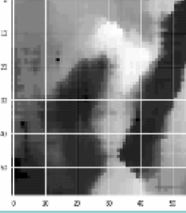
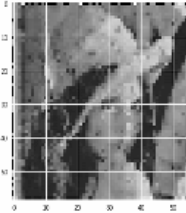
Image bruitée <u>sp_ratio</u> = 0,1 <u>sp_amount</u> = 0,1	Filtre médian (taille 3) PSNR = 70,2194 RMSE = 0,0786	Filtre médian (taille 5) PSNR = 67,9888 RMSE = 0,1016	Filtre médian (taille 10) PSNR = 65,1102 RMSE = 0,1415	Méthode Cobra PSNR = 73.6212 RMSE = 0.05314
				
<u>sp_ratio</u> = 0,1 <u>sp_amount</u> = 0,3	PSNR = 64.5019\$ RMSE = 0.1518	PSNR = 65.6326 RMSE = 0.1333	PSNR = 63.6455 RMSE = 0.1675	PSNR = 69.2125 RMSE = 0.0882
				
<u>sp_ratio</u> = 0,1 <u>sp_amount</u> = 0,5	PSNR = 59.7381 RMSE = 0.2628	PSNR = 60.8328 RMSE = 0.2317	PSNR = 61.0161 RMSE = 0.2268	PSNR = 66.8859 RMSE = 0.1154
				

Figure 20: Résultats obtenus pour le paramétrage du filtre médian

VII Conclusion

Finalement, notre méthode de débruitage utilisant l'agrégation COBRA améliore les performances des méthodes classiques. L'un des avantages de notre méthode est de ne pas être spécifique à un type de bruit donné (comme par exemple le filtre médian qui est utilisé spécialement pour le bruit poivre-et-sel). On obtient donc des images débruitées de bonne qualité sans avoir besoin de connaître le type de bruit présent dans l'image. Cet atout est particulièrement utile lorsque l'image présente un bruit multiple. Notre méthode est alors la seule à donner un débruitage satisfaisant.

Lorsque l'on dispose de suffisamment d'images d'entraînement, on peut tout de même utiliser la connaissance sur le type de bruit présent dans l'image à débruiter (c'est-à-dire entraîner le modèle uniquement sur ce type de bruit) pour améliorer encore les résultats.

Notre approche se révèle aussi efficace dans le paramétrage automatique des méthodes classiques de débruitage comme le montre l'exemple du filtre médian.

Cette méthode de débruitage est donc prometteuse. Il serait intéressant de l'étendre à des méthodes classiques plus sophistiquées comme par exemple le BM3D, la méthode Bayésienne non-locale ou le DnCNN.

VIII Références

Gérard Biau, Aurélie Fischer, Benjamin Guedj, and James D. Malley. Cobra : A comined regression strategy. *Journal of Multivariate Analysis*, pages 18–28, 04 2015.

Benjamin Guedj and Bhargav Srinivasa Desikan. Pycobra : A python toolbox for ensemble learning and visualisation. *Journal of Machine Learning Research*, pages 1–5, 04 2018.

Antonin Chambolle. Total variation minimization and a class of binary mrf models. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 3757:132–152, 2005.

William Hadley Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62:55–59, 1972.

Leon Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 19:745, 06 1974.

Steven Damelin and Nguyen Hoang. On surface completion and image inpainting by biharmonic functions: Numerical aspects. *International Journal of Mathematics and Mathematical Sciences*, 2018:8, 01 2018.

Charles Chuiab and Hrushikesh Mhaskar. Mra contextual-recovery extension of smooth functions on manifolds. *Applied and Computational Harmonic Analysis*, 28:104–113, 01 2010.