Projet ARF
Inpainting

Zanivan Thomas, Marlot Maxime

10 juin 2018

Table des matières

1 Introduction

Dans ce rapport nous nous intéressons à une méthode pour reconstruire une image détériorée ou avec des parties manquantes. Pour ce faire nous utilisons les patchs non détériorés pour créer un dictionnaire et ainsi trouver grâce à LASSO la combinaison de patch la plus intéressante.

2 Prise en main de LASSO

La régression ridge favorise un nombre quelconque de coefficients non-nuls mais avec de très petites valeurs absolues (proches de 0). Alors que la régression lasso favorise un petit nombre de coefficients non-nuls avec de petites valeurs absolues.

De plus on observe que le lasso effectue une sélection des features en mettant un grand nombre de features à 0. Ce qui est particulièrement utile dans notre contexte où on peut avoir un très grand nombre de patchs redondants dans le dictionnaire, dans ce cas le lasso en choisit un avec un gros coefficient et met les autres à 0.

3 Résultats d'expériences par reconstruction simple

3.1 avec un alpha par image

Dans un premier nous testons une méthode naïve qui ne gère pas le bruit sur les bords de l'image (nous expliquons juste après comment nous l'avons géré) et qui utilise un alpha fixe pour tous les pixels corrigés.

FIGURE 3.1 – Correction d'une image faiblement bruitée (b=0.01)

(a) Image source

(b) Image reconstituée

FIGURE 3.2 – Correction d'une image fortement bruitée (b=0.3)

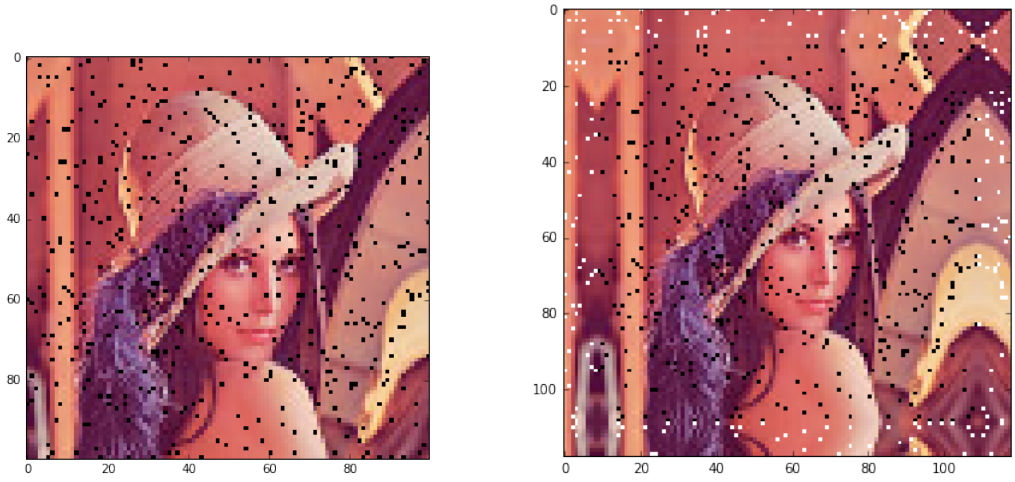
(a) Image source

(b) Image reconstituée

On peut voir que dans certains cas, les valeurs prédites ne sont pas très cohérentes et que notamment avec un bruit fort, on peut obtenir des valeurs en prédiction qui sortent du range [0,255] et donne donc des couleurs totalement incohérentes. Pour éviter ce type d'erreur on peut recalculer le alpha pour chaque pixel bruité avant de lancer le fit du modèle. Même si c'est beaucoup plus long, c'est ce que nous ferons dans la suite.

3.2 gestion du bruit sur les bords de l'image

Maintenant, pour pouvoir gérer le bruit sur les bords de l'image, on procède à une extension en miroir des bords telle qu'illustrée ci-dessous.

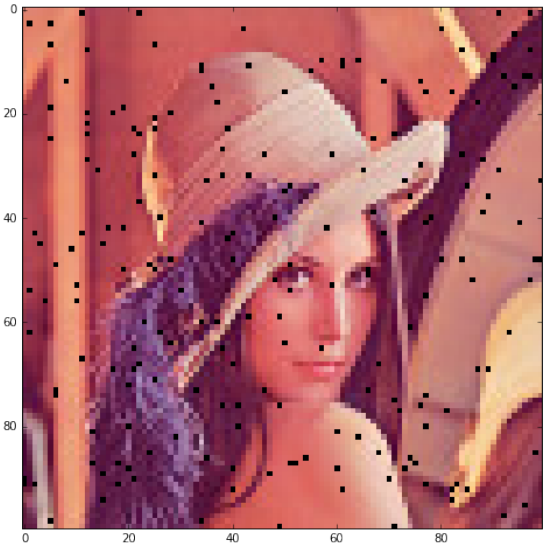


(a) Source bruitée

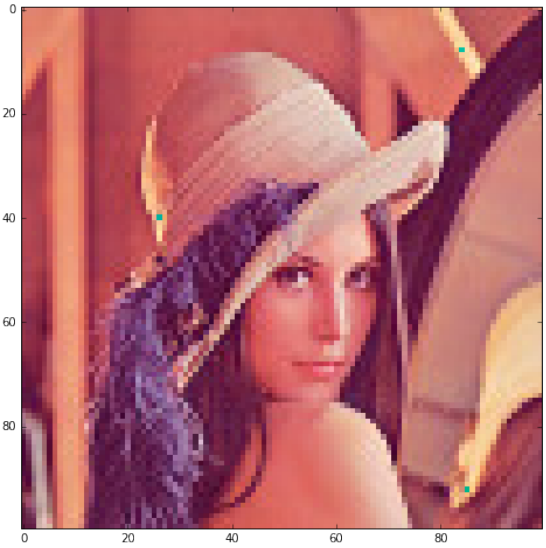
(b) extension de l'image avec effet miroir sur les bords (avec en blanc le bruit sur la partie étendue)

3.3 avec du bruit sur les bords et un alpha par pixel bruité

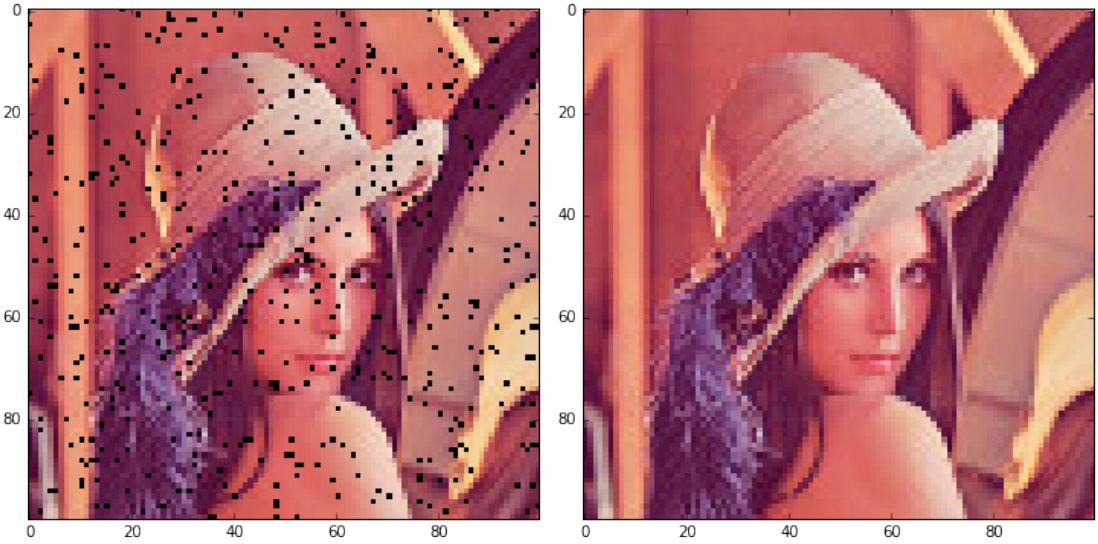
En combinant le recalcule du alpha pour chaque pixel et en utilisant la méthode d'extension d'image par miroir sur les bords précédente, on obtient maintenant de bons résultats tel que les exemples obtenus ci-dessous.



(a) Image source faiblement bruitée



(b) Image reconstituée en utilisant l'alpha optimal pour chaque pixel et une extension des bords en miroir



(a) Image source avec beaucoup de bruit

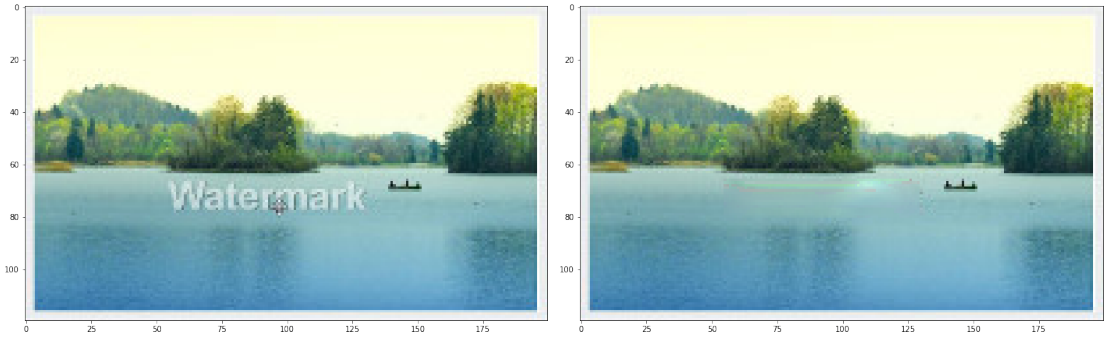
(b) correction pixel par pixel avec predict, recalcule du alpha pour chaque pixel (patch de taille 3)

4 Résultats d'expériences par reconstruction en spirale pixel par pixel

Dans cette section nous poursuivons la reconstruction d'une image à partir des patchs "propres" mais en utilisant une stratégie différente. En effet lorsque nous avions tout un carré de l'image qui était bruité, il était difficile d'obtenir un résultat satisfaisant sans reconstruire l'image dans un ordre précis. Pour ce faire nous décidons de reconstruire l'image en spirale (ou escargot), en recalculant à chaque itération le alpha pour le nouveau patch à reconstruire (grâce à LassoCV). Cela revient à remplir d'abord les bords puis à aller vers le centre.

Ce processus est long mais reste celui avec lequel nous avons obtenu les meilleurs résultats comme on peut le constater en 4.1 ci-dessous.

FIGURE 4.1 – Suppression d'un watermark



(a) Image source

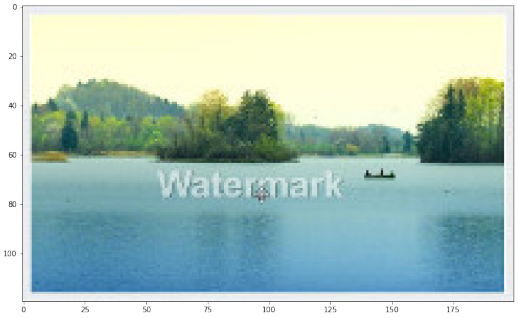
(b) Image reconstituée en utilisant l'alpha optimal pour chaque pixel

5 Résultats d'expériences par reconstruction en spirale par patchs entiers

Dans la section précédente nous avons utilisé une méthode qui reconstruit l'image en spirale en calculant pour chaque patch l'alpha optimal, ce qui demandait plusieurs dizaines de minutes pour une image de seulement 128x256 pixels.

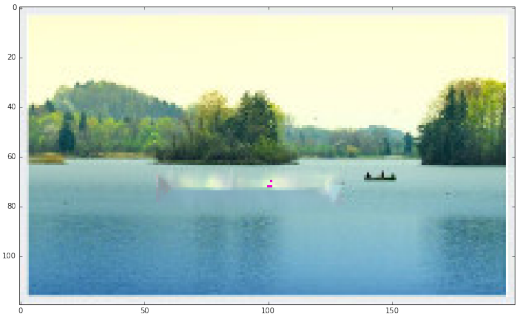
Nous souhaitons à présent essayer d'améliorer la rapidité de notre algorithme quitte à perdre en précision. Pour cela nous avons essayé plusieurs idées, tout d'abord lorsqu'un patch à reconstruire possède plusieurs pixels bruités, nous admettons qu'un pixel prédit peut être utilisé pour les autres pixels bruités de ce patch. Dans un deuxième temps nous décidons de tester le calcul d'un alpha moyen sur un ensemble de patch et de l'utiliser pour tous les patchs en prédiction. Ces deux méthodes ont pour avantage d'accélérer fortement le calcul de reconstruction, cependant on peut voir sur la sous-figure 5.1.(c) ci-dessous que les résultats sont moins précis qu'en recalculant à chaque fois le alpha comme sur la sous-figure (d).

FIGURE 5.1 – Suppression d'un watermark

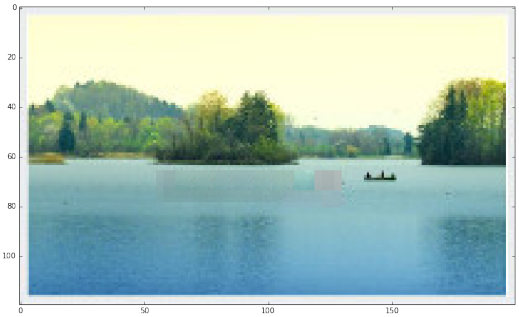


(a) Image source

(b) suppression du watermark



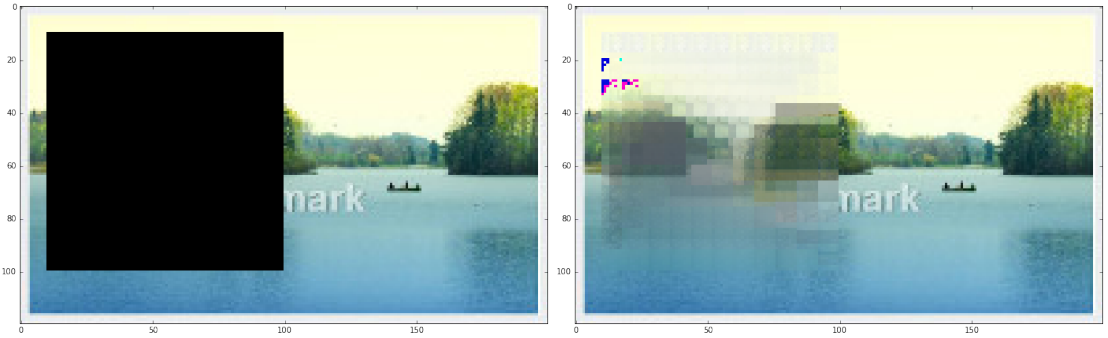
(c) Reconstitution par patch en prenant une seule moyenne de la valeur alpha pour tous les patchs



(d) Reconstitution par patch via predict avec recalcule du alpha par patch

On a pu observer qu'utiliser des patchs de petites tailles (idéalement de taille 3, le plus petit) donne de biens meilleurs résultats que des patchs de grande taille, même si cela est plus long.

FIGURE 5.2 – Reconstruction d'un grand carré manquant



(a) Source dont on a enlevé un grand carré

(b) Résultat de la reconstruction avec un patch de taille 15

Évidemment on constate que si une trop grande partie de l'image est manquante la reconstruction est difficile mais on obtient quand même quelque chose de cohérent.

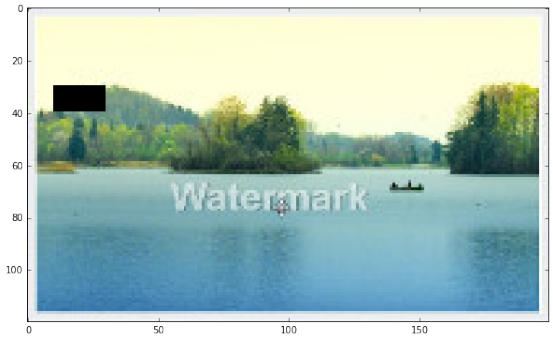
5.1 avantages de cette méthode

Dans ce projet nous avons pu expérimenter plusieurs méthodes de reconstruction d'une image bruité, y compris avec toute une portion de l'image manquante. Nous avons obtenu les meilleurs résultats en utilisant une méthode de reconstruction en spiral pixel par pixel ainsi qu'en recalculant la valeur du alpha pour LASSO à chaque nouveau pixel. Ce processus s'est montré long mais offre les meilleurs résultats. Une méthode alternative a été de remplir par patchs entiers et de calculer un unique alpha moyen. Le meilleur compromis temps/résultat s'avère être l'utilisation d'un remplissage par patch d'assez petite taille, avec un recalcul du alpha pour chaque patch.

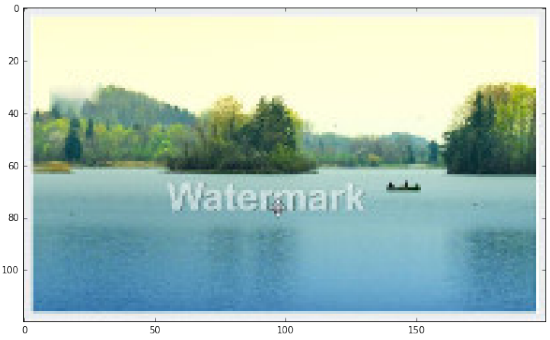
5.2 inconvénients de cette méthode et variations

Cette méthode fonctionne bien mais provoque souvent un effet de flou lorsqu'on utilise le predict. Pour palier à ce problème on peut utiliser à la place l'argmax, c'est-à-dire prendre directement le patch le plus proche pour remplir, tel qu'illustré ci-dessous.

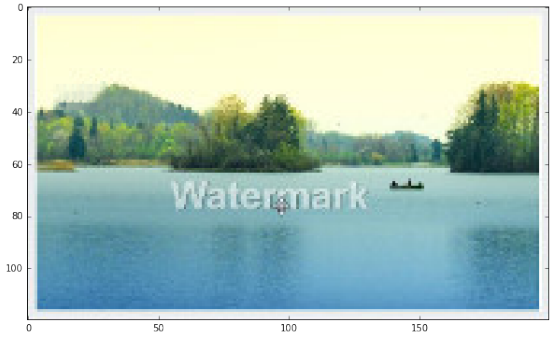
FIGURE 5.3 – Reconstruction d'un carré bruité : résolution du flou



(a) Source dont on a enlevé un rectangle



(b) Résultat de la reconstruction avec predict, effet de flou



(c) Résultat de la reconstruction avec argmax, pas d'effet de flou

Cependant cette méthode peut provoquer des effets de bords, on a observé que dès lors qu'on se trompe en remplissant d'une manière un peu erronée, comme on va ensuite chercher à corriger le patch juste à coté en spirale, on va propager l'erreur. On se rend donc compte que l'ordre de remplissage a une importance et que se contenter de remplir des bords vers le centre en spirale n'est pas forcément suffisant.

5.3 Possibilités d'améliorations

Comme expliqué dans l'article de recherche [3], on peut améliorer ce procédé en procédant à un tri à chaque itération de la priorité des patchs à corriger sur la bordure et en priorisant les patchs qui contiennent un gradient élevé en plus de contenir beaucoup de pixels déjà valides. Pour cela on applique un filtre de détection de bords (gradient de l'image) comme illustré ci-dessous.

(a) Source

(b) après détection de bords, pour prioriser les patchs

De cette manière on priorise les patchs qui contiennent des éléments de continuité visuelle et on évite les étalements en effets de bords comme vu dans les inconvénients de la méthode précédente, ce qui donne des résultats bien plus cohérents avec une véritable photographie.

6 Références

[1] Bin Shen and Wei Hu and Zhang, Yimin and Zhang, Yu-Jin, Image Inpainting via Sparse Repre- sentation Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '09

[2] Julien Mairal Sparse coding and Dictionnary Learning for Image Analysis INRIA Visual Recogni- tion and Machine Learning Summer School, 2010

[3] A. Criminisi, P. Perez, K. Toyama Region Filling and Object Removal by Exemplar-Based Image Inpainting IEEE Transaction on Image Processing (Vol 13-9), 2004