INSTITUT DE MATHEMATIQUES DE BORDEAUX (IMB) - Juillet 2019

RAPPORT DE STAGE DE FIN DE MASTER 1

ETUDE D'UN ARTICLE SUR LE DEBRUITAGE DE VIDEO PAR PARCIMONIE ET APPROXIMATION DE RANG FAIBLE

Encadré par : Yann Traonmilin

Fait par :
Paul Gicquel

Table des matières

1 Notations

— σ écart type pour le bruit gaussien
— p écart en pixel entre le centre de deux patchs consécutifs
— K nombres de patchs similaires
— U_i matrice contenant les K patch les plus proches du patch i
— N nombre de patch sur l'image
— $V = (U_i)_{i=1}^{N}$
— W transformation pour la parcimonie
— $\hat{D}_i$ ou D_i matrice de rang faible
— m nombres d'image de la fenêtre temporelle (impaire)
— $SVD(M)$ la décomposition en valeurs singulières de la matrice M
— H_ρ l'opérateur de hard-tresholding
— γ_s le poids de la partie parcimonie
— γ_l le poids de la partie rang faible
— γ_f le poids de la matrice de départ

2 Introduction

J'ai effectué un stage de 2 mois à la fin de mon Master 1 : "EDP modélisation" à l'IMB durant lequel j'ai travaillé sur du traitement d'image. J'ai étudié l'article [1] qui porte sur le débruitage de vidéo. Le bruit apparaît sur les photos ou les vidéos prises en condition de basse luminosité, on peut modéliser ce bruit par un bruit gaussien. Ici, on va débruiter la vidéo image par image à l'aide des autres images composant la vidéo. Ainsi on cherche à approximer u l'image sans bruit à partir de l'image capturée v où $v = u + \epsilon$ avec ϵ suivant une loi normale. On va travailler sur des images en niveaux de gris, chaque image est représenté par une matrice avec des coefficients entre 0 et 255.
Dans ce rapport, je vais tout d'abord expliquer brièvement l'algorithme de l'article puis expliquer comment je l'ai codé en Python, quels modifications j'ai essayé de faire et ensuite présenter les différents résultats que j'ai obtenu.

3 Présentation de l'algorithme global

3.1 Découpage de l'image en patchs

Pour débruiter la vidéo on va tout d'abord découper chaque image en patchs se chevauchant et de même taille (par exemple de taille 8x8 pixels).
On va ensuite débruiter chaque image, prenons une image de la vidéo, alors pour chaque patch de l'image on va chercher les K ($K = 15$ par exemple) patchs les plus ressemblant, ici les plus proches pour la distance euclidienne (somme des différences de chaque pixels au carré) dans l'image mais aussi dans les $m/2 - 1$ images précédentes et $m/2 - 1$ images suivantes de la vidéo (si elles existent) (on prendra m entre 2 et 10). Pour un gain de temps, on ne va pas chercher dans l'entièreté de chaque image mais dans une fenêtre autour du patch de départ (exemple, une fenêtre de 40x40 pixels).
On note pour le patch numéro i de l'image à débruiter la matrice U_i des K patchs les plus proches, où la colonne k de U_i est le k-ième patch le plus proche (avec chaque patch stocké en une colonne et donc concaténé colonne par colonne). On va ensuite travailler sur les matrices U_i de l'image.

3.2 Parcimonie et rang faible

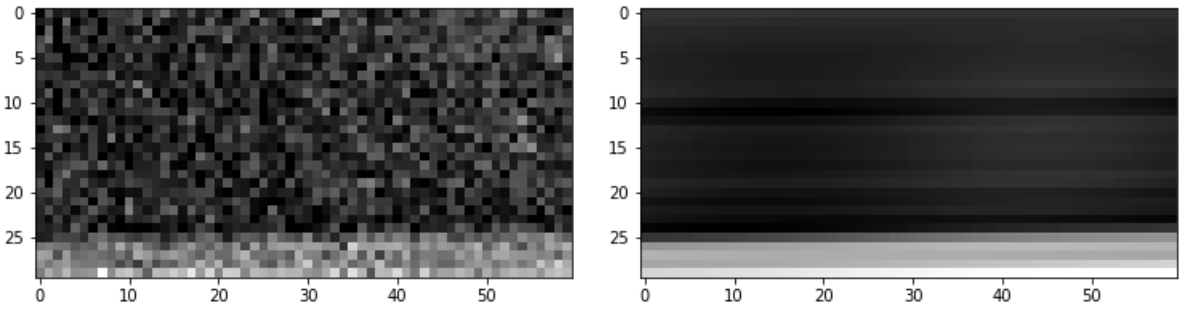
On va d'un côté, approximer la matrice U_i par une matrice de rang plus faible. On a une double contrainte, rang faible et proche de U_i, ce qui correspond à chercher :

$$\hat{D_i} = \underset{D_i}{\operatorname{argmin}} \; ||U_i - D_i||_F^2 + \theta^2 rang(D_i) \tag{1}$$

Cette opération va débruiter l'image, par exemple si on a deux patchs qui correspondent à la même image sans bruit alors dans D_i ces deux patchs vont être remplacer par deux colonnes égales qui seront la moyenne des deux patchs, ainsi un patch débruité.
D'un autre côté, on va rendre la matrice U_i parcimonieuse dans un certain domaine, c'est à dire qu'après une certaine transformation linéaire la matrice sera composée de nombreux zéros. En notant u_i le vecteur obtenu par concaténation des colonnes de U_i, on va chercher un vecteur contenant peu d'éléments non nuls et proches de $W\,u_i$ après transformation :

$$\hat{\alpha_i} = \underset{\alpha_i}{\operatorname{argmin}} \; ||W\,u_i - \alpha_i||_2^2 + \rho^2\,||\alpha_i||_0 \tag{2}$$

avec $||x||_0$ égale au nombre de d'éléments non nuls de x et W la matrice que l'on va choisir.
Ensuite l'image débruitée est $d_i = W^{-1}\,\hat{\alpha_i}$.
Si une matrice est parcimonieuse alors l'information contenue dans la matrice est inférieur à sa taille et donc les composantes de la matrice dans l'espace de départ sont liés ainsi cela aura pour effet de lisser l'image. Par exemple, en prenant l'image de gauche ci-dessous, W la transformée en cosinus discrète et $\rho = 50$ on obtient l'image de droite :



Ainsi on a un débruitage à l'aide de la structure global de l'image (patch par patch) avec l'approximation de rang faible et un débruitage local (pixel par pixel) avec la parcimonie. Dans l'article, les deux sont fait séparément et le résultat de chaque opération est simplement additionné. J'ai aussi essayé d'écrire un algorithme qui fait les deux à la fois pour obtenir une matrice parcimonieuse et de rang faible.

3.3 Reconstruction de l'image

Enfin, on reconstruit l'image en sommant chaque patch contenu dans les matrices U_i avec un poids pour chaque patch correspondant au nombre de fois où il apparaît dans $V = (U_i)_i$.

4 Détails sur la partie parcimonie et rang faible

4.1 Approximation de rang faible

Tout d'abord, pour l'approximation de rang faible on connaît une solution à l'équation (1) donnée par le théorème suivant.

Théorème 1. *Soit* $A \in \mathbb{R}^{m\times n}$.
Si on note $SVD(A) = U\Sigma V^T$ *et* H_ρ *l'opérateur de hard-tresholding défini par :*

$$(H_\rho(x))_i = \begin{cases} 0 \ si \ |x_i| < \rho \\ x_i \ si \ |x_i| \geq \rho \end{cases}$$

alors

$$\underset{D}{argmin} \ ||A - D||_F^2 + \theta^2 rang(D) = UH_\theta(\Sigma)V^T \tag{3}$$

On va démontrer le théorème 1 à l'aide du théorème suivant :

Théorème 2. *Soit* $A \in \mathbb{R}^{m\times n}$. *On note* $0 \leq \sigma_n \leq \sigma_{n-1} \leq ... \leq \sigma_1$ *les valeurs singulières de* A, $\Sigma_r = diag(\sigma_1, ..., \sigma_r, 0, ..., 0)$.

$$\underset{L, rang(L)\leq r}{argmin} \ ||A - L||_F^2 = U\Sigma_r V^T \tag{4}$$

et

$$||A - U\Sigma_r V^T||_F^2 = \sum_{i=k+1}^{n} \sigma_i^2 \tag{5}$$

Démonstration. Preuve donnée dans [3] □

Démonstration du théorème 1. Soit $A \in \mathbb{R}^{m\times n}$. On garde les notations du théorème précédent et on note $\hat{D} = \underset{D}{\text{argmin}} \ ||A - D||_F^2 + \theta^2 rang(D)$.

On suppose qu'il existe k tel que $\sigma_{k+1} < \theta \leq \sigma_k$, montrons alors que $\hat{D}$ est de rang $\leq k$ et le théorème 2 nous donnera le résultat attendu. Soit C une matrice de rang $l > k$. Montrons que

$$||A - \hat{D}||_F^2 + \theta^2 rang(\hat{D}) < ||A - C||_F^2 + \theta^2 l \tag{6}$$

Posons $A_k := UH_\theta(\Sigma)V^T$, comme $\sigma_{k+1} < \theta \leq \sigma_k$, $A_k = U\Sigma_r V^T$. D'après le théorème 2, on a donc :

$$||A - A_k||_F = \sum_{i=k+1}^{n} \sigma_i^2$$

et,

$$\sum_{i=l+1}^{n} \sigma_i^2 \leq ||A - C||_F$$

Donc,

$$\begin{aligned}||A - A_k||_F + \theta^2 k &= \sum_{i=k+1}^{l} \sigma_i^2 + \sum_{i=l+1}^{n} \sigma_i^2 + \theta^2 k \\ &< (l-k)\theta^2 + \sum_{i=l+1}^{n} \sigma_i^2 + \theta^2 k \\ &\leq ||A - C||_F^2 + \theta^2 l.\end{aligned}$$

Enfin comme

$$||A - \hat{D}||_F + \theta^2 rang(\hat{D}) \leq ||A - A_k||_F + \theta^2 k$$

on a bien (6) et donc $rang(\hat{D}) \leq k$ puis d'après [3], $\hat{D} = A_k = UH_\theta(\Sigma)V^T$.
S'il n'existe pas k tel que $\sigma_k \geq \theta$ et $\sigma_{k+1} < \theta$ alors ou bien $\theta > \sigma_1$ ou bien $\theta \leq \sigma_n$ et on va trouver de manière similaire à ce qui précède, respectivement $\hat{D} = 0$ et $\hat{D} = A$, ce qui termine la preuve. □

On remarque que dans l'article [1], on choisit de faire une approximation de rang faible à l'aide de l'équation (3) plutôt que de l'équation (4) ce qui entraîne un temps de calcul plus long. En effet dans le premier cas on va devoir calculer la SVD avec toutes les colonnes de U et V correspondant aux valeurs singulières non nulles. Et dans le deuxième cas on va fixer un rang r pour notre résultat et on peut alors calculer seulement les r premières colonnes de U et V. Cependant le choix de l'article présente un avantage puisque le rang du résultat va s'adapter aux valeurs singulières de A alors que l'autre est fixe.

4.2 Parcimonie

4.2.1 Solution et preuve

Pour la parcimonie aussi on a une solution directe à l'équation (2) :
Théorème 3. *Pour tout $v \in \mathbb{R}^n$,*

$$H_\rho(v) \in \underset{s}{argmin} ||v - s||_2^2 + \rho^2 ||s||_0$$

Démonstration. Comme la norme 0 n'est pas convexe il y a peut y avoir plusieurs solutions à (2). Montrons que $H_\rho(v)$ est une solution. On cherche à minimiser

$$||v - s||_2^2 + \rho^2 ||s||_0 = \sum_{k=1}^{n} (v_k - s_k)^2 + \rho^2 \mathbb{1}_{\mathbb{R}^*}(s_k)$$

ce qui revient à minimiser

$$(v_k - s_k)^2 + \rho^2 \mathbb{1}_{\mathbb{R}^*}(s_k), \forall k \in \{1, .., n\}$$

On note $f(x) = (v_k - x)^2 + \rho^2 \mathbb{1}_{\mathbb{R}^*}(x)$ la fonction à minimiser.
Si $|v_k| > \rho$: alors $(H_\rho(v))_k = v_k$ et $f(v_k) = \rho^2 < f(0) = v_k^2$ et $\forall x \neq 0, f(x) = (v_k - x)^2 + \rho^2 \geq f(v_k) = \rho^2$. Ainsi le minimum de f est bien atteint en $(H_\rho(v))_k$ dans ce cas.
Si $|v_k| \leq \rho$: alors $(H_\rho(v))_k = 0$ et $\forall x \neq 0, f(x) = (v_k - x)^2 + \rho^2 \geq \rho^2 \geq v_k^2 = f(0)$. Ainsi le minimum de f est bien atteint en $(H_\rho(v))_k = 0$ dans ce cas.
Ainsi on obtient bien $H_\rho(v) \in \underset{s}{\text{argmin}} ||v - s||_2^2 + \rho^2 ||s||_0$. □

4.2.2 Transformation

On s'intéresse maintenant au choix de la transformation W, dans l'article W est une matrice unitaire ($WW^T = I$) et va s'adapter à chaque image de la vidéo. En effet pour chaque image la parcimonie va se faire en 3 étapes, on note $Y = (u_i)_{i=1}^N$ avec N le nombre de patch sur l'image, et W_{-1} la transformation de l'image précédente :
1. 1ère parcimonie : $S = H_\rho(W_{-1}Y)$
2. Mise à jour de la transformation : $\hat{W} = \underset{W,\ WW^T=I}{\operatorname{argmin}} ||WY - S||_F$
3. 2ème parcimonie : $\hat{S} = H_\rho(\hat{W}Y)$

Pour la 1ère image on initialise W_{-1} par la transformée en cosinus discrète.
On a ici aussi une solution pour $\hat{W}$ à l'aide d'une SVD.
Théorème 4 (Problème de Procuste orthogonal)**.** *Soit Y, S deux matrices.*
On note $SVD(SY^T) = U\Sigma V^T$ et alors :

$$\underset{W,\ WW^T=I}{argmin} ||WY - S||_F = UV^T \tag{7}$$

Démonstration. Preuve donnée dans [4] □

Pour une raison de coût computationnel on va choisir de faire la parcimonie sur les m (m environ de 5) patchs les plus proches au lieu des K les plus proches. Ainsi on va remplacer dans ce qui précède u_i par $u_i^m = (u_i)_{k=1}^{mn}$ où n est le nombre de pixels que contient un patch.

4.3 Union de la parcimonie et du rang faible

Dans l'article [1] on additionne le résultat de la parcimonie, du rang faible et aussi la matrice de départ avec les formules suivantes, on pose $\hat{U}_i = [\hat{U}_{i,1}|\hat{U}_{i,2}]$ où $\hat{U}_{i,1} \in \mathbb{R}^{n\times m}$ et $\hat{U}_{i,2} \in \mathbb{R}^{n\times(K-m)}$:

$$\hat{U}_{i,1} = \frac{\gamma_s vec^{-1}(C_i(\hat{S})) + C_{1:m}(\gamma_l \hat{D}_i + \gamma_f U_i)}{\gamma_s + \gamma_l + \gamma_f} \tag{8}$$

$$\hat{U}_{i,2} = \frac{C_{m+1:K}(\gamma_l \hat{D}_i + \gamma_f U_i)}{\gamma_l + \gamma_f} \tag{9}$$

où vec^{-1} transforme un vecteur en une matrice colonne par colonne et $C_i(M)$ est la i-ème colonne de M.
J'ai essayé d'autres façons, d'abord de faire la parcimonie non pas sur la matrice de départ U_i mais sur la matrice de rang faible D_i et j'ai ensuite additionné comme avec les formules précédente 1.
Ensuite, un autre algorithme dont le but est d'obtenir une matrice de rang faible et parcimonieuse. J'ai utilisé un algorithme de minimisation alternée [2] pour garantir un rang faible combiné avec la formule de parcimonie 2. L'algorithme de minimisation alternée itère sur la matrice de départ U_i et converge vers une matrice de rang faible (rang fixé, j'ai pris un rang de 6 pour une matrice contenant $K = 15$ patch c'est à dire 15 colonnes) et après chaque itération j'ai ajouté l'étape de parcimonie sur la matrice transformée. Or, j'ai alors observé des taches sur l'image renvoyée par l'algorithme. En exécutant l'étape de parcimonie non pas à chaque itération mais toutes les 3 itérations j'ai obtenu de meilleurs résultats.

Algorithm 1 Parcimonie après rang faible

Require: W, $V = (U_i)_{i=1}^N$
  for $i = 1, 2, ..., N$ **do**
    $SVD(U_i) = \Lambda_i \Sigma_i \Delta_i^T$
    $\hat{D}_i = \Lambda_i H_\theta(\Sigma_i) \Delta_i^T$
  end for
  $\forall i$ $d_i^m = vec(C_{1:m}(D_i))$
  $\hat{S} = H_\rho(WY)$ (avec $Y = (d_i^m)_{i=1}^N$)
  $A, \Sigma, B^T = SVD(\hat{S}Y^T)$
  $W = AB^T$
  $\hat{S} = H_\rho(WY)$
  Reconstruction des U_i avec les formules (8) (9)

Algorithm 2 Minimisation alternée et parcimonie

Require: $W1$, $W2$, $V = (U_i)_{i=1}^N$ et $M1$, $M2$ deux matrices aléatoires
  for $i = 1, 2, ..., N$ **do**
    $P = M1$, $L = M2$
    for $1, 2, ..., 8$ **do**
      $L = P^+ M$ (P^+ pseudo-inverse de P)
      $P = ML^+$
      if $i = 1[3]$ **then**
        $vec(C_{1:m}(P)) = H_\rho(W1 vec(C_{1:m}(P)))$
        Mise à jour de W1
        $vec(C_{1:m}(P)) = H_\rho(W1 vec(C_{1:m}(P)))$
        $vec(C_{1:m}(L)) = H_\rho(W2 vec(C_{1:m}(L)))$
        Mise à jour de W2
        $vec(C_{1:m}(L)) = H_\rho(W2 vec(C_{1:m}(L)))$
      end if
    end for
    $U_i = PL$
  end for

5 Résultats

5.1 Paramètres

J'ai testé mon code [1] sur plusieurs vidéos (en niveaux de gris) en ajoutant un bruit gaussien d'écart type $\sigma = 5, 10, 15$ ou 20. Pour établir les paramètres de base de mon code je me suis inspiré des paramètres données dans [1] et je les ai adaptés en fonction de ce que j'obtenais. Globalement, j'ai utilisé des paramètres moins coûteux que ceux de l'article. J'ai pris un pas $p = 2$ pour le saut entre les patch ($p = 1$ pour [1]), des patch carrés de taille $n1 \times n2 = 8 \times 8$, une fenêtre de recherche temporel de 3 images avant 3 images après, une fenêtre de recherche spatiale de 10×10. Un nombre de patch similaires $K = 15$, des poids $\gamma_s = \gamma_l = 1$, $\gamma_f = 0.1$, $\rho = 7\sqrt{\sigma}$, $\theta = 0.5\sigma(\sqrt{K} + n1)$. Et pour les bords temporels et spatiaux on prend une fenêtre de recherche moins grande et adaptée, par exemple pour la 1ère on cherchera seulement dans les 3 images suivantes. J'ai testé mes algorithmes principalement sur des vidéos de faibles qualités d'une vingtaine d'images pour avoir un temps de calcul plutot rapide, l'algorithme étant coûteux et mon code peu optimisé. Je vais afficher des images tirées de ces vidéos pour illustrer mes résultats.

5.2 Comparaison des algorithmes et des paramètres

Tout d'abord pour les paramètres avec $\sigma = 15$ ci-dessus j'obtiens avec mon implémentation de l'algorithme SALT proposé par [1] la figure 1.

J'ai testé l'algorithme en choisissant différent pas p pour les patch, j'ai pris $p = 1$, $p = 2$ et $p = 4$ (figure 2). Les résultats pour les pas $p = 1$ et $p = 2$ semblent très proches alors que pour $p = 4$ le résultat semble moins bon, j'ai donc choisi de garder un pas $p = 2$ pour le reste des tests.

J'ai ensuite comparé l'algorithme parcimonie plus rang faible avec un algorithme comprenant seulement la parcimonie et un comprenant seulement le rang faible, j'ai constaté que les résultats du premier algorithme sont meilleurs que les deux autres et le rang faible un peu meilleur que la parcimonie. On a par exemple sur la figure 3 une illustration des trois (avec un bruit d'écart type $\sigma = 15$).

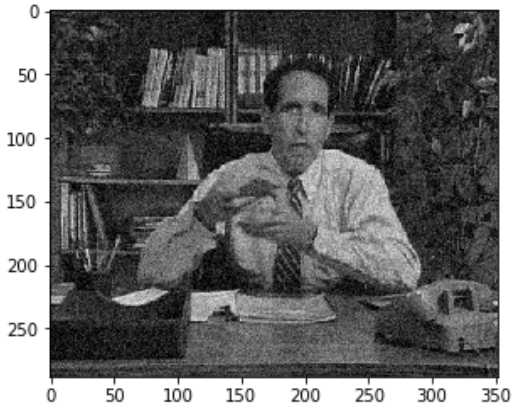
Et enfin, en essayant de combiner parcimonie et rang faible en même temps à l'aide des algorithmes 1 et 2, je n'ai pas réussi à obtenir de meilleurs résultats qu'avec les 3 algorithmes précédents (de la figure 3). J'obtiens par exemple avec $\sigma = 15$ la figure 4.

5.3 Vidéo

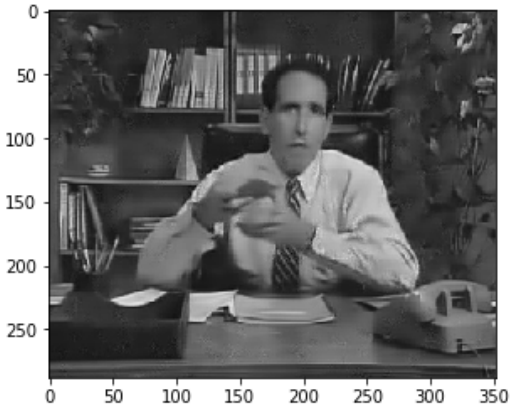
Et pour avoir une idée de ce que l'algorithme peut donner concrètement, je mets en lien une vidéo débruité à l'aide de mon code Python de l'algorithme [1] (parcimonie + rang faible) en comparaison avec la vidéo bruité d'écart type $\sigma = 15$: https://1drv.ms/u/s!AmxflszZlT6PhxD2UfKtaR7xq4Lz?e=7CIQvq

1. code sur : https ://github.com/pgicquel/video-denoising

(a) Image d'origine

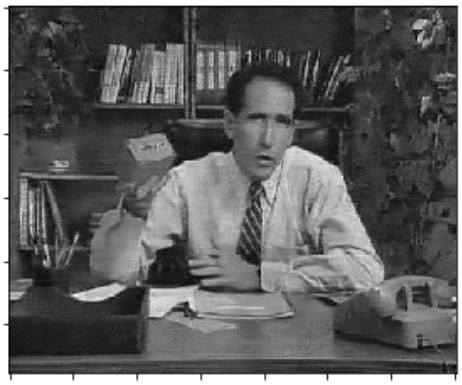


(b) Bruit gaussien $\sigma = 15$

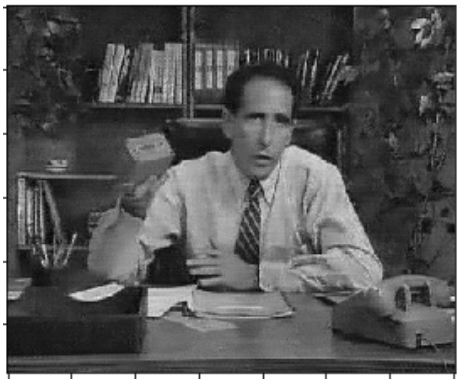


(c) Débruité par SALT

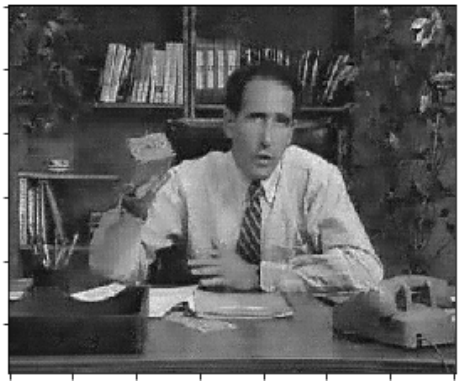
FIGURE 1 – Algorithme SALT



(a) $p = 1$

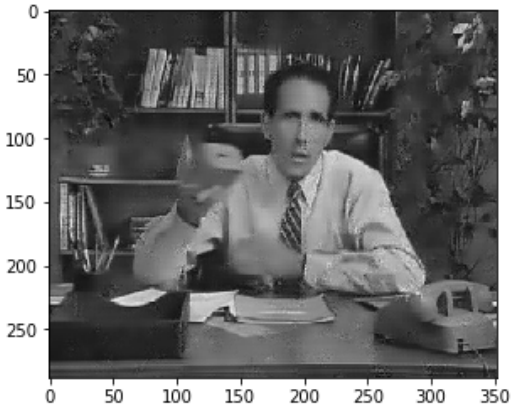


(b) $p = 2$

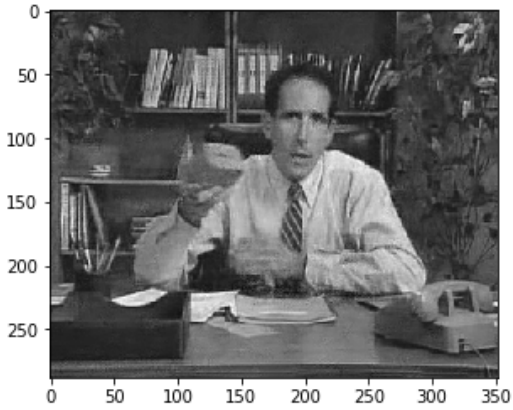


(c) $p = 4$

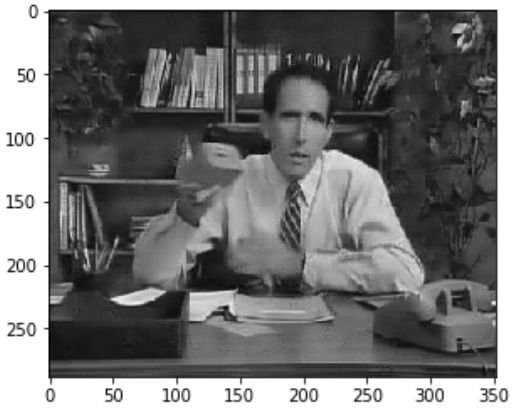
FIGURE 2 – Comparaison pas



(a) Rang faible

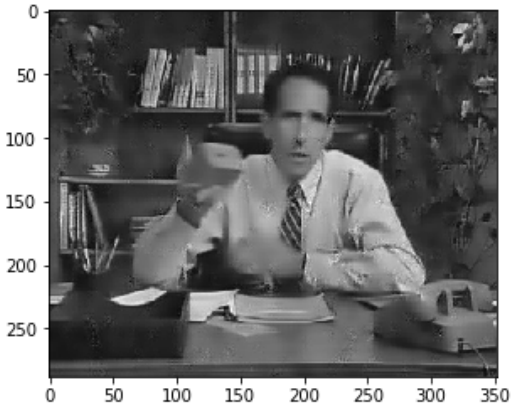


(b) Parcimonie

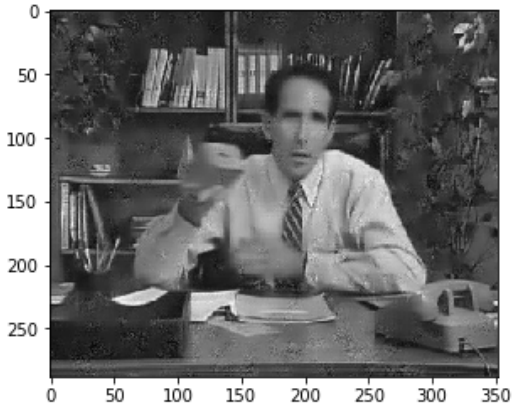


(c) Rang faible + parcimonie

FIGURE 3 – Rang faible et/ou parcimonie



(a) Rang faible puis parcimonie



(b) Minimisation alternée

FIGURE 4 – Parcimonie avec rang faible

Références

[1] Wen Bihan et al. "Joint Adaptive Sparsity and Low-Rankness on the Fly :An Online Tensor Reconstruction Scheme for Video Denoising". In : (2017).

[2] Zhao Tuo, Wang Zhaoran et Liu Han. "A Nonconvex Optimization Framework for Low Rank Matrix Estimation". In : (2015).

[3] Wikipedia. "Low-rank approximation". In : (). Proof of Eckart–Young–Mirsky theorem (for Frobenius norm).

[4] Wikipedia. "Orthogonal Procrustes theorem". In : (2019).