

Benoit AUNE  
Université de Bordeaux  
Année 2018-2019



## RAPPORT DE STAGE

MATHÉMATIQUES APPLIQUÉES AU TRAITEMENT D'IMAGES

---

# La super résolution d'images à base de patches

---



*Encadrants :* Jean-François AUJOL  
Yannick BERTHOUMIEU  
Dominique BERNARD



# **Remerciements**

Je tiens à remercier M. Aujol, M. Berthoumieu et M. Bernard de m'avoir accepté pour ce stage et de m'avoir guidé dans son avancement.

Je désire également exprimer ma reconnaissance à tous ceux qui m'ont soutenu et encouragé durant ce stage et durant mon Master.

# Introduction générale

Le travail de ce rapport s'intéresse à la reconstruction d'une image en Haute Résolution (HR) à partir d'une images en basse résolution (LR). Il s'agit donc d'augmenter la résolution spatiale (nombre de pixels de l'image) en améliorant au maximum le niveau de détail de l'image. On peut trouver beaucoup d'applications civiles ou militaires telles que la détection ou la reconnaissance. Mais aussi dans d'autres domaines, tels que la santé, l'astronomie, l'audiovisuel ou la science des matériaux. Par exemple, les techniques d'acquisitions à grandes résolutions peuvent être limitées par le coût et la taille du capteur de haute définition. Les développements récents en imagerie ont énormément modifié les approches en sciences des matériaux. Les enjeux de l'étude de ces matériaux peuvent être importants. C'est pourquoi, on cherche à comparer certaines approches de Super-Résolution (SR).

On va s'intéresser à des approches incrémentales mais il n'est pas possible de considérer ce problème dans toute sa complexité.

Il existe de très nombreuses applications concrètes de la super-résolution. La littérature sur le sujet est très vaste et on pourra se référer à [7] pour une étude complète des méthodes développées jusqu'en 2014.

En utilisant un modèle de mélange gaussien global, Yu et al. [3] ont proposé une approche bayesienne basée patch et Sandeep a étendu cette modélisation pour de la modélisation HR-LR jointe appliquée au problème de super résolution [10]. Pour le débruitage on peut aussi se référer à [4] et à [5] pour du compressing sensing.

Il est possible de considérer des modèles de densité plus flexibles pour s'adapter à la forme de la donnée comme dans les articles [1], [2], [9] et [11].

Il est aussi possible de considérer des modèles plus simple locaux comme le fait Niknejad dans son article [6] pour la restauration. Certains de ces articles peuvent être étendus à la super résolution et le but de ce rapport est de les comparer.

## Table des matières

<b>1 Choix du modèle</b>	<b>5</b>
1.1 Loi normale multidimensionnelle	5
1.2 Gaussian Mixture Model (GMM)	6
1.3 Generalized Gaussian Mixture Model (GGMM) à moyenne nulle	6
<b>2 Estimation des paramètres de <math>p</math></b>	<b>7</b>
2.1 Estimateur du maximum de vraisemblance (ML)	7
2.2 L'algorithme Expectation-Maximisation (EM)	7
<b>3 Estimation des patchs</b>	<b>9</b>
3.1 Minimum mean square error (MMSE)	9
3.1.1 MMSE utilisant une composante d'un modèle de mélange	10
3.1.2 MMSE utilisant toutes les composantes d'un GMM	10
3.2 Maximum A Posteriori (MAP)	11
3.2.1 Expected Patch Log-Likelihood (EPLL) de Zoran et Weiss [12]	12
3.2.2 Fast EPLL (FEPLL) de Deledalle pour le GMM à moyenne nulle [8]	14
3.3 Discussion	17
<b>4 Reconstruction de l'image</b>	<b>17</b>
4.1 Pixels centraux	17
4.2 Moyenne pondérée sur les patchs	18
<b>5 Article 1 : Méthode de Sandeep et Jacob pour la Super-Résolution [10]</b>	<b>18</b>
5.1 Introduction de la méthode	18
5.2 Estimation des paramètres par la méthode GMM Learning	19
5.3 Restauration des patchs	20
5.4 Reconstruction de l'image $\hat{X}$	20
5.5 Résultats Article 1	21
5.5.1 Résultats en utilisant le MMSE avec une seule composante (sous-sous-section 3.1.1) et l'EPLL (sous-sous-section 3.2.1)	21
5.5.2 Résultats en utilisant le MMSE avec toutes les composantes (sous-sous-section 3.1.2)	31
<b>6 Article 2 : Méthode de Niknejad, Rabbani et Babaie-Zadeh pour la Super-Résolution [6]</b>	<b>35</b>
6.1 Introduction de la méthode	35
6.2 L'algorithme LINC	35
6.2.1 Regroupement par la méthode kNN (k plus proche voisin)	35
6.2.2 Estimation des paramètres	36
6.2.3 Restauration des patchs	36
6.3 Reconstruction de l'image $\hat{X}$	36
6.4 Résultats Article 2	37
6.4.1 Débruitage	37
6.4.2 Inpainting	39
6.4.3 Super Résolution	39
<b>7 Discussion</b>	<b>43</b>
<b>8 Autres résultats</b>	<b>43</b>
8.1 Comparaison entre les GMM, LMM et GGMM en utilisant l'EPLL	43
8.2 Comparaison entre l'EPLL et le FEPLL	44
<b>9 Conclusion</b>	<b>45</b>

<b>A Annexe MMSE</b>	<b>46</b>
<b>B Annexe Approximations</b>	<b>47</b>

# Introduction

On note  $p(v; \Theta)$  la densité de probabilité d'un patch  $v$  de paramètre  $\Theta$ . On se limite au cas de modèles linéaires de type :

$$y = Hx + b \quad (1)$$

avec  $b \in \mathbb{R}^F$  un bruit blanc gaussien d'une variance  $\sigma^2$ ,  $H : \mathbb{R}^E \longrightarrow \mathbb{R}^F$  un opérateur linéaire et  $x \in \mathbb{R}^E, y \in \mathbb{R}^F$  sont les représentations vectorielles des images  $X, Y$  avec  $E$  et  $F$  leur nombre de pixels respectivement.

On cherche toujours à estimer une image  $\hat{X}$  à partir de  $Y$  observée. Parfois on connaît une partie de  $X$ .

Dans ce rapport, nous allons présenter les différentes méthodes à patchs en traitements d'image. Ces méthodes se découpent en quatre parties après l'extraction des patchs de l'image. La section 1 montre les différents choix des densités de probabilités  $p$  que l'on peut utiliser. La section 2 explique les différentes méthodes d'estimation de paramètres de  $p$ . Ensuite, la section 3 expose les différentes manières d'estimer nos nouveaux patchs selon les paramètres de  $p$  calculés. Enfin, la section 4 indique les solutions pour reconstruire l'image grâce aux patchs reconstruits.

Les deux sections suivantes sont des présentations d'articles réutilisant les méthodes citées précédemment ainsi que leurs résultats.

## Extraction des patchs :

On notera pour le reste du rapport  $\{x_i\}_{i=1}^N$  et  $\{y_l\}_{l=1}^M$  les ensembles de  $N$  et  $M$  patchs sur  $X$  et  $Y$  respectivement avec  $\forall i x_i \in \mathbb{R}^{\tau_N}$  et  $\forall l y_l \in \mathbb{R}^{\tau_M}$ .  $\tau_N$  et  $\tau_M$  sont donc les tailles des patchs sur  $X$  et  $Y$  respectivement que l'on définira à l'avance et elles peuvent être égales (cela dépend du problème posé).

## 1 Choix du modèle

Dans la suite du rapport, on va utiliser plusieurs densités de probabilité à priori sur nos patchs :

### 1.1 Loi normale multidimensionnelle

$$p(v) = p(v; \Theta) = \mathcal{N}(v|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{\tau}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(v-\mu)^T \Sigma^{-1} (v-\mu)} \quad (2)$$

modélisant une gaussienne en fonction de nos patchs  $v \in \mathbb{R}^\tau$  et de deux paramètres :  $\mu \in \mathbb{R}^\tau$  la moyenne et  $\Sigma^{\tau \times \tau}$  la covariance.  $\Theta = (\mu, \Sigma)$  caractérise la loi. Cette loi est utilisée dans l'article [6] (voir section 6) en usant d'un modèle local (traitant les régions de l'image indépendamment des autres).

## 1.2 Gaussian Mixture Model (GMM)

$$p(v) = p(v; \Theta) = \sum_{k=1}^K \omega_k \mathcal{N}(v | \mu_k, \Sigma_k) \quad (3)$$

modélisant  $K$  gaussiennes en fonction de nos patchs  $v \in \mathbb{R}^\tau$  et de trois paramètres  $\forall k = 1 \dots K$  : les poids  $\omega_k$  tels que  $\sum_{k=1}^K \omega_k = 1$ ,  $\mu_k \in \mathbb{R}^\tau$ , les moyennes et  $\Sigma_k^{\tau \times \tau}$  les covariances.  $\Theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$  caractérise donc complètement le GMM. Les modèles de mélanges sont des modèles globaux (traitant l'image entière). Le GMM est utilisé dans les articles [5], [8], [10] (voir section 5) et [12] pour nos résultats.

## 1.3 Generalized Gaussian Mixture Model (GGMM) à moyenne nulle

On définit la loi généralisant la loi normale multidimensionnelle :

$$\mathcal{G}(z, 0, \Sigma, \nu) = \frac{\Phi}{2|\Sigma_\nu|^{1/2}} \exp(-||\Sigma_\nu^{-1/2} z||_\nu^\nu) \quad (4)$$

avec

$$||u||_\nu^\nu = \sum_{j=1}^\tau |u_j|^{\nu_j}$$

où

$$\Phi = \prod_{j=1}^\tau \frac{\nu_j}{\Gamma(1/\nu_j)} \text{ et } \Sigma_\nu^{1/2} = \Sigma^{1/2} \begin{pmatrix} \sqrt{\frac{\Gamma(1/\nu_1)}{\Gamma(3/\nu_1)}} & & \\ & \ddots & \\ & & \sqrt{\frac{\Gamma(1/\nu_P)}{\Gamma(3/\nu_P)}} \end{pmatrix}$$

Le modèle de mélange devient :

$$p(z) = p(z; \Theta) = \sum_{k=1}^K \omega_k \mathcal{G}(z, 0, \Sigma_k, \nu_k) \quad (5)$$

avec  $z \in \mathbb{R}^\tau$  nos patchs,  $K$  le nombre de composantes,  $\forall k = 1 \dots K$  on a :  $\Sigma_k \in \mathbb{R}^{\tau \times \tau}$  les covariances,  $\omega_k > 0$  les poids tels que  $\sum_{k=1}^K \omega_k = 1$  et  $\nu_k \in \mathbb{R}^\tau$  les paramètres de forme.

$\Theta = \{\omega_k, \Sigma_k, \nu_k\}_{k=1}^K$  caractérise donc complètement le GGMM à moyenne nulle. Le GGMM est utilisé dans l'article [9].

**1.3.1 Proposition.** : Si on utilise la décomposition en valeurs propres, on a  $\Sigma = UDU^t$  avec  $D = \text{diag}(\lambda_1, \dots, \lambda_\tau)^2$  et  $U \in \mathbb{R}^{\tau \times \tau}$  unitaire, c'est à dire :  $U^t = U^{-1}$ . De ce fait on a que  $\Sigma^{\frac{1}{2}} = U D^{\frac{1}{2}}$ ,  $\Sigma^{-\frac{1}{2}} = D^{-\frac{1}{2}} U^t$  et :

$$\mathcal{G}(z, 0, \Sigma, \nu) = \prod_{j=1}^\tau \mathcal{G}((U^t z)_j, 0, \lambda_j, \nu_j) \quad (6)$$

avec

$$\mathcal{G}(u, 0, \lambda, \nu) = \frac{\Phi}{2\lambda_\nu} \exp\left(-\left(\frac{|u|}{\lambda_\nu}\right)^\nu\right)$$

où

$$\Phi = \frac{\nu}{\Gamma(1, \nu)} \text{ et } \lambda_\nu = \lambda \sqrt{\frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}}.$$

## 2 Estimation des paramètres de $p$

### 2.1 Estimateur du maximum de vraisemblance (ML)

On appelle vraisemblance de  $\Theta$  au vu des observations (dans notre cas des patchs de taille  $\tau$ )  $(v_1, \dots, v_i, \dots, v_n; \Theta)$  d'un n-échantillon indépendamment et identiquement distribué selon la loi  $p$ , le nombre :

$$L(v_1, \dots, v_i, \dots, v_n; \Theta) = p(v_1; \Theta) \times p(v_2; \Theta) \times \dots \times p(v_n; \Theta) = \prod_{i=1}^n p(v_i; \Theta) \quad (7)$$

On peut aussi calculer la - log-vraisemblance :

$$-\log L(v_1, \dots, v_i, \dots, v_n; \Theta) = -\sum_{i=1}^n \log p(v_i; \Theta) \quad (8)$$

Et le but est de trouver le paramètre  $\Theta$  qui maximise cette vraisemblance sur nos données :

$$\begin{aligned} \hat{\Theta} &= \operatorname{argmax}_{\Theta} L(v_1, \dots, v_i, \dots, v_n; \Theta) \\ &= \operatorname{argmax}_{\Theta} \prod_{i=1}^n p(v_i; \Theta) \\ &= \operatorname{argmin}_{\Theta} -\sum_{i=1}^n \log p(v_i; \Theta) \end{aligned} \quad (9)$$

Selon  $p$ , le problème peut être non convexe. Dans ce cas, l'algorithme EM permet de se rapprocher de la solution.

### 2.2 L'algorithme Expectation-Maximisation (EM)

L'algorithme consiste à initialiser  $\Theta$  le paramètre de la densité de probabilité puis à alterner entre deux étapes pour estimer une nouvelle fois au mieux le paramètre  $\Theta$  sachant nos données (ici des patchs appartenant à  $\mathbb{R}^\tau$ ) :

- L'étape de prévision (E-step) qui calcule la distribution conditionnelle déterminée par le théorème de Bayes selon chaque composante  $\forall k = 1 \dots K$  dont le paramètre  $\Theta$  est fixé.

- L'étape de Maximisation (M-step) qui ajuste le paramètre  $\Theta$  en utilisant les résultats calculés dans l'étape de prévision.

On recalcule la vraisemblance avec  $\Theta$  fixé. Si on atteint un certain seuil de convergence  $\delta$  pour la vraisemblance on retourne le paramètre  $\Theta$ , sinon on recommence à l'étape de prévision.

#### EM avec le GMM :

Dans ce cas avec  $\Theta$  fixé, le calcul de la log-vraisemblance de nos données devient :

$$L = \frac{1}{n} \sum_{i=1}^n \log\left(\sum_{k=1}^K \omega_k \mathcal{N}(v_i, \mu_k, \Sigma_k)\right) \quad (10)$$

### Expectation step (E-Step)

Pour toutes les gaussiennes ( $\forall k = 1 \dots K$ ) et pour tous les échantillons connus de l'image ( $\forall i = 1 \dots n$ ) on calcule :

$$\gamma_{k,i} = \frac{\omega_k \mathcal{N}(v_i, \mu_k, \Sigma_k)}{\sum_{l=1}^K \omega_l \mathcal{N}(v_i, \mu_l, \Sigma_l)} \quad (11)$$

et  $\forall k = 1 \dots K$  :

$$\eta_k = \sum_{i=1}^n \gamma_{k,i} \quad (12)$$

### Maximization step (M-Step)

Pour toutes les gaussiennes ( $\forall k = 1 \dots K$ ), on calcule :

$$\omega_k = \frac{\eta_k}{n} \quad (13)$$

$$\mu_k = \frac{1}{\eta_k} \sum_{i=1}^n \gamma_{k,i} v_i \quad (14)$$

et

$$\Sigma_k = \frac{1}{\eta_k} \sum_{i=1}^n \gamma_{k,i} (v_i - \mu_k)(v_i - \mu_k)^t \quad (15)$$

On peut appliquer une régularisation sur la covariance pour s'assurer de son inversibilité  $\forall k = 1 \dots K$  :

$$\Sigma_k = \Sigma_k + \epsilon I \quad (16)$$

où  $\epsilon$  est une petite constante à définir à l'avance.

### EM avec le GGMM :

Dans ce cas avec  $\Theta$  fixé, le calcul de la log-vraisemblance de nos données devient :

$$L = \frac{1}{n} \sum_{i=1}^n \log\left(\sum_{k=1}^K \omega_k \mathcal{G}(z_i, \mu_k, \Sigma_k)\right) \quad (17)$$

### Expectation step (E-Step)

Pour toutes les gaussiennes ( $\forall k = 1 \dots K$ ) et pour tous les échantillons connues de l'image ( $\forall i = 1 \dots n$ ) on calcule :

$$\varepsilon_{k,i} = \frac{\omega_k \mathcal{G}(z_i, 0, \Sigma_k, \nu_k)}{\sum_{l=1}^K \omega_l \mathcal{G}(z_i, 0, \Sigma_l, \nu_l)} \quad (18)$$

### Maximization step (M-Step)

Pour toutes les gaussiennes ( $\forall k = 1 \dots K$ ), on calcule :

$$\omega_k = \frac{\sum_{i=1}^n \varepsilon_{k,i}}{\sum_{l=1}^K \sum_{i=1}^n \varepsilon_{l,i}} \quad (19)$$

et

$$\Sigma_k = \frac{\sum_{i=1}^n \varepsilon_{k,i} z_i z_i^t}{\sum_{i=1}^n \varepsilon_{k,i}}. \quad (20)$$

On effectue une décomposition en valeurs propres de  $\Sigma_k$  :

$$\Sigma_k = U_k D_k U_k^t \quad (21)$$

où

$$D_k = \text{diag}(\lambda_{k,1}, \dots, \lambda_{k,\tau})^2. \quad (22)$$

Pour toutes les gaussiennes ( $\forall k = 1 \dots K$ ) et pour tous les pixels du patch ( $\forall j = 1 \dots \tau$ ), on calcule :

$$\chi_{k,j} = \frac{\sum_{i=1}^n \varepsilon_{k,i} |(U_k^t z_i)_j|}{\sum_{i=1}^n \varepsilon_{k,i}} \quad (23)$$

et

$$(\nu_k)_j = \min(\max(X_{k,j}, 3), 2). \quad (24)$$

avec

$$X_{k,j} = F^{-1}\left(\frac{\chi_{k,j}^2}{\lambda_{k,j}^2}\right) \quad (25)$$

et

$$F(x) = \frac{\Gamma(2/x)^2}{\Gamma(3/x)\Gamma(1/x)}. \quad (26)$$

## 3 Estimation des patchs

### 3.1 Minimum mean square error (MMSE)

Supposons  $\hat{x}(y)$  l'estimation du vecteur  $x \in \mathbb{R}^E$  grâce au vecteur  $y \in \mathbb{R}^F$ .

Le calcul du MSE (Mean Square Error) consiste à faire :

$$MSE(\hat{x}(y)|y) = \mathbb{E}\{(x - \hat{x}(y))^2|y\}. \quad (27)$$

Le MMSE consiste donc à calculer (voir Annexe MMSE) :

$$\begin{aligned} \hat{x}_{MMSE}(y) &= \underset{\hat{x}(y)}{\operatorname{argmin}} MSE(\hat{x}(y)|y) \\ &= \mathbb{E}[x|y] \end{aligned} \quad (28)$$

L'application sur nos patchs se fait de la même manière avec  $\forall i = 1 \dots N$   $x_i \in \mathbb{R}^{\tau_N}$  et  $\forall l = 1 \dots M$   $y_l \in \mathbb{R}^{\tau_M}$  et dans certains cas le calcul peut encore être simplifié.

### 3.1.1 MMSE utilisant une composante d'un modèle de mélange

L'estimation du vecteur  $x$  sachant  $y$  devient donc le calcul de  $\mathbb{E}[x|y]$ . On choisit donc la meilleure composante pour calculer cette espérance. Pour cela on utilise la vraisemblance.

Soit  $\Theta_{\hat{k}}$  l'ensemble des paramètres les plus vraisemblables selon nos données. On a pour le GMM,  $\Theta_{\hat{k}} = \{\omega_{\hat{k}}, \mu_{\hat{k}}, \Sigma_{\hat{k}}\}$  avec  $\mu_{\hat{k}} = \begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}$  et  $\Sigma_{\hat{k}} = \begin{pmatrix} \Sigma_X & \Sigma_{X,Y} \\ \Sigma_{X,Y}^T & \Sigma_Y \end{pmatrix}$ .

Selon l'Annexe MMSE, on calcule :

$$\hat{x} = \mathbb{E}[x|y] \quad (29)$$

$$= \mu_X + \Sigma_{X,Y} \Sigma_Y^{-1} (y - \mu_Y) \quad (30)$$

### 3.1.2 MMSE utilisant toutes les composantes d'un GMM

Soient  $x \in \mathbb{R}^E$  et  $y \in \mathbb{R}^F$  des vecteurs et un GMM à  $K$  composantes, on a la probabilité de densité :

$$p(x) = \sum_{k=1}^K \omega_k \mathcal{N}(x; \mu_k, \Sigma_k)$$

avec  $\mu_k$  les moyennes et  $\Sigma_k$  les covariances. On rappelle que notre problème de départ est  $y = Hx + b$  avec  $b$  un bruit blanc gaussien de variance  $\sigma^2$  et  $H$  un opérateur linéaire. On note  $\varsigma$  la matrice diagonale contenant la variance de  $b$ . On a  $p(y|x) = \mathcal{N}(y|Hx, \varsigma)$ . En prenant un  $k$  fixe on obtient :

$$\begin{aligned} p(y, x; \mu_k, \Sigma_k) &= p(y|x)p(x; \mu_k, \Sigma_k) \\ &= \mathcal{N}(y|Hx, \varsigma) \mathcal{N}(x; \mu_k, \Sigma_k) \end{aligned} \quad (31)$$

Après un arrangement des résultats on obtient :

$$\mathcal{N} \left( \begin{pmatrix} y \\ x \end{pmatrix} \middle| \begin{pmatrix} H\mu_k \\ \mu_k \end{pmatrix}, \begin{pmatrix} \varsigma^{-1} & -\varsigma^{-1}H \\ -H'\varsigma^{-1} & H'\varsigma^{-1}H + \Sigma_k^{-1} \end{pmatrix}^{-1} \right)$$

On peut aussi écrire :

$$p(x, y; \mu_k, \Sigma_k, \varsigma) = \mathcal{N}(x; \eta_k(y; \mu_k, \Sigma_k, \varsigma), C_k(y; \mu_k, \Sigma_k, \varsigma))$$

et

$$p(y; \mu_k, \Sigma_k, \varsigma) = \mathcal{N}(y; H\mu_k, R_k(\mu_k, \Sigma_k, \varsigma))$$

où

$$\begin{aligned} \eta_k(y; \mu_k, \Sigma_k, \varsigma) &= \mu_k + (H'\varsigma^{-1}H + \Sigma_k^{-1})^{-1}H'\varsigma^{-1}(y - H\mu_k), \\ C_k(y; \mu_k, \Sigma_k, \varsigma) &= (H'\varsigma^{-1}H + \Sigma_k^{-1})^{-1}, \\ R_k(\mu_k, \Sigma_k, \varsigma) &= (\varsigma^{-1} - \varsigma^{-1}H(H'\varsigma^{-1}H + \Sigma_k^{-1})^{-1}H'\varsigma^{-1})^{-1}. \end{aligned} \quad (32)$$

Soit  $\Theta = \{\varsigma\} \cup \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$ , on peut écrire :

$$\begin{aligned} p(k, x, y; \Theta) &= p(k)p(x|y; k)p(y; k) \\ &= \omega_k p(x|y; \mu_k, \Sigma_k, \varsigma)p(y; \mu_k, \Sigma_k, \varsigma) \\ &= \omega_k \mathcal{N}(x; \eta_k(y; \Theta), C_k(y; \Theta)) \mathcal{N}(y; H\mu_k, R_k(\Theta)) \end{aligned} \quad (33)$$

On peut encore simplifier l'équation (32) :

$$\begin{aligned} \eta_k(y; \mu_k, \Sigma_k, \varsigma) &= \mu_k + \Sigma_k H' (\varsigma + H \Sigma_k H')^{-1} (y - H \mu_k), \\ C_k(y; \mu_k, \Sigma_k, \varsigma) &= \Sigma_k - \Sigma_k H' (\varsigma + H \Sigma_k H')^{-1} H \Sigma_k, \\ R_k(\mu_k, \Sigma_k, \varsigma) &= \varsigma + H \Sigma_k H' \end{aligned} \quad (34)$$

La distribution marginale de  $y$  devient donc :

$$p(y; \Theta) = \sum_{k=1}^K \omega_k \mathcal{N}(y; H\mu_k, R_k(\Theta)) \quad (35)$$

et la distribution à posteriori de  $(k, x)$  sachant  $y$  est donc :

$$\begin{aligned} p(k, x|y; \Theta) &= p(k|y)p(x|y, k) \\ &= \rho_k(y; \Theta) \mathcal{N}(x; \eta_k(y; \Theta), C_k(y; \Theta)) \end{aligned} \quad (36)$$

où

$$\rho_k(y; \Theta) = \frac{\omega_k \mathcal{N}(y; H\mu_k, R_k(\Theta))}{\sum_{l=1}^K \omega_l \mathcal{N}(y; H\mu_l, R_l(\Theta))} \quad (37)$$

De plus, grâce à l'équation (36), on continue en donnant la distribution à posteriori de  $x$  :

$$p(x|y; \Theta) = \sum_{k=1}^K \rho_k(y; \Theta) \mathcal{N}(x; \eta_k(y; \Theta), C_k(y; \Theta)) \quad (38)$$

Maintenant que nous avons nos densités de probabilité, on peut reprendre le calcul du MMSE, on obtient :

$$\begin{aligned} \hat{x}_{MMSE}(y; \Theta) &= \mathbb{E}[x|y; \Theta] \\ &= \sum_{k=1}^K \rho_k(y; \Theta) \int x \mathcal{N}(x; \eta_k(y; \Theta), C_k(y; \Theta)) dx \\ &= \sum_{k=1}^K \rho_k(y; \Theta) \eta_k(y; \Theta) \end{aligned} \quad (39)$$

### 3.2 Maximum A Posteriori (MAP)

On suppose que selon la loi  $p$ , le paramètre  $\Theta$  est fixé. Calculer le MAP pour estimer nos patchs consiste à calculer :

$$\begin{aligned} \hat{x}_i &= \operatorname{argmax}_x p(x|y_i; \Theta) \\ &= \operatorname{argmax}_x p(y_i|x; \Theta)p(x; \Theta) \\ &= \operatorname{argmin}_x -\log(p(y_i|x; \Theta)p(x; \Theta)) \end{aligned} \quad (40)$$

La deuxième ligne est obtenue grâce à la formule de Bayes. Selon  $p$ , le problème peut être non convexe. Dans ce cas, l'algorithme EPLL permet de se rapprocher de la solution.

### 3.2.1 Expected Patch Log-Likelihood (EPLL) de Zoran et Weiss [12]

On rappelle que l'on travaille sur  $y = Hx + b$

avec  $b \in \mathbb{R}^F$  un bruit blanc gaussien d'une variance  $\sigma^2$ ,  $H : \mathbb{R}^E \rightarrow \mathbb{R}^F$  un opérateur linéaire et  $x \in \mathbb{R}^E, y \in \mathbb{R}^F$  sont les représentations vectorielles des images  $X, Y$  avec  $E$  et  $F$  leur nombre de pixels respectivement.

$\{x_i\}_{i=1}^N$  et  $\{y_l\}_{l=1}^M$  sont les  $N$  et  $M$  patchs associés à  $X$  et  $Y$  respectivement avec  $\forall i x_i \in \mathbb{R}^{\tau_N}$  et  $\forall l y_l \in \mathbb{R}^{\tau_M}$ .  $\tau_N$  et  $\tau_M$  sont donc les tailles des patchs de  $X$  et  $Y$  respectivement.

On cherche toujours à estimer une image  $\hat{X}$  à partir de  $Y$  observée.

L'algorithme EPLL permet d'estimer les patchs d'une image par l'estimation du Maximum à Posteriori (MAP), cela correspond au problème de minimisation :

$$\underset{x \in \mathbb{R}^E}{\operatorname{argmin}} \frac{\tau_N}{2\sigma^2} \|Hx - y\|^2 - \sum_{i=1}^N \log p(\mathcal{P}_i x) \quad (41)$$

avec  $\mathcal{P}_i : \mathbb{R}^N \rightarrow \mathbb{R}^{\tau_N}$  l'opérateur linéaire qui extrait le patch de l'image centré au pixel  $i$ ,  $\tau_N$  la taille des patchs.

Dans beaucoup de cas,  $p(x)$  est pris non convexe. Le problème devient donc difficile à calculer. L'EPLL consiste à utiliser la technique **Optimization with half-quadratic splitting**.

Elle considère un problème de pénalisation en introduisant  $N$  vecteurs inconnus auxiliaires  $z_i \in \mathbb{R}^{\tau_N}$  :

$$\underset{\substack{x \in \mathbb{R}^E \\ z_1, \dots, z_N \in \mathbb{R}^{\tau_N}}}{\operatorname{argmin}} \frac{\tau_N}{2\sigma^2} \|Hx - y\|^2 + \frac{\beta}{2} \sum_{i=1}^N \|\mathcal{P}_i x - z_i\|^2 - \sum_{i=1}^N \log p(z_i). \quad (42)$$

Lorsque  $\beta \rightarrow \infty$  le nouveau problème est équivalent à l'original. En pratique, on augmente  $\beta$  à chaque itération de l'algorithme. On peut l'initialiser par  $\hat{x} = y$  et l'optimisation se fait donc en alternant les minimisations de  $\hat{x}$  et de  $\hat{z}_i \forall i$  :

$$\hat{x} = \underset{x \in \mathbb{R}^E}{\operatorname{argmin}} \frac{\tau_N}{2\sigma^2} \|Hx - y\|^2 + \frac{\beta}{2} \sum_{i=1}^N \|\mathcal{P}_i x - \hat{z}_i\|^2 \quad (43)$$

$$= (H^t H + \frac{\beta \sigma^2}{\tau_N} \sum_{i=1}^N \mathcal{P}_i^t \mathcal{P}_i)^{-1} (H^t y + \frac{\beta \sigma^2}{\tau_N} \sum_{i=1}^N \mathcal{P}_i^t \hat{z}_i) \quad (44)$$

avec les  $\hat{z}_i$  fixés et  $\mathcal{P}_i^t \mathcal{P}_i$  la matrice diagonale dont les éléments correspondent au nombre de patchs qui se chevauchent à la position  $i$ .

$$\hat{z}_i = \underset{z_i \in \mathbb{R}^{\tau_N}}{\operatorname{argmin}} \frac{\beta}{2} \|\tilde{z}_i - z_i\|^2 - \log p(z_i) \quad (45)$$

avec  $\hat{x}$  fixé et  $\tilde{z}_i = \mathcal{P}_i \hat{x}$ . La convexité et la difficulté de résolution de ce problème dépend beaucoup du choix de  $p$  à priori et le problème correspond à un problème de débruitage d'une variance de  $\frac{1}{\beta}$  et utilisant le MAP.

Avec le GMM à moyenne nulle :

$$p(z) = \sum_{k=1}^K \omega_k \mathcal{N}(z, 0, \Sigma_k). \quad (46)$$

avec  $z \in \mathbb{R}^{\tau_N}$ ,  $K$  le nombre de gaussiennes,  $\Sigma_k \in \mathbb{R}^{\tau_N \times \tau_N}$  la covariance et  $\omega_k > 0$  des poids tels que  $\sum_{k=1}^K \omega_k = 1$ . Les paramètres peuvent être appris grâce à l'algorithme EM.

Les calculs des  $z_i$  deviennent :

$$\hat{z}_i = \underset{z_i \in \mathbb{R}^{\tau_N}}{\operatorname{argmin}} \frac{\beta}{2} \|\tilde{z}_i - z_i\|^2 - \log \sum_{k=1}^K \omega_k \mathcal{N}(z_i, 0, \Sigma_k). \quad (47)$$

Dans cette situation non convexe, Zoran et Weiss proposent une approximation dans l'algorithme EPLL afin de contourner le problème. Cela consiste à trouver la composante  $k_i^*$  optimale  $\forall i$  qui représente le mieux le patch  $z_i$  connaissant déjà  $\tilde{z}_i$ , c'est à dire, celle qui maximise le mieux la vraisemblance :

$$\begin{aligned} k_i^* &= \underset{1 \leq k \leq K}{\operatorname{argmax}} p(\tilde{z}_i | k) \omega_k \\ &= \underset{1 \leq k \leq K}{\operatorname{argmax}} \mathcal{N}(\tilde{z}_i, 0, \Sigma_k) \omega_k \end{aligned} \quad (48)$$

Avec le GGMM à moyenne nulle [9] :

$$p(z) = \sum_{k=1}^K \omega_k \mathcal{G}(z, 0, \Sigma_k, \nu_k) \quad (49)$$

avec  $z \in \mathbb{R}^{\tau_N}$ ,  $K$  le nombre de gaussiennes,  $\Sigma_k \in \mathbb{R}^{\tau_N \times \tau_N}$  la covariance,  $\omega_k > 0$  des poids tels que  $\sum_{k=1}^K \omega_k = 1$  et  $\nu \in \mathbb{R}^{\tau_N}$  un paramètre de forme. Ils peuvent aussi être appris grâce à l'EM.

Dans ce cas, le calcul des  $z_i$  devient :

$$\hat{z}_i = \underset{z_i \in \mathbb{R}^{\tau_N}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|\tilde{z}_i - z_i\|^2 - \log \sum_{k=1}^K \omega_k \mathcal{G}(z_i, 0, \Sigma_k, \nu_k) \quad (50)$$

De la même façon, le problème est non convexe. On cherche donc les  $k_i^*$  qui maximisent les vraisemblances  $\forall i = 1 \dots N$  :

$$\begin{aligned} k_i^* &= \underset{1 \leq k \leq K}{\operatorname{argmax}} \omega_k p(\tilde{z}_i | k) \\ &= \underset{1 \leq k \leq K}{\operatorname{argmin}} -\log \omega_k - \log p(\tilde{z}_i | k) \end{aligned} \quad (51)$$

Et ce afin d'estimer les  $\hat{z}_i$  :

$$\begin{aligned} \hat{z}_i &= \underset{z_i \in \mathbb{R}^{\tau_N}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|\tilde{z}_i - z_i\|^2 - \log \mathcal{G}(z_i, 0, \Sigma_{k_i^*}, \nu_{k_i^*}) \\ &= \underset{z_i \in \mathbb{R}^{\tau_N}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|\tilde{z}_i - z_i\|^2 + \|\Sigma_{\nu_{k_i^*}}^{-1/2} z_i\|_\nu^\nu \end{aligned} \quad (52)$$

La deuxième ligne est obtenue grâce à la proposition (Voir Proposition 1.3.1). L'avantage est que l'on sépare le problème en  $\tau_N$  optimisations à une dimension plus faciles à calculer :

$$\hat{z}_i = U_{k_i^*} \hat{u}_i \quad (53)$$

où

$$(\hat{u}_i)_j = s((\tilde{u}_i)_j, \sigma, (\lambda_{k_i^*})_j, (\nu_{k_i^*})_j) \quad (54)$$

avec

$$\tilde{u}_i = U_{k_i^*}^t \tilde{z}_i \quad (55)$$

et

$$s(u, \sigma, \lambda, \nu) = \operatorname{argmin}_{t \in \mathbb{R}} \frac{1}{2\sigma^2}(u - t)^2 + \frac{|t|^\nu}{\lambda_\nu^\nu} \quad (56)$$

où

$$\lambda_\nu = \lambda \sqrt{\frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}} \quad (57)$$

La fonction  $s : \mathbb{R} \rightarrow \mathbb{R}$  est appelée **fonction de rétrécissement** (shrinkage function) et est non linéaire lorsque  $\nu \neq 2$ . Il devient donc difficile de la calculer et il faudra peut-être l'approcher.

Une autre difficulté réside dans le calcul des vraisemblances. On peut utiliser le fait que  $\forall i \tilde{z}_i = z_i + b_i$  avec deux variables aléatoires indépendantes  $z_i \sim \mathcal{G}(0, \Sigma_k, \nu_k)$  et  $b_i \sim \mathcal{N}(0, \sigma^2 Id_p)$  cette dernière représentant le bruit sur le patch centré en  $i$ .

Le calcul de la distribution devient un calcul de convolution entre ces deux dernières :

$$\begin{aligned} -\log p(\tilde{z}_i | k) &= -\log \int_{\mathbb{R}^{\tau_N}} \mathcal{G}(\tilde{z}_i - z, 0, \Sigma_k, \nu_k) \mathcal{N}(z, 0, \sigma^2 Id_{\tau_N}) dz \\ &= \sum_{j=1}^{\tau_N} f((U_k^t \tilde{z}_i)_j, \lambda_{k,j}, \nu_{k,j}) \end{aligned} \quad (58)$$

La deuxième ligne est obtenue grâce à la proposition 2.3.1 (Voir 1.3.1). On transforme le problème en  $\tau_N$  intégrations à une dimension.  $f$  est la **fonction de contradiction** (discrepancy function) de  $\mathbb{R}$  dans  $\mathbb{R}$  telle que :

$$f(\tilde{u}_i, \sigma, \lambda, \nu) = f_{\sigma, \lambda}^\nu(x) = -\log \int_{\mathbb{R}} \mathcal{G}(\tilde{u}_i - t, 0, \lambda, \nu) \mathcal{N}(t, 0, \sigma^2) dt. \quad (59)$$

Ici on a que  $\tilde{u}_i = U_{k_i^*}^t \tilde{z}_i$ . Le but est de calculer ou de s'approcher au mieux du résultat de la fonction de contradiction  $f$  (voir Annexe B) qui est non quadratique lorsque  $\nu \neq 2$ .

### 3.2.2 Fast EPPL (FEPLL) de Deledalle pour le GMM à moyenne nulle [8]

Pour rappel, on a la décomposition de la covariance  $\Sigma_k = U_k D_k U_k^t$  avec  $U_k \in \mathbb{R}^{\tau_N \times \tau_N}$  unitaire  $\forall k$  et  $D_k = \operatorname{diag}(\lambda_1, \dots, \lambda_{\tau_N})^2$  diagonale contenant les éléments  $\lambda_j^2$  positifs. L'algorithme 1 résume l'EPPL dans le cas d'un GMM :

**Algorithme 1** EPPL via Eigenspace Implementation for the GMM

$$\hat{x} = y$$

$$\text{On boucle } l \text{ fois} \left\{ \begin{array}{l} \tilde{z}_i \leftarrow \mathcal{P}_i \hat{x} \\ \{\tilde{u}_i^k \leftarrow U_k^t \tilde{z}_i\}_{i=1 \dots N}^{k=1 \dots K} \\ \{k_i^* \leftarrow \operatorname{argmin}_{1 \leq k \leq K} -2 \log \omega_k + \sum_{j=1}^{\tau_N} (\log v_j^k + \frac{[\tilde{u}_i^k]_j^2}{v_j^k})\}_{i=1 \dots N} \\ \{[\hat{u}_i]_j \leftarrow \gamma_j^{k_i^*} [\tilde{u}_i^k]_j\}_{i=1 \dots N}^{j=1 \dots \tau_N} \\ \{\hat{z}_i \leftarrow U_{k_i^*} \hat{u}_i\}_{i=1 \dots N} \\ \tilde{x} \leftarrow \left( \sum_{i=1}^N \mathcal{P}_i^t \mathcal{P}_i \right)^{-1} \sum_{i=1}^N \mathcal{P}_i^t \hat{z}_i \\ \hat{x} \leftarrow (H^t H + \beta \sigma^2 Id_N)^{-1} (H^t y + \beta \sigma^2 \tilde{x}) \end{array} \right.$$

avec  $v_j^k = [D_k]_{jj} + \frac{1}{\beta}$ ,  $\gamma_j^k = \frac{[D_k]_{jj}}{v_j^k}$  et où  $[\tilde{u}]_j$  est le j-ème coefficient de  $\tilde{u}$  et  $[D_k]_{jj}$  est le j-ème coefficient diagonal de  $D_k$ . A chaque itération  $l$  on augmente  $\beta$ .

L'algorithme Fast EPLL propose trois accélérations :

- (1) projeter sur un plus petit sous-espace de l'espace des vecteurs propres de la covariance.
- (2) réduire le nombre de vraisemblance à comparer pour trouver la meilleure composante gaussienne du GMM pour chaque patch.
- (3) analyser seulement un sous-ensemble (aléatoire) des N patchs.

### Speed-Up via Flat Tail Spectrum Approximation

On veut éviter de calculer les  $\tau_N$  coefficients  $\tilde{u}_i^k \forall k$ . Pour cela, on dit qu'il existe un rang  $r_k$  tel que  $\forall j > r_k$  les valeurs propres sont constantes  $[D_k]_{jj} = \lambda_k$ . On note donc  $\tilde{U}_k \in \mathbb{R}^{\tau_N \times r_k}$  contenant les  $r_k$  premières colonnes de  $U_k$  et  $\tilde{U}_k^c \in \mathbb{R}^{\tau_N \times r_k^c}$  les dernières colonnes de  $U_k$ , ici  $r_k^c = \tau_N - r_k$ . On a  $\tilde{U}_k^c(\tilde{U}_k^c)^t = Id^{\tau_N} - \tilde{U}_k \tilde{U}_k^t$ .

Pour la suite on se sert du fait que :

$$(\Sigma_k + \frac{1}{\beta} Id_{\tau_N})^{-1} = \tilde{U}_k (\tilde{D}_k + \frac{1}{\beta} Id_{r_k})^{-1} \tilde{U}_k^t + (\lambda_k + \frac{1}{\beta})^{-1} (Id_{\tau_N} - \tilde{U}_k \tilde{U}_k^t) \quad (60)$$

et

$$(\Sigma_k + \frac{1}{\beta} Id_{\tau_N})^{-1} \Sigma_k = \tilde{U}_k (\tilde{D}_k + \frac{1}{\beta} Id_{r_k})^{-1} \tilde{D}_k \tilde{U}_k^t + \lambda_k (\lambda_k + \frac{1}{\beta})^{-1} (Id_{\tau_N} - \tilde{U}_k \tilde{U}_k^t). \quad (61)$$

L'algorithme 1 peut donc être récrit :

### Algorithme 2 EPLL via Eigenspace Implementation for the GMM (improved)

$$\hat{x} = y$$

$$\text{On boucle 1 fois} \left\{ \begin{array}{l} \tilde{z}_i \leftarrow \mathcal{P}_i \hat{x} \\ \{\tilde{u}_i^k \leftarrow \tilde{U}_k^t \tilde{z}_i\}_{i=1 \dots N}^{k=1 \dots K} \\ \{k_i^* \leftarrow \underset{1 \leq k \leq K}{\operatorname{argmin}} -2 \log \omega_k + r_k^c \log v_{\tau_N}^k + \frac{\|\tilde{z}_i\|^2}{v_{\tau_N}^k} + \sum_{j=1}^{r_k} (\log v_j^k + \frac{[\tilde{u}_i^k]_j^2}{v_j^k} - \frac{[\tilde{u}_i^k]_j^2}{v_{\tau_N}^k})\}_{i=1 \dots N} \\ \{[\hat{u}_i]_j \leftarrow (\gamma_j^{k_i^*} - \gamma_{\tau_N}^{k_i^*}) [\tilde{u}_i^{k_i^*}]_j\}_{i=1 \dots N}^{j=1 \dots \tau_N} \\ \{\hat{z}_i \leftarrow \tilde{U}_{k_i^*} \hat{u}_i\}_{i=1 \dots N} + \gamma_{\tau_N}^{k_i^*} \tilde{z}_i \\ \hat{x} \leftarrow (\sum_{i=1}^N \mathcal{P}_i^t \mathcal{P}_i)^{-1} \sum_{i=1}^N \mathcal{P}_i^t \hat{z}_i \\ \hat{x} \leftarrow (H^t H + \beta \sigma^2 Id_N)^{-1} (H^t y + \beta \sigma^2 \hat{x}) \end{array} \right.$$

où  $v_{\tau_N}^k = \lambda_k + \frac{1}{\beta}$  et  $\gamma_{\tau_N}^k = \frac{\lambda_k}{v_{\tau_N}^k}$ .

En pratique, on approche la covariance  $\Sigma_k$  en remplaçant les valeurs propres les moins significatives par  $\lambda_k$ . Cela permet de garder une bonne approximation tout en réduisant beaucoup les temps de calculs.

## Speed-Up via a Balanced Search Tree

Cette méthode est ce qui fait gagner le plus de temps, car elle permet de réduire considérablement le nombre de comparaison de vraisemblance dans la recherche du  $k_i^*$  optimal pour chaque patch  $i$ .

Le but est de construire un arbre de recherche (une hiérarchie ou clustering) sur les paramètres du GMM. Chaque niveau de l'arbre correspond à un GMM dont on a réduit le nombre de composantes (par 2 par exemple). Chaque noeud  $k$  d'un étage correspond à un ensemble de paramètres  $\Theta_k$  du GMM de l'étage sur lequel on se trouve. Chaque feuille correspond à un ensemble de paramètres  $\Theta_k$  du GMM sur lequel on travaille à la base.

Pour créer cette hiérarchie, on calcule les similarités entre les covariances deux à deux mesurées par la divergence symétrique de KullBack-Leibler (KL),  $\Sigma_1$  et  $\Sigma_2$  sont deux covariances à comparer :

$$KL = (\Sigma_1, \Sigma_2) = \frac{1}{2} Tr(\Sigma_2^{-1} \Sigma_1 + \Sigma_1^{-1} \Sigma_2 - 2 Id_{\tau_N}). \quad (62)$$

En se servant de cette divergence, sur chaque étage  $e$ , on cherche à regrouper nos paramètres en  $L < K$  clusters. On cherche donc la partition  $\Omega^e$  pour un étage  $e$  solution du problème :

$$\operatorname{argmin}_{\Omega^e} \sum_{l=1}^L \sum_{k_1, k_2 \in \Omega_l^e} KL = (\Sigma_{k_1}, \Sigma_{k_2}) \quad (63)$$

tel que  $\bigcup_{l=1}^L \Omega_l^e = [K]$  et  $\Omega_{l_1}^e \cap \Omega_{l_2}^e = \emptyset$  et où  $\Omega_l^e$  est le  $l$ -ème ensemble de composantes gaussiennes de l'étage  $e$ .

Étant donné un étage  $e$ , les nouveaux paramètres du GMM à moyenne nulle de l'étage  $e-1$  sont obtenus en calculant  $\forall 1 \leq l \leq L$  :

$$\omega_l^{e-1} = \sum_{k \in \Omega_l^e} \omega_k^e \quad (64)$$

et

$$\Sigma_l^{e-1} = \frac{1}{\omega_l^{e-1}} \sum_{k \in \Omega_l^e} \omega_k^e \Sigma_k^e \quad (65)$$

avec  $\omega_k^e$  et  $\Sigma_k^e$  les poids et matrices de covariances du  $k$ -ème noeud (ou de la  $k$ -ème composante du GMM de l'étage  $e$ ).

On choisit donc  $L$  comme étant une succession de nombres entiers décroissants jusqu'à 1.

## Speed-Up via the Restriction to a Random Subset of Patches

Lors de l'extraction des patchs on a souvent l'habitude de prendre autant de patchs qu'il y a de pixels dans l'image ( $E = N$  avec  $E$  le nombre de pixels de l'image  $X$  que l'on cherche à reconstruire et  $N$  le nombre de patchs sur lequel on travaille).

Cette accélération consiste à sous-échantillonner l'ensemble des patchs c'est à dire réduire  $N$ . On peut procéder de façon régulière en créant une grille avec un maillage régulier mais plus espacé d'un nombre  $s \in [1, \sqrt{\tau_N}]$ .

On peut aussi procéder de façon stochastique un peu aléatoire en prenant chaque point du maillage régulier mais en y ajoutant une perturbation aléatoire. Par exemple sur chaque point  $(i_0, j_0)$  de la grille, la nouvelle position  $(i, j)$  est donnée par :

$$i_0 - \lfloor \frac{\sqrt{\tau_N} - s}{2} \rfloor \leq i \leq i_0 + \lfloor \frac{\sqrt{\tau_N} - s}{2} \rfloor \quad (66)$$

et

$$j_0 - \lfloor \frac{\sqrt{\tau_N} - s}{2} \rfloor \leq j \leq j_0 + \lfloor \frac{\sqrt{\tau_N} - s}{2} \rfloor \quad (67)$$

Cela peut réduire considérablement le nombre de patchs sur lequel on travaille et donc réduire aussi la complexité en temps et en mémoire.

### 3.3 Discussion

- La plupart du temps, le MMSE et le MAP donnent des résultats différents (cela dépend du choix de  $p$ ) mais les deux se calculent grâce à la densité de probabilité à posteriori.

Exemple sur une densité  $p$  :

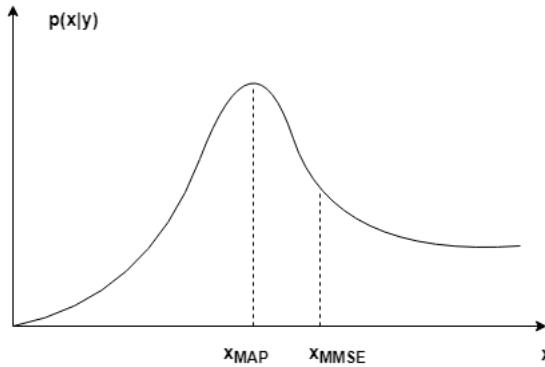


FIGURE 1 – Les estimateurs MMSE et MAP sont évalués en utilisant la p.d.f à posteriori. Le MMSE est le centre de masse alors que le MAP est le maximum de cette p.d.f.

- Le MAP est sans doute plus facile à calculer lorsque la p.d.f contient des exponentielles.

- L'algorithme EPLL peut être très coûteux en temps. L'algorithme FEPLL a le même but que l'algorithme EPLL, il utilise juste des moyens plus ingénieux pour diminuer près des 100 fois le temps de calcul.

- Dans l'algorithme EPLL (et donc FEPLL), les fonctions de rétrécissement et de contradiction sont très difficiles à calculer en pratique, il faudra donc les approcher.

- Même en procédant avec des densités à moyenne nulle pour l'EPLL, on peut retrouver notre problème initial en soustrayant la moyenne à nos patchs à la fin.

## 4 Reconstruction de l'image

Supposons que l'on cherche à reconstruire une image  $\hat{X}$  ayant  $E$  pixels à partir d'un ensemble de patchs  $\{\hat{x}_i\}_{i=1}^N$ . Les patchs se chevauchent et couvrent toute l'image.  $E$  et  $N$  peuvent être égaux. Il y a donc plusieurs choix possibles.

### 4.1 Pixels centraux

Sur chaque patch, on peut prendre le (lorsque  $E$  et  $N$  sont égaux) ou les pixels centraux de telle sorte que ce qu'on en récupère le bon nombre. Chaque centre retourne à sa position initiale.

## 4.2 Moyenne pondérée sur les patchs

Les patchs se chevauchent, donc utiliser une moyenne pondérée pour la restauration d'image n'est pas une mauvaise idée. Les poids  $\{w_i\}_{i=1}^N$  sont définis en fonction des méthodes utilisées.

Enfin pour reconstruire l'image, on utilise :

$$\hat{x} = \frac{\sum_{i=1}^N (w_i)_z (\hat{x}_i)_z}{\sum_{i=1}^N (w_i)_z} \quad (68)$$

Avec le vecteur  $(\hat{x}_i)_z \in \mathbb{R}^E$  égal à  $\hat{x}_i$  lorsqu'on se trouve sur le patch et égal à 0 sinon. De la même façon,  $(w_i)_z \in \mathbb{R}^E$  est égal à  $w_i$  lorsqu'on se trouve sur le patch et est égal à 0 sinon.

Il faudra remettre  $\hat{x}$  dans la bonne dimension pour recréer  $\hat{X}$ .

## 5 Article 1 : Méthode de Sandeep et Jacob pour la Super-Résolution [10]

Plan :

- Extraction des patchs
- Choix de la densité de probabilité : GMM (sous-section 1.2)
- Estimation des paramètres : EM (sous-section 2.2)
- Estimation des patchs : MMSE (sous-section 3.1)
- Reconstruction de l'image : Moyenne pondérée (sous-section 4.2)

### 5.1 Introduction de la méthode

Le but de ce travail est d'estimer une image  $\hat{X}$  à partir d'une image en LR (Basse Résolution) dégradée et observée  $Y$ . On suppose que l'on connaît au moins une partie de  $X$  l'image en HR (Haute Résolution) que l'on cherche à estimer (voir Figure 2).

On assume le fait que :

$$y = SHx + b \quad (69)$$

avec  $S$  un opérateur de sous-échantillonnage de facteur entier  $q$  (facteur de rétrécissement),  $H$  un opérateur de flou et  $b$  du bruit blanc Gaussien.  $x$  et  $y$  sont les représentations vectorielles des images LR et HR respectivement.

A partir de  $Y$  on définit un ensemble de patchs de taille  $\tau \times \tau$  avec un chevauchement maximum :  $\{y_i\}_{i=1}^N$  avec  $y_i \in \mathbb{R}^{\tau^2}$  le vecteur représentant le  $i$ -ème patch et  $N$  le nombre total de patchs.

Chaque patch de  $Y$  peut correspondre à un patch de  $X$  de taille  $\tau q \times \tau q$ . On peut donc noter de la même façon  $\{x_i\}_{i=1}^N$  les  $N$  patchs correspondant aux patchs de  $Y$  avec  $x_i \in \mathbb{R}^{\tau^2 q^2}$ .

On définit les vecteurs  $v_i \in \mathbb{R}^{\tau^2(q^2+1)}$  tels que  $v_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \forall i$ . On a donc un ensemble  $\{v_i\}_{i=1}^n$ .

On rappelle que l'on ne connaît qu'une partie de  $X$ . C'est pourquoi en pratique, l'ensemble  $\{v_i\}_{i=1}^n$  ne contiendra que les patchs où l'image  $X$  est connue ainsi que les patchs de  $Y$  correspondant.  $n$  est alors le nombre total de patchs connus sur  $X$ .

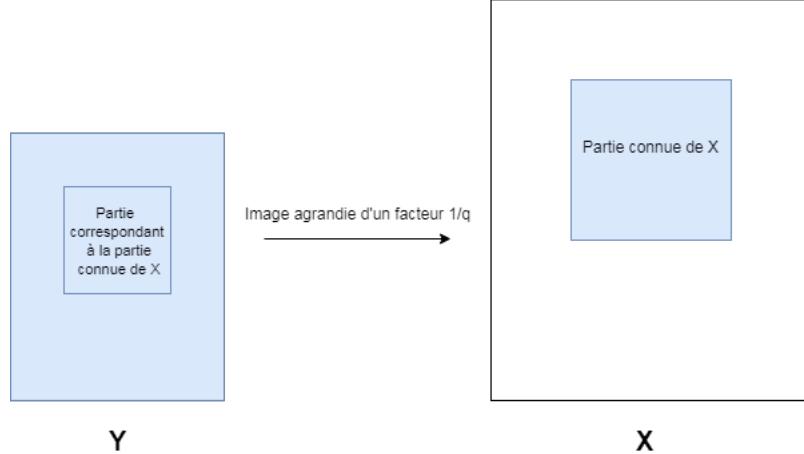


FIGURE 2 – (en bleu : ce que l'on observe)

## 5.2 Estimation des paramètres par la méthode GMM Learning

Cette méthode utilise le GMM, qui considère un modèle global sur toute l'image. On rappelle la densité de probabilité à priori d'un vecteur en utilisant le GMM (Voir Gaussian Mixture Model (GMM)) :

$$p(v) = \sum_{k=1}^K \omega_k \mathcal{N}(v|\mu_k, \Sigma_k) \quad (70)$$

avec

$$\mathcal{N}(v|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{\tau}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(v-\mu)^T \Sigma^{-1}(v-\mu)} \quad (71)$$

la densité de probabilité de la loi normale multidimensionnelle. On travaille sur un patch  $v \in \mathbb{R}^\tau$  avec  $\tau$  la taille du patch,  $\omega$  représentant le poids des Gaussiennes,  $\mu \in \mathbb{R}^\tau$  la moyenne et  $\Sigma \in \mathbb{R}^{\tau \times \tau}$  la covariance.  $K$  est le nombre de Gaussiennes choisi à l'avance.

$\Theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$  caractérise complètement le GMM.

Ici, on va travailler sur les patchs  $\{v_i\}_{i=1}^n$  de taille  $\tau^2(q^2 + 1)$  la concaténation des patchs  $\{x_i\}_{i=1}^n$  et  $\{y_i\}_{i=1}^n$ . On note donc les paramètres de GMM pour les vecteurs  $v_i$  :

$$\mu_{V_k} = \begin{pmatrix} \mu_{H_k} \\ \mu_{L_k} \end{pmatrix}$$

et

$$\Sigma_{V_k} = \begin{pmatrix} \Sigma_{H_k} & \Sigma_{H L_k} \\ \Sigma_{H L_k}^T & \Sigma_{L_k} \end{pmatrix}$$

avec  $\mu_{H_k}$  et  $\mu_{L_k}$  les vecteurs moyens,  $\Sigma_{H_k}$  et  $\Sigma_{L_k}$  les matrices de covariance des parties HR et LR respectivement. La matrice de covariance croisée  $\Sigma_{H L_k}$  caractérise les correspondances entre les paires de patchs HR et LR contenues dans  $v_i$ .

On note donc  $\Theta_V = \{\omega_{V_k}, \mu_{V_k}, \Sigma_{V_k}\}_{k=1}^K$  les paramètres du GMM lorsqu'on travaille sur les patchs  $\{v_i\}_{i=1}^n$ .

Le but est d'estimer au mieux ces paramètres par la méthode d'apprentissage grâce au GMM et à nos observations  $[v_1 \dots v_n]$ . On estime le maximum de vraisemblance sur nos données :

$$\tilde{\Theta}_V = \operatorname{argmax}_{\Theta_V} p(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n | \Theta_V) \quad (72)$$

$$\begin{aligned} &= \operatorname{argmax}_{\Theta_V} \prod_{i=1}^n p(\mathbf{v}_i | \Theta_V) \\ &= \operatorname{argmax}_{\Theta_V} \prod_{i=1}^n \sum_{k=1}^K \omega_{V_k} \mathcal{N}(\mathbf{v}_i; \mu_{V_k}, \Sigma_{V_k}) \\ &= \operatorname{argmin}_{\Theta_V} - \sum_{i=1}^n \log \sum_{k=1}^K \omega_{V_k} \mathcal{N}(\mathbf{v}_i; \mu_{V_k}, \Sigma_{V_k}) \end{aligned} \quad (73)$$

Ceci n'est pas un problème convexe, c'est pourquoi il est difficile d'en calculer une solution exacte. L'algorithme EM (Expectation-Maximisation) permet de contourner ce problème en calculant itérativement un minimum global et non local. (Voir L'algorithme Expectation-Maximisation (EM))

### 5.3 Restauration des patchs

Cette fois ci, on prend tous les patchs sur  $Y$  comme au début de la sous-section 5.1 et  $N$  est toujours le nombre total de patchs de  $Y$ . Dans cette partie, nous allons estimer à posteriori les patchs  $\{\hat{x}_i\}_{i=1}^N$  à partir des patchs  $\{y_i\}_{i=1}^N$  et des paramètres  $\Theta_V$  que l'on vient de calculer.

On détermine pour chaque patch  $y_i$  les paramètres  $\{\omega_{L_k}, \mu_{L_k}, \Sigma_{L_k}\}$  qui le décrivent le mieux. On utilise la vraisemblance :

$$\gamma_{L_k}^i = p(y_i | \mu_{L_k}, \Sigma_{L_k}) \omega_{L_k} \quad (74)$$

$$\hat{k}_i = \operatorname{argmax}_{k=[1, \dots, K]} \gamma_{L_k}^i \quad (75)$$

Pour un patch donné, on trouve nos trois paramètres.

En supposant que le vecteur concaténé  $v_i$  est généré par la composante Gaussienne indexée par  $\hat{k}_i$ , on retrouve les paramètres  $\{\omega_{V_{\hat{k}_i}}, \mu_{V_{\hat{k}_i}}, \Sigma_{V_{\hat{k}_i}}\}$  pour estimer  $\hat{x}_i$ . On calcule le MMSE (Voir Minimum mean square error (MMSE)) :

$$\hat{x}_i = \mathbb{E}[x_i | y_i] \quad (76)$$

$$= \mu_{H_{\hat{k}_i}} + \Sigma_{H_{\hat{k}_i}} \Sigma_{L_{\hat{k}_i}}^{-1} (y_i - \mu_{L_{\hat{k}_i}}) \quad (77)$$

La deuxième ligne est calculée dans l'Annexe MMSE.

### 5.4 Reconstruction de l'image $\hat{X}$

On utilise la moyenne pondérée des patchs estimés  $\{\hat{x}_i\}_{i=1}^N$  se chevauchant pour construire l'image restaurée  $\hat{X}$  (Voir Moyenne pondérée sur les patchs). Les poids pris sont :

$$w_i = e^{-\frac{\gamma}{2} (\hat{x}_i - \mu_{\hat{k}_i})^t \Sigma_{\hat{k}_i}^{-1} (\hat{x}_i - \mu_{\hat{k}_i})}. \quad (78)$$

## 5.5 Résultats Article 1

### 5.5.1 Résultats en utilisant le MMSE avec une seule composante (sous-sous-section 3.1.1) et l'EPLL (sous-sous-section 3.2.1)

Pour rappel,  $q$  est le facteur d'agrandissement,  $\tau \times \tau$  est la taille des patchs de  $Y$ ,  $K$  est le nombre de composantes. Le PSNR (en dB) et le SSIM sont des algorithmes pour comparer l'image originale avec l'image reconstruite. Le PSNR doit être le plus haut possible tandis que le SSIM doit être le plus proche de 1. Le temps est aussi un résultat important.

On compare trois méthodes, une méthode d'interpolation (bicubic) avec les nôtres censés être plus performantes (global GMM learning en utilisant le MMSE et l'EPLL). L'algorithme est exécuté plusieurs fois (ici 10 fois), les résultats sont les moyennes de toutes les exécutions.

Les images  $X$  sur lesquelles on travaille (Figure 3) sont de taille  $512 \times 512$  et les parties connues de  $X$  sont de taille  $256 \times 256$ . Pour l'instant, les parties connues sont les quarts en haut à gauche des images.

L'initialisation des covariances et des moyennes pour l'algorithme EM se fait en utilisant l'algorithme kNN (ici 37 plus proche voisins) sur  $K$  patchs aléatoires de l'image. Le paramètre de régularisation de la covariance  $\varepsilon$  est  $10^{-6}$ . La constante  $\gamma$  est définie pour obtenir la bonne échelle de poids pour le calcul de la moyenne des patchs, elle est réglée à  $10^{-7}$ .



FIGURE 3 – A gauche : les images originales ( $512 \times 512$ ). A droite : les parties connues des images originales ( $256 \times 256$ )

Pour un facteur d'agrandissement  $q = 2$  on prend  $\tau = 4$  et pour  $q = 3$  on prend  $\tau = 3$ . Cela pour limiter le temps de calcul. On fait varier  $K$  (Tableaux 1 et 2).

q=2 $\tau = 4$		K=100		K=200		K=300	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena	MMSE	33.764	0.89805	33.949	0.9015	34.024	0.90216
	EPLL	<b>34.522</b>	<b>0.90219</b>	<b>34.556</b>	<b>0.90251</b>	<b>34.546</b>	<b>0.9024</b>
	bicubic	29.764	0.85095	29.764	0.85095	29.764	0.85095
Cameraman	MMSE	32.004	<b>0.77713</b>	32.08	<b>0.78185</b>	32.157	<b>0.78709</b>
	EPLL	<b>32.457</b>	0.7738	<b>32.441</b>	0.77019	<b>32.433</b>	0.76912
	bicubic	28.045	0.72454	28.045	0.72454	28.045	0.72454
Barbara	MMSE	25.008	0.77614	25.01	0.77773	25.009	<b>0.77835</b>
	EPLL	<b>25.151</b>	<b>0.77739</b>	<b>25.148</b>	<b>0.77773</b>	<b>25.135</b>	0.77709
	bicubic	23.99	0.71101	23.99	0.71109	23.99	0.71109
Hill	MMSE	30.903	<b>0.83459</b>	30.99	<b>0.83833</b>	31.142	<b>0.84406</b>
	EPLL	<b>31.153</b>	0.82698	<b>31.188</b>	0.8284	<b>31.205</b>	0.82843
	bicubic	28.552	0.76773	28.552	0.76773	28.552	0.76773
Shapes	MMSE	<b>26.729</b>	<b>0.97708</b>	<b>25.945</b>	<b>0.97601</b>	<b>25.934</b>	<b>0.97589</b>
	EPLL	24.344	0.90847	24.765	0.90148	24.228	0.89239
	bicubic	23.518	0.90256	23.518	0.90256	23.518	0.90256

TABLE 1 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et différents nombres de composantes avec  $q = 2$  et  $\tau = 4$ . On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic).

q=3 $\tau = 3$		K=100		K=200		K=300	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena	MMSE	29.596	0.82158	29.615	0.82328	29.598	0.82481
	EPLL	<b>30.434</b>	<b>0.83522</b>	<b>30.463</b>	<b>0.8353</b>	<b>30.534</b>	<b>0.83578</b>
	bicubic	26.243	0.7634	26.243	0.7634	26.243	0.7634
Cameraman	MMSE	27.802	0.68415	27.879	0.68868	28.083	<b>0.69793</b>
	EPLL	<b>29.089</b>	<b>0.69351</b>	<b>29.087</b>	<b>0.69294</b>	<b>29.018</b>	0.69282
	bicubic	24.482	0.62084	24.482	0.62084	24.482	0.62084
Barbara	MMSE	23.828	0.67013	23.848	0.67418	23.872	0.67816
	EPLL	<b>24.028</b>	<b>0.68289</b>	<b>24.015</b>	<b>0.68151</b>	<b>24.009</b>	<b>0.68071</b>
	bicubic	22.569	0.61135	22.569	0.61135	22.569	0.61135
Hill	MMSE	27.907	0.70483	27.93	0.70828	28.006	<b>0.71416</b>
	EPLL	<b>28.407</b>	<b>0.71107</b>	<b>28.469</b>	<b>0.71396</b>	<b>28.467</b>	0.71403
	bicubic	25.847	0.63783	25.847	0.63783	25.847	0.63783
Shapes	MMSE	<b>24.242</b>	<b>0.93276</b>	23.525	0.92626	21.508	<b>0.92108</b>
	EPLL	24.172	0.92465	<b>24.869</b>	<b>0.92726</b>	<b>24.524</b>	0.91822
	bicubic	21.364	0.8763	21.364	0.8763	21.364	0.8763

TABLE 2 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et différents nombres de composantes avec  $q = 3$  et  $\tau = 3$ . On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic).

Lors de la reconstruction avec nos méthodes (Figures 4 à 13), on voit que les surfaces sont plus lissées que dans la méthode d'interpolation. On le voit nettement sur les fonds. Cela est dû à la reconstruction des patchs en utilisant une moyenne pondérée. De plus, et cela se voit bien sur l'image "shapes", le bruit est mieux réduit lorsque qu'on utilise le MMSE plutôt que l'EPLL.

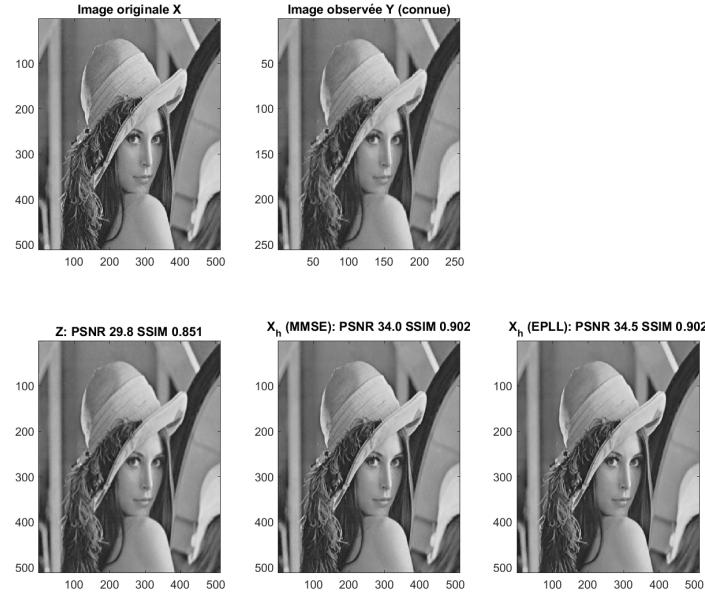


FIGURE 4 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (EPLL)

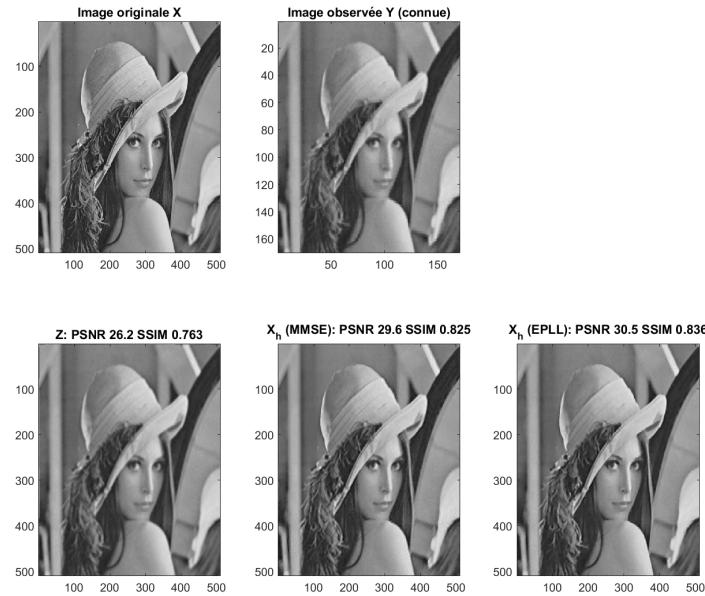


FIGURE 5 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (EPLL)

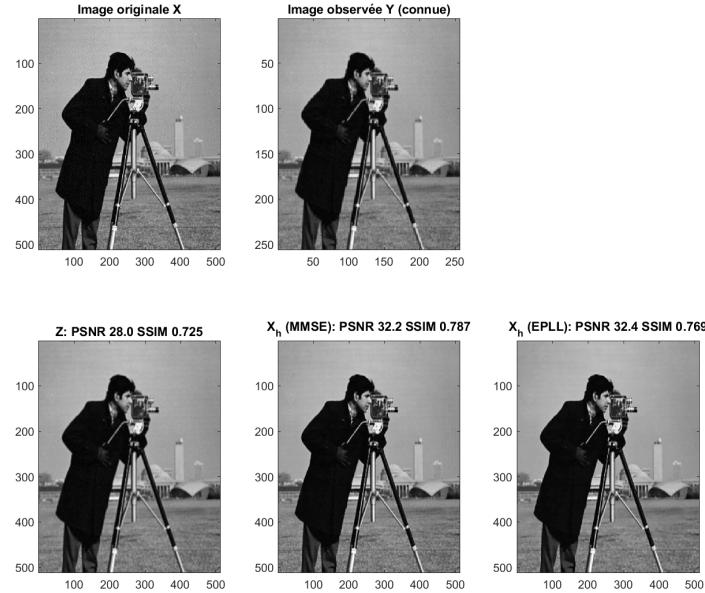


FIGURE 6 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (EPLL)

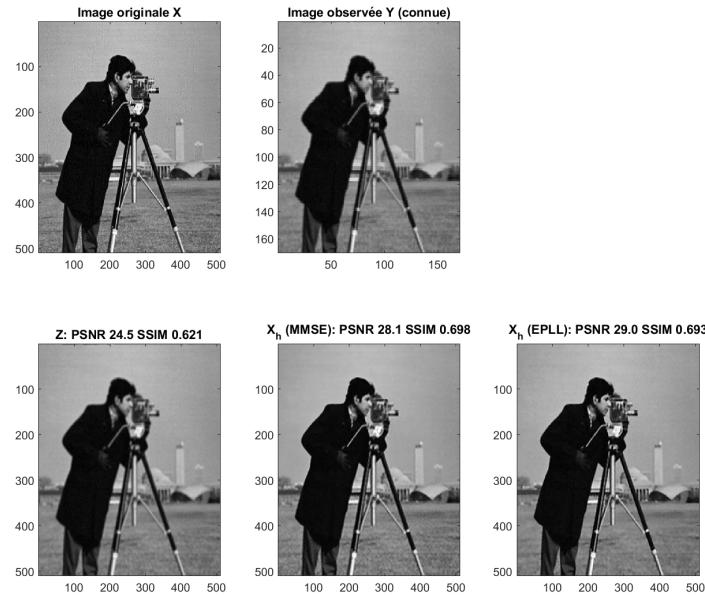


FIGURE 7 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (EPLL)

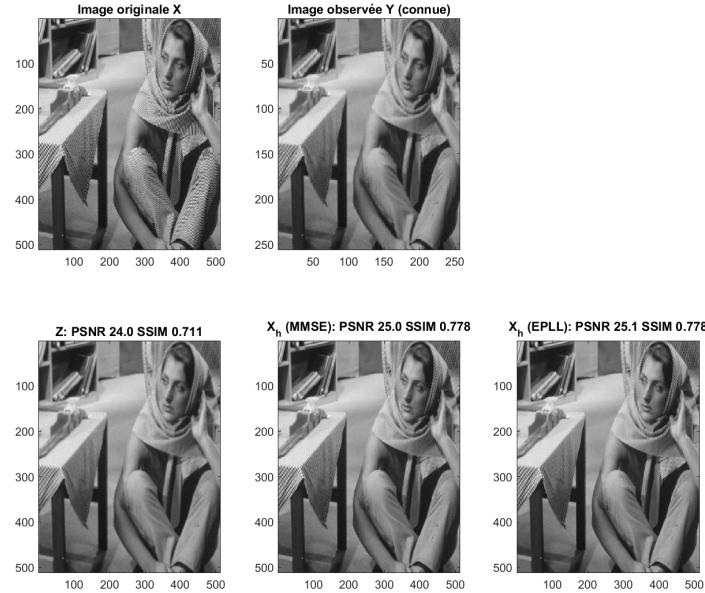


FIGURE 8 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (EPLL)

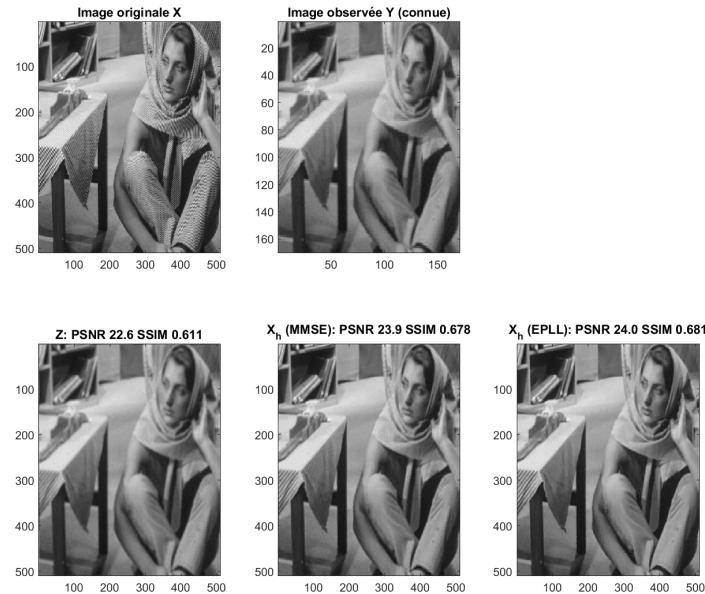


FIGURE 9 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (EPLL)

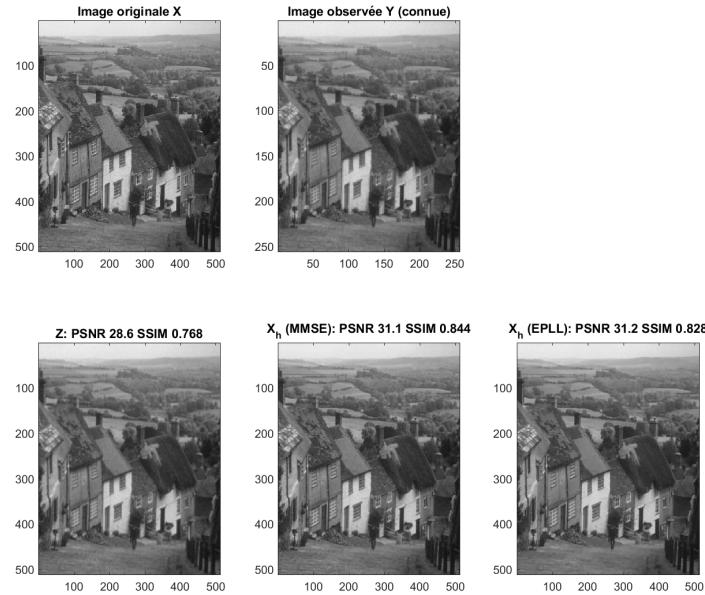


FIGURE 10 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (EPLL)

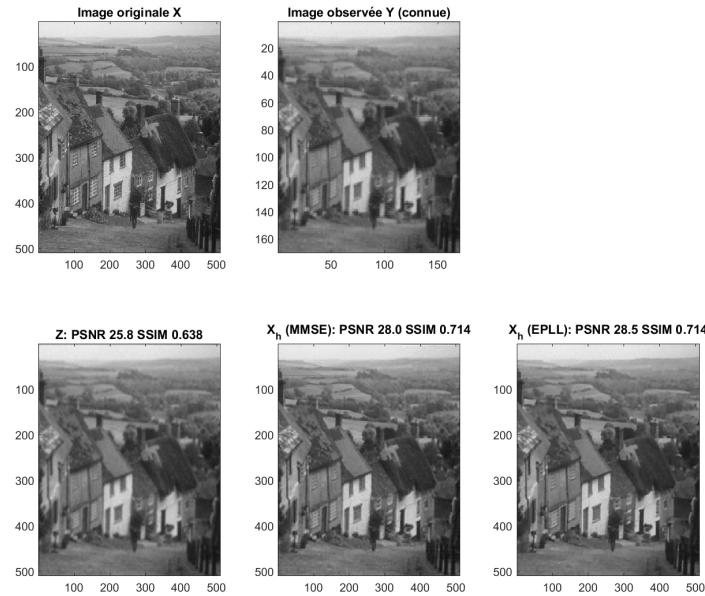


FIGURE 11 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 300$ ) (EPLL)

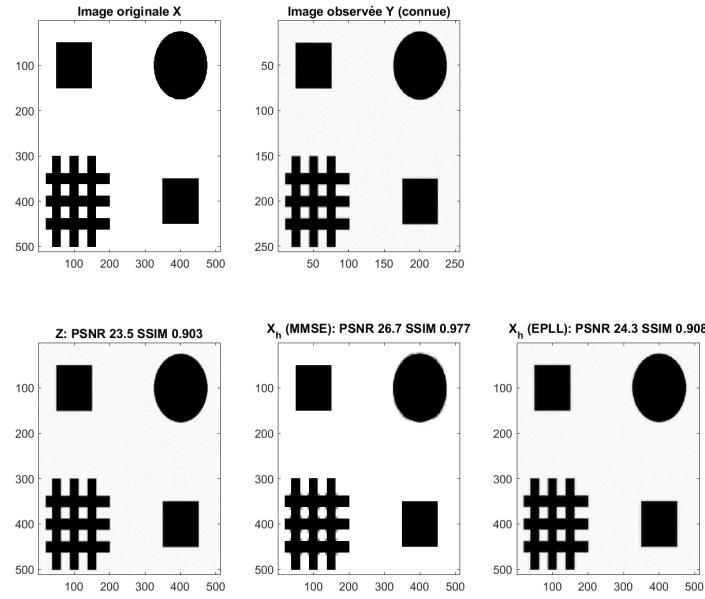


FIGURE 12 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 100$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 100$ ) (EPLL)

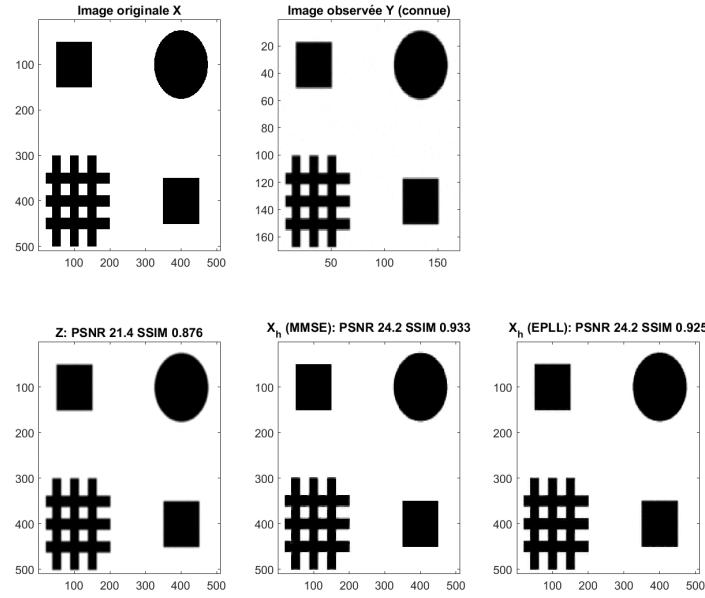


FIGURE 13 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 100$ ) (MMSE) e) Image reconstruite par notre méthode ( $q = 3 \tau = 3$  et  $K = 100$ ) (EPLL)

Dans les Tableaux 1 et 2, on voit que plus  $K$  est grand, plus les résultats sont satisfaisants. Cependant, lorsque  $K$  est trop grand (300 par exemple) le PSNR et le SSIM peuvent diminuer. Ce nombre  $K$  (où les résultats stagnent puis diminuent) est différent si on le calcule avec le MMSE ou l'EPLL. En effet, avec le MMSE les résultats continuent d'augmenter même avec 300 composantes (avec 400 ce n'est plus le cas). Avec l'EPLL cela stagne après les 200 composantes. Les résultats restent meilleurs que la méthode d'interpolation bicubic.

Pour parler du temps de calcul, les reconstructions avec le MMSE et l'EPLL ont un temps de calcul qui dépend linéairement de  $K$ . Cependant, il se peut qu'avec une grande quantité de composantes l'algorithme EM converge plus vite, ce qui peut faire gagner beaucoup de temps.

Le cas particulier est l'image "shapes" qui ne contient que très peu de textures différentes. En effet, dans ce cas même avec  $K = 100$  les résultats ne sont pas satisfaisants et deviennent moins bon lorsque  $K$  augmente. Je pense qu'il vaut mieux limiter le nombre de composantes afin de moins coller aux données et éviter la sur-interprétation. On teste donc pour  $K = 10$  (Tableaux 3 et 4).

q=2 $\tau = 4$		K=10		K=100	
		PSNR	SSIM	PSNR	SSIM
Shapes	MMSE	<b>31.064</b>	<b>0.96707</b>	<b>26.729</b>	<b>0.97708</b>
	EPLL	25.564	0.935	24.344	0.90847
	bicubic	23.533	0.90344	23.518	0.90256

TABLE 3 – Résultats de la super résolution (PSNR et SSIM) sur l'images "shapes" et différents nombres de composantes avec  $q = 2$  et  $\tau = 4$ . On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic).

q=3 $\tau = 3$		K=10		K=100	
		PSNR	SSIM	PSNR	SSIM
Shapes	MMSE	24.544	0.90975	<b>24.242</b>	<b>0.93276</b>
	EPLL	<b>26.309</b>	<b>0.95084</b>	24.172	0.92465
	bicubic	21.364	0.8763	21.364	0.8763

TABLE 4 – Résultats de la super résolution (PSNR et SSIM) sur l'images "shapes" et différents nombres de composantes avec  $q = 3$  et  $\tau = 3$ . On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic).

On obtient de meilleurs résultats sur cette image lorsque l'on prend un  $K$  petit.

Regardons maintenant les images partie par partie. On peut déjà voir nettement si on zoom sur l'image "shapes" (Figures 12 et 13) que les carrés en haut à gauche et en bas à droite sont extrêmement bien reconstruit lorsqu'on utilise le MMSE. On rappelle que le carré en haut à gauche fait partie de la zone connue en haute résolution de l'image. Le MMSE pourrait mieux reconstruire les textures déjà connues que l'EPLL. Regardons pour  $q = 2$  et  $\tau = 4$  (Tableaux 5 et 6) :

q=2 $\tau = 4$		K=100		K=200		K=300	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena	MMSE	<b>35.217</b>	<b>0.89244</b>	<b>35.965</b>	<b>0.90932</b>	<b>36.741</b>	<b>0.92471</b>
	EPLL	34.893	0.87633	34.951	0.87669	34.995	0.87704
	bicubic	30.944	0.82484	30.944	0.82484	30.944	0.82484
Cameraman	MMSE	<b>32.747</b>	<b>0.73481</b>	<b>33.036</b>	<b>0.75765</b>	33.605	<b>0.7838</b>
	EPLL	32.561	0.72041	32.581	0.71907	<b>32.612</b>	0.71865
	bicubic	28.781	0.68879	28.78	0.68879	28.78	0.68879
Barbara	MMSE	<b>35.051</b>	<b>0.93363</b>	<b>35.81</b>	<b>0.94609</b>	<b>36.645</b>	<b>0.95748</b>
	EPLL	34.091	0.90463	34.184	0.90654	34.199	0.90673
	bicubic	29.044	0.84526	29.047	0.84534	29.047	0.84534
Hill	MMSE	<b>32.24</b>	<b>0.86026</b>	<b>33.052</b>	<b>0.88614</b>	<b>34.099</b>	<b>0.9119</b>
	EPLL	31.82	0.83535	31.891	0.837	31.924	0.83743
	bicubic	28.999	0.76635	28.999	0.76635	28.999	0.76635

TABLE 5 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et différents nombres de composantes avec  $q = 2$  et  $\tau = 4$ . On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic). On regarde les parties connues (haut gauche).

q=2 $\tau = 4$		K=100		K=200		K=300	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena	MMSE	35.85	0.9116	35.891	0.90969	35.665	0.90277
	EPLL	<b>36.479</b>	<b>0.92129</b>	<b>36.551</b>	<b>0.9217</b>	<b>36.586</b>	<b>0.92179</b>
	bicubic	32.073	0.89062	32.073	0.89062	32.073	0.89062
Cameraman	MMSE	32.037	<b>0.85031</b>	32.058	<b>0.85019</b>	31.967	<b>0.84817</b>
	EPLL	<b>32.419</b>	0.84444	<b>32.296</b>	0.83671	<b>32.259</b>	0.83494
	bicubic	28.231	0.77304	28.233	0.77317	28.233	0.77317
Barbara	MMSE	20.806	0.54684	20.792	0.54364	20.778	0.54137
	EPLL	<b>20.933</b>	<b>0.56399</b>	<b>20.926</b>	<b>0.56285</b>	<b>20.911</b>	<b>0.56073</b>
	bicubic	20.422	0.50018	20.422	0.50024	20.422	0.50024
Hill	MMSE	31.02	<b>0.82553</b>	30.937	0.82143	30.954	0.82141
	EPLL	<b>31.36</b>	0.82122	<b>31.422</b>	<b>0.82311</b>	<b>31.445</b>	<b>0.82331</b>
	bicubic	28.855	0.77581	28.855	0.77581	28.855	0.77581

TABLE 6 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et différents nombres de composantes avec  $q = 2$  et  $\tau = 4$ . On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic). On regarde les parties non connues (bas droite).

Comme on le disait, les parties connues sont mieux reconstruites avec le MMSE plutôt qu'avec l'EPLL (Tableau 5) et cela est d'autant plus vrai lorsqu'on augmente  $q$ . A l'inverse, l'EPLL reconstruit mieux les parties non connues (Tableau 6). Une méthode pour bien reconstruire une image avec le MMSE consisterait donc à bien choisir la ou les parties à haute résolution contenant la plus grande diversité de textures de l'image. En revanche, si ce travail d'initialisation est difficile, l'EPLL peut être la meilleure option.

Dans cette optique, l'initialisation de l'image "cameraman" n'est pas adaptée. Cette fois ci réessayons en prenant une autre partie connue (Figure 14).

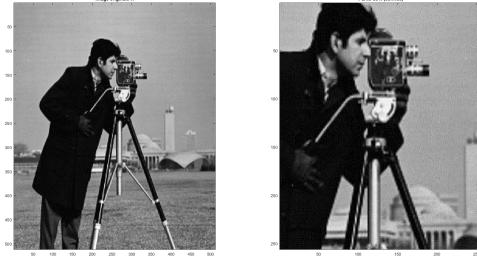


FIGURE 14 – A gauche : l'image originale ( $512 \times 512$ ). A droite : la partie connue de l'image originale ( $256 \times 256$ )

q=2 $\tau = 4$		K=300	
		PSNR	SSIM
Cameraman (Partie connue en haut a gauche)	MMSE	32.157	<b>0.78709</b>
	EPLL	<b>32.433</b>	0.76912
	bicubic	28.045	0.72454
Cameraman (Partie connue centrale)	MMSE	32.318	<b>0.7774</b>
	EPLL	<b>32.574</b>	0.77158
	bicubic	28.045	0.72454

TABLE 7 – Résultats de la super résolution (PSNR et SSIM) sur l'image "cameraman" pour  $K = 300$ ,  $q = 2$  et deux initialisations différentes. On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic).

q=3 $\tau = 3$		K=300	
		PSNR	SSIM
Cameraman (Partie connue en haut a gauche)	MMSE	28.083	<b>0.69793</b>
	EPLL	<b>29.018</b>	0.69282
	bicubic	24.482	0.62084
Cameraman (Partie connue centrale)	MMSE	28.804	0.69342
	EPLL	<b>29.213</b>	<b>0.6974</b>
	bicubic	24.482	0.62084

TABLE 8 – Résultats de la super résolution (PSNR et SSIM) sur l'image "cameraman" pour  $K = 300$ ,  $q = 3$  et deux initialisations différentes. On compare entre nos méthodes (utilisant le MMSE ou l'EPLL) et une méthode d'interpolation (bicubic).

Dans les Tableaux 7 et 8, on peut voir que la deuxième initialisation est la meilleure pour le PSNR. Cela a une grande influence sur les résultats utilisant le MMSE (plus 0.8 dB pour  $q = 3$  ce qui est beaucoup) mais a aussi apporté des changements positifs non négligeables pour l'EPLL. Cela est encore plus vrai lorsque  $q$  est pris plus grand.

### 5.5.2 Résultats en utilisant le MMSE avec toutes les composantes (sous-sous-section 3.1.2)

Cette fois ci on compare, en faisant varier  $K$ , le MMSE avec une composante et avec toutes les composantes. On reprend les initialisations de la Figure 3.

q=2 $\tau = 4$		K=100		K=200		K=300	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena	MMSE (with one component)	<b>33.841</b>	<b>0.8989</b>	<b>33.921</b>	<b>0.90174</b>	<b>33.956</b>	<b>0.9022</b>
	MMSE (with all the components)	31.838	0.87583	31.836	0.87586	31.831	0.87582
	bicubic	29.748	0.85087	29.749	0.85074	29.747	0.85076
Cameraman	MMSE (with one component)	<b>32.012</b>	<b>0.77741</b>	<b>32.093</b>	<b>0.78135</b>	<b>32.149</b>	<b>0.78704</b>
	MMSE (with all the components)	30.052	0.73895	30.064	0.74191	30.094	0.743
	bicubic	28.042	0.72445	28.039	0.72432	28.042	0.72449
Barbara	MMSE (with one component)	<b>25.008</b>	<b>0.77698</b>	<b>24.993</b>	<b>0.77721</b>	<b>24.989</b>	<b>0.77799</b>
	MMSE (with all the components)	24.556	0.73895	24.553	0.73896	24.551	0.73928
	bicubic	23.985	0.71071	23.985	0.71081	23.986	0.71083
Hill	MMSE (with one component)	<b>30.859</b>	<b>0.83369</b>	<b>30.979</b>	<b>0.838</b>	<b>31.051</b>	<b>0.8421</b>
	MMSE (with all the components)	29.392	0.78105	29.364	0.77905	29.354	0.7783
	bicubic	28.545	0.76736	28.547	0.76755	28.547	0.76753

TABLE 9 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et différents nombres de composantes avec  $q = 2$  et  $\tau = 4$ . On compare entre nos méthodes (utilisant le MMSE avec une composante ou toutes les composantes) et une méthode d’interpolation (bicubic).

q=3 $\tau = 3$		K=100		K=200		K=300	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena	MMSE (with one component)	<b>29.679</b>	<b>0.82225</b>	<b>29.725</b>	<b>0.82377</b>	<b>29.779</b>	<b>0.82678</b>
	MMSE (with all the components)	27.722	0.78513	27.673	0.78312	27.628	0.7809
	bicubic	26.241	0.76287	26.243	0.7629	26.239	0.76269
Cameraman	MMSE (with one component)	<b>27.914</b>	<b>0.6871</b>	<b>27.977</b>	<b>0.69061</b>	<b>28.029</b>	<b>0.69588</b>
	MMSE (with all the components)	25.972	0.64324	26.033	0.64444	26.02	0.64456
	bicubic	24.486	0.6212	24.487	0.62109	24.484	0.62111
Barbara	MMSE (with one component)	<b>23.803</b>	<b>0.66781</b>	<b>23.834</b>	<b>0.67197</b>	<b>23.862</b>	<b>0.67569</b>
	MMSE (with all the components)	23.31	0.63821	23.311	0.6378	23.307	0.63677
	bicubic	22.569	0.61108	22.567	0.61097	22.567	0.61103
Hill	MMSE (with one component)	<b>27.909</b>	<b>0.70483</b>	<b>27.997</b>	<b>0.70986</b>	<b>28.03</b>	<b>0.71526</b>
	MMSE (with all the components)	26.504	0.64256	26.493	0.64075	26.472	0.63899
	bicubic	25.845	0.63777	25.841	0.63771	25.843	0.63752

TABLE 10 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et différents nombres de composantes avec  $q = 2$  et  $\tau = 4$ . On compare entre nos méthodes (utilisant le MMSE avec une composante ou toutes les composantes) et une méthode d’interpolation (bicubic).

En considérant toutes les composantes (Tableaux 9 et 10), les résultats deviennent finalement moins bon. En effet, lorsque plusieurs composantes sont aussi vraisemblables, plutôt que de faire un choix, comme avec la première méthode, on choisit de faire la moyenne de toutes les reconstructions. Finalement, ce dernier choix nous amène à une reconstruction finale erronée. Dans ce cas, on perd même l'avantage d'avoir une meilleures reconstruction que l'EPLL sur les parties connues.

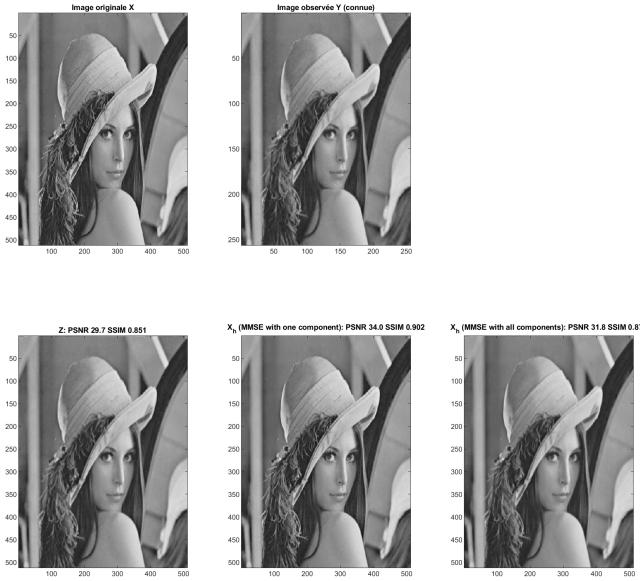


FIGURE 15 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE avec une composante) e) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE avec toutes les composantes)



FIGURE 16 – De gauche à droite puis de bas en haut a) Image originale b) Image observée c) Image reconstruite par interpolation (bicubic) d) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE avec une composante) e) Image reconstruite par notre méthode ( $q = 2 \tau = 4$  et  $K = 300$ ) (MMSE avec toutes les composantes)

On peut noter dans les Figures 15 et 16 qu'en prenant toutes les composantes, on obtient une bonne réduction du bruit. On perd surtout de la qualité au niveau des structures et des contours.

## 6 Article 2 : Méthode de Niknejad, Rabbani et Babaie-Zadeh pour la Super-Résolution [6]

Plan :

- Algorithme LINC (boucler  $l$  fois) :
  - Extraction des patchs
  - Choix de la densité de probabilité : Loi normale (sous-section 1.1)
  - knn
  - Estimation des paramètres : ML (sous-section 2.1)
  - Estimation des patchs : MAP (sous-section 3.2)
  - Reconstruction de l'image : Moyenne pondérée (sous-section 4.2)

### 6.1 Introduction de la méthode

Cette fois ci, on assume le fait que :

$$y = Hx + b \quad (79)$$

avec  $b$  un bruit blanc gaussien,  $H$  un opérateur linéaire non inversible et  $x, y$  sont les représentations vectorielles des images  $X, Y$  respectivement.

On cherche toujours à estimer une image  $\hat{X}$  à partir de  $Y$  mais cette fois ci, on ne connaît rien de  $X$ .

On initialise dans un premier temps  $\hat{X} = Y$ .

La méthode consiste à prendre des patchs exemples choisis avec un pas uniforme sur toute l'image  $\hat{X}$ . On note ces patchs exemples  $\{\hat{x}_r\}_{r=1 \dots R}$  avec  $R$  le nombre total de région. Ces régions sont de taille  $P \times P$  et peuvent se chevaucher pour mieux estimer les patchs sur leurs bords.

On note les patchs à l'intérieur des régions  $\{\hat{x}_{ir}\}_{r=1 \dots R}^{i=1 \dots P^2}$  et  $\{y_{ir}\}_{r=1 \dots R}^{i=1 \dots P^2}$  les patchs correspondant sur  $Y$ .

### 6.2 L'algorithme LINC

Cette méthode similaire à la précédente (utilisant un GMM global sur l'image entière), utilise la loi normale multidimensionnelle localement selon chaque région de l'image. L'algorithme LINC consiste à reconstruire  $\hat{X}$  en faisant itérativement trois étapes (Regroupement, Estimation et Restauration) sur les vecteurs  $\{\hat{x}_{ir}\}_{r=1 \dots R}^{i=1 \dots P^2}$ . Ensuite, on reconstruit l'image entière  $\hat{X}$  avec la même méthode que dans la sous-section 5.4.

#### 6.2.1 Regroupement par la méthode kNN (k plus proche voisin)

Dans chaque région  $r$ , on compare le patch exemple  $x_r$  avec les autres patchs  $\{\hat{x}_{ir}\}_{i=1 \dots P^2}$  et on ne retient que les  $k$  plus similaires, c'est à dire ceux qui minimisent le plus la distance  $d = \|\hat{x}_{ir} - \hat{x}_r\|_2^2$ .

On note  $S_r$  l'ensemble de ces  $k$  patchs dans la région  $r$ .

### 6.2.2 Estimation des paramètres

On utilise la densité de probabilité de la loi normale multidimensionnelle (Voir Loi normale multidimensionnelle) :

$$\mathcal{N}(v|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{\tau}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(v-\mu)^T \Sigma^{-1} (v-\mu)} \quad (80)$$

Avec  $v \in \mathbb{R}^\tau$  nos patchs de taille  $\tau$ ,  $\mu \in \mathbb{R}^\tau$  la moyenne et  $\Sigma \in \mathbb{R}^{\tau \times \tau}$  la covariance.

On cherche à estimer les paramètres des distributions Gaussiennes sur chaque ensemble  $S_r$ . On obtient donc  $\{\mu_r\}_{r=1 \dots R}$  et  $\{\Sigma_r\}_{r=1 \dots R}$  associés à  $\{\hat{x}_r\}_{r=1 \dots R}$ . Pour les obtenir, on estime le maximum de vraisemblance (Voir Estimateur du maximum de vraisemblance (ML)) sur les données choisies :

$$(\hat{\mu}_r, \hat{\Sigma}_r) = \underset{\mu_r, \Sigma_r}{\operatorname{argmax}} \log \mathcal{N}(x \in S_r | \mu_r, \Sigma_r) \quad (81)$$

Les estimations trouvées sont la moyenne ainsi que la matrice de covariance des  $k$  échantillons.

$$\hat{\mu}_r = \frac{1}{k} \sum_{i \in S_r} \hat{x}_{ir} \quad (82)$$

$$\hat{\Sigma}_r = \frac{1}{k} \sum_{i \in S_r} (\hat{x}_{ir} - \hat{\mu}_r)(\hat{x}_{ir} - \hat{\mu}_r)^T \quad (83)$$

### 6.2.3 Restauration des patchs

Après avoir estimé nos paramètres sur chaque région, on cherche maintenant à restaurer les patchs  $\{\hat{x}_{ir}\}_{r=1 \dots R}^{i=1 \dots P^2}$  sachant que l'on connaît  $\{y_{ir}\}_{r=1 \dots R}^{i=1 \dots P^2}$ .

On calcule alors le maximum à posteriori (MAP) (Voir Maximum A Posteriori (MAP)) :

$$\begin{aligned} \hat{x}_{ir} &= \underset{x}{\operatorname{argmax}} \log \mathcal{N}(x|y_i, \hat{\mu}_r, \hat{\Sigma}_r) \\ &= \underset{x}{\operatorname{argmax}} \log(\mathcal{N}(y_i|x, \hat{\mu}_r, \hat{\Sigma}_r) \mathcal{N}(x|\hat{\mu}_r, \hat{\Sigma}_r)) \\ &= \underset{x}{\operatorname{argmin}} \|y_{ir} - H_i x\|_2^2 + \sigma^2 (x - \hat{\mu}_r)^T \hat{\Sigma}_r^{-1} (x - \hat{\mu}_r) \end{aligned} \quad (84)$$

Pour la dernière ligne on se sert du fait que  $x_{ir} \sim \mathcal{N}(\hat{\mu}_r, \hat{\Sigma}_r)$  et  $b \sim \mathcal{N}(0, \sigma I)$ .

Le calcul devient :

$$\hat{x}_{ir} = (H_i^T H_i + \sigma^2 \hat{\Sigma}_r^{-1})^{-1} (H_i^T y_{ir} + \sigma^2 \hat{\Sigma}_r^{-1} \hat{\mu}_r) \quad (85)$$

Ces trois étapes se font récursivement. Après avoir fait assez de boucles, on peut reconstruire  $\hat{X}$ .

## 6.3 Reconstruction de l'image $\hat{X}$

De la même façon que dans la sous-section 5.4, on utilise la moyenne pondérée des patchs estimés  $\{\hat{x}_{ir}\}_{r=1 \dots R}^{i=1 \dots P^2}$  se chevauchant pour construire l'image restaurée (Voir Moyenne pondérée

sur les patchs).

On utilise les poids suivant pour le patch  $\hat{x}_{ir}$  :

$$\omega_{(i,r)} = e^{-\frac{\gamma}{2}(\hat{x}_{ir} - \hat{\mu}_r)^T \hat{\Sigma}_r^{-1} (\hat{x}_{ir} - \hat{\mu}_r)} \quad (86)$$

avec  $\gamma$  une constante de régularisation à déterminer.

Ensuite on calcule  $\hat{x}$  la forme vectorielle de l'image  $\hat{X}$  :

$$\hat{x} = \frac{\left( \sum_r \sum_i \omega_{(i,r)_z}^T \hat{x}_{ir_z} \right)}{\left( \sum_r \sum_i \omega_{(i,r)_z} \right)} \quad (87)$$

avec  $\omega_{(i,r)_z}$  et  $\hat{x}_{ir_z}$  des vecteurs de la taille de l'image qui ont leurs propres valeurs lorsque cela correspond à la bonne position de l'image, et qui ont la valeur 0 sinon.

Grâce au vecteur  $\hat{x}$ , l'image  $\hat{X}$  peut être reconstruite.

## 6.4 Résultats Article 2

Voici quelques exemples d'application grâce à l'algorithme LINC.

### 6.4.1 Débruitage

L'opérateur  $H$  est donc l'opérateur identité.

On rappelle que  $\tau \times \tau$  est la taille des patchs et  $P \times P$  est la taille de chaque région égale à  $30 \times 30$ .

$k$  sera le nombre de voisins à choisir par région lors du kNN fixé à 40,  $\sigma^2$  sera la variance du bruit de l'image. L'espacement entre chaque région en colonne et en ligne est de 6.

Le nombre d'itération de l'algorithme LINC est de 20.

On fait varier  $\sigma$  :

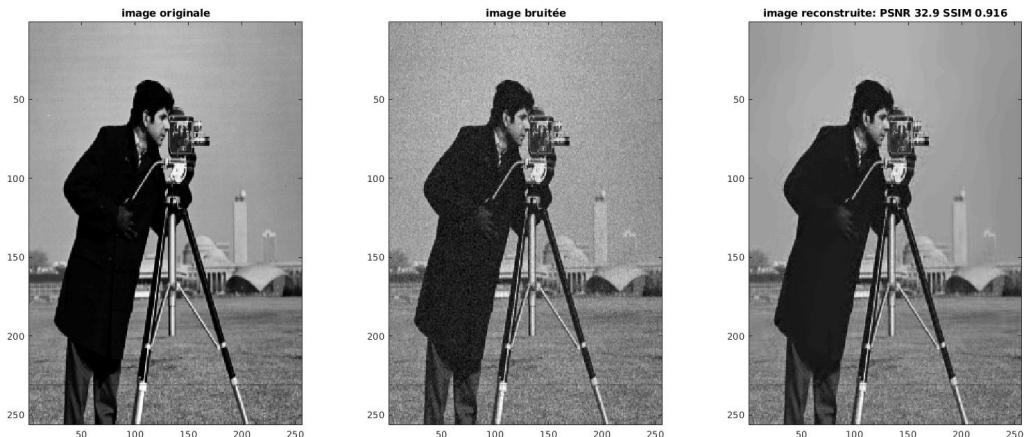


FIGURE 17 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  ( $\sigma = 10, \tau = 8, P = 30, k = 40, r = 6$ , Temps = 231,015) c) Image  $\hat{X}$  reconstruite avec notre méthode (LINC)

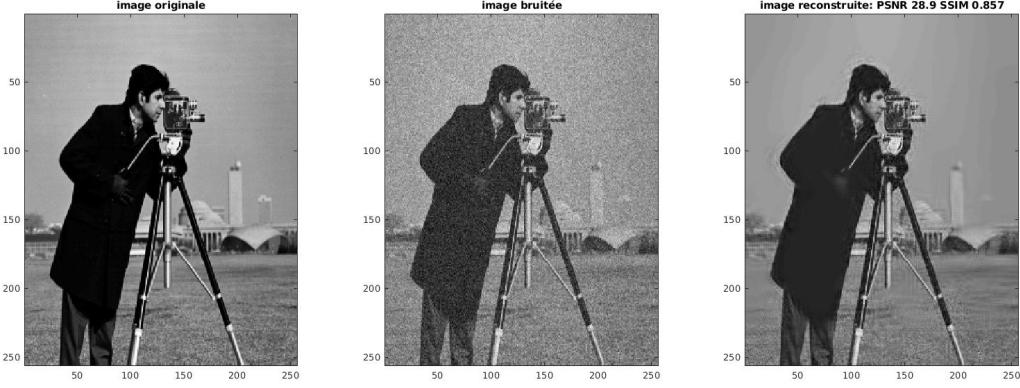


FIGURE 18 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  ( $\sigma = 20, \tau = 8, P = 30, k = 40, r = 6$ , Temps = 233,049) c) Image  $\hat{X}$  reconstruite avec notre méthode (LINC)

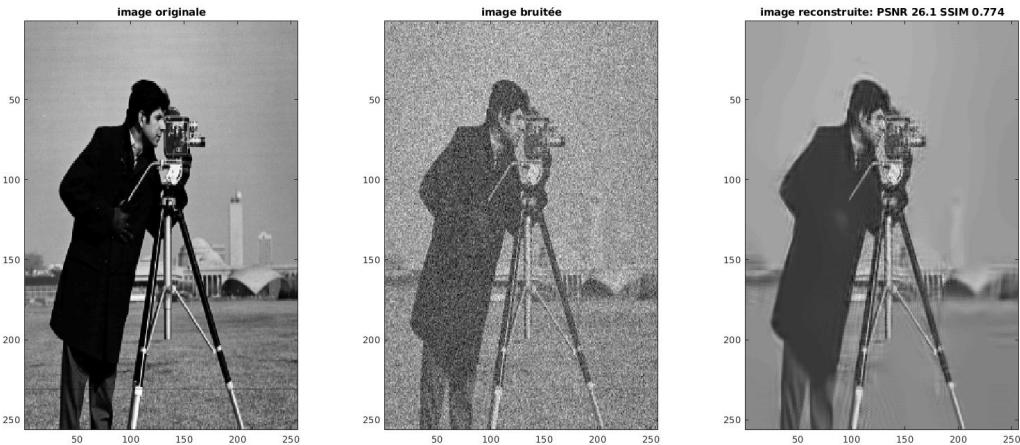


FIGURE 19 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  ( $\sigma = 20, \tau = 8, P = 30, k = 40, r = 6$ , Temps = 238,153) c) Image  $\hat{X}$  reconstruite avec notre méthode (LINC)

### 6.4.2 Inpainting

Dans ce cas l'opérateur est un masque.  $p$  est le pourcentage de pixels manquant. On fait varier  $p$  et on garde les mêmes paramètres :



FIGURE 20 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  ( $p = 0.3, \tau = 8, P = 30, k = 40, r = 6$ , Temps = 232,763) c) Image  $\hat{X}$  reconstruite avec notre méthode (LINC)

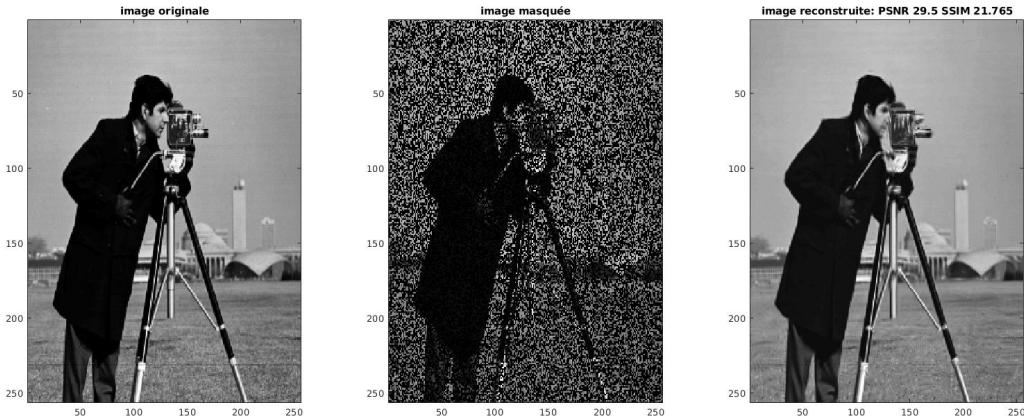


FIGURE 21 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  ( $p = 0.6, \tau = 8, P = 30, k = 40, r = 6$ , Temps = 235,812) c) Image  $\hat{X}$  reconstruite avec notre méthode (LINC)

### 6.4.3 Super Résolution

Ce qui nous intéresse est la super résolution avec l'opérateur  $H$  égal à  $SG$  avec  $S$  un opérateur de sous-échantillonage de facteur de rétrécissement  $q$  et  $G$  un opérateur de flou gaussien.  $\sigma^2$  la variance du bruit est fixée à 2 (bruit standard). On teste pour  $q = 2$  et  $q = 3$  :

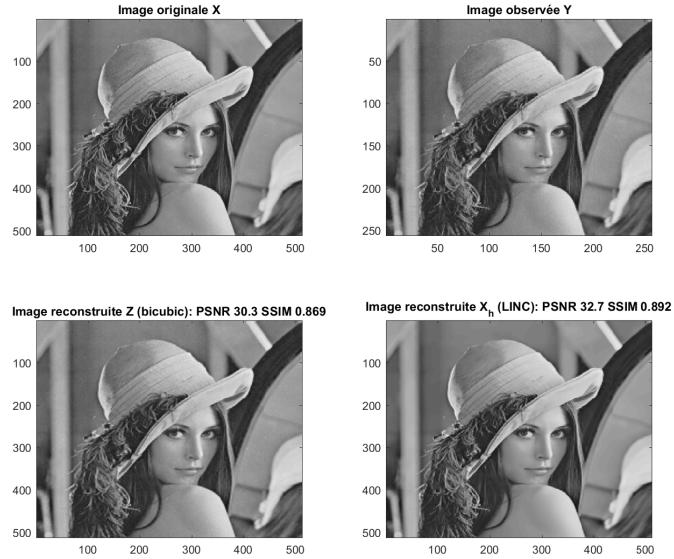


FIGURE 22 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  c) Image reconstruite par interpolation (bicubic) d) Image  $\hat{X}$  reconstruite avec notre méthode (LINC) ( $q = 2$ )

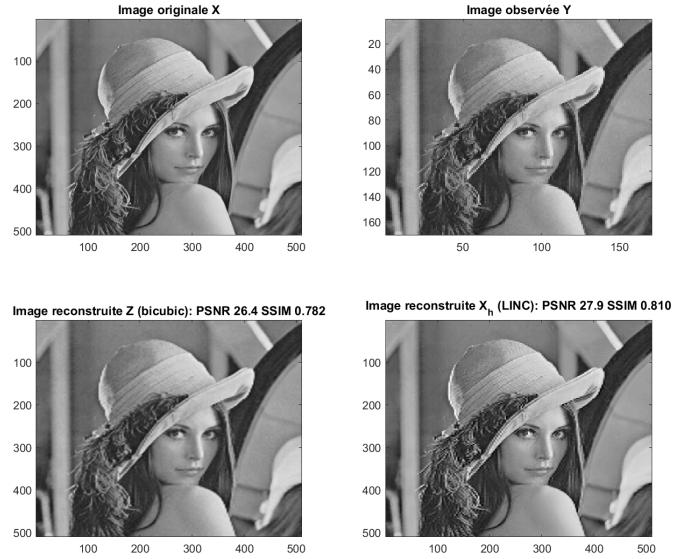


FIGURE 23 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  c) Image reconstruite par interpolation (bicubic) d) Image  $\hat{X}$  reconstruite avec notre méthode (LINC) ( $q = 3$ )

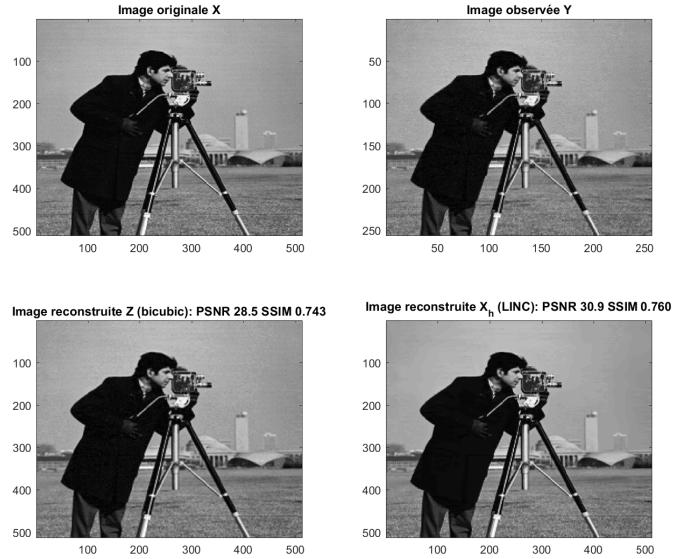


FIGURE 24 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  c) Image reconstruite par interpolation (bicubic) d) Image  $\hat{X}$  reconstruite avec notre méthode (LINC) ( $q = 2$ )

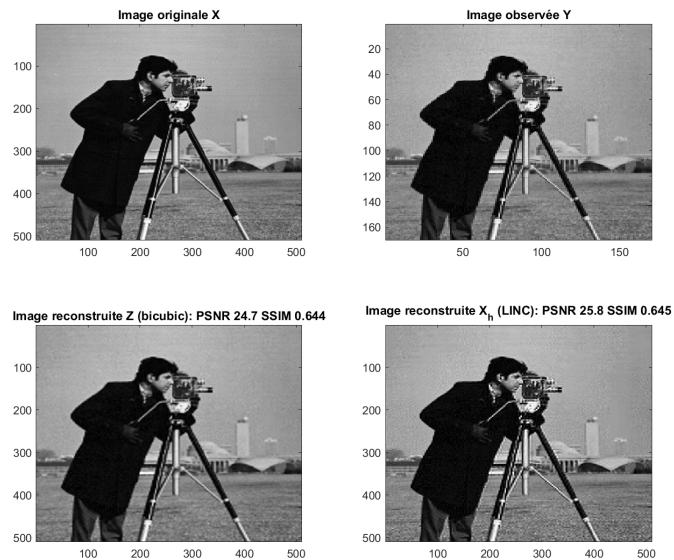


FIGURE 25 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  c) Image reconstruite par interpolation (bicubic) d) Image  $\hat{X}$  reconstruite avec notre méthode (LINC) ( $q = 3$ )

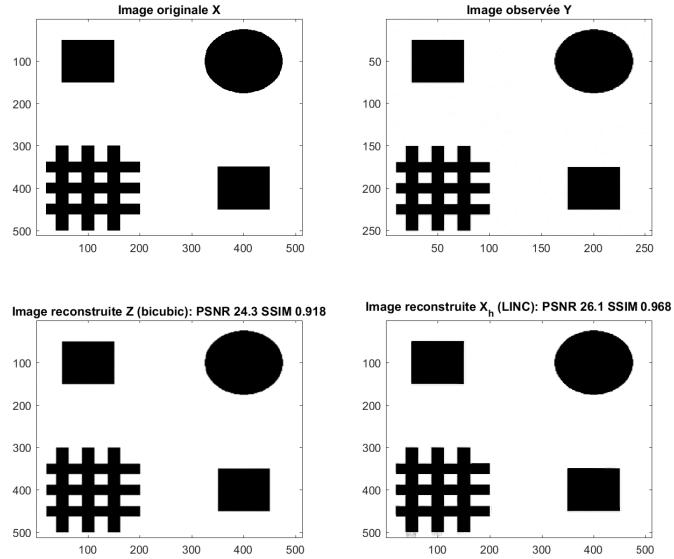


FIGURE 26 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  c) Image reconstruite par interpolation (bicubic) d) Image  $\hat{X}$  reconstruite avec notre méthode (LINC) ( $q = 2$ )

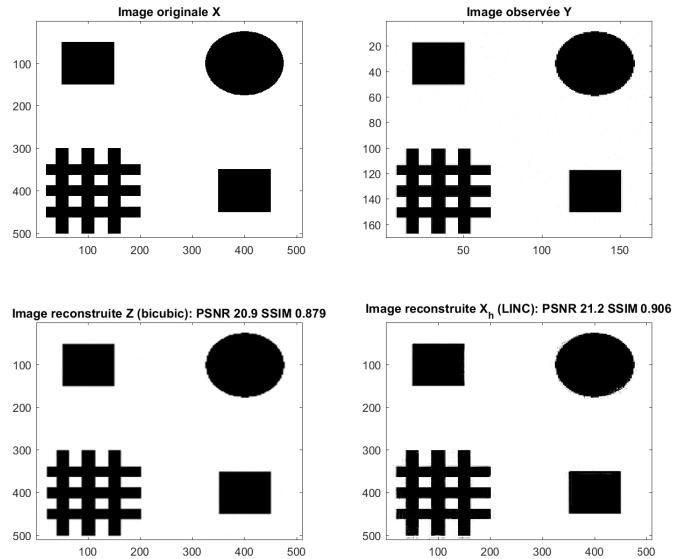


FIGURE 27 – De gauche à droite puis de haut en bas a) Image originale  $X$  b) Image observée  $Y$  c) Image reconstruite par interpolation (bicubic) d) Image  $\hat{X}$  reconstruite avec notre méthode (LINC) ( $q = 3$ )

Si on compare les résultats avec nos modèles globaux, ils ne sont pas à la hauteur pour de la super-résolution. Ce modèle local reste performant pour du débruitage ou de l'inpainting. C'est pourquoi, même pour de la super-résolution, le bruit est bien réduit. On perd cependant beaucoup de détails au niveau des contours.

Si on garde en tête que cette fois ci, nous ne connaissons pas de partie connue en haute résolution, cette méthode peut avoir son avantage. Le temps de calcul dépend linéairement du nombre d'itération de l'algorithme LINC. Comme le modèle est local, la complexité en mémoire est énormément réduite.

## 7 Discussion

- Dans l'article 2, le fait d'avoir  $R$  régions avec  $R$  gaussiennes associées ne change pas vraiment d'un modèle de mélange gaussien. Les méthodes se ressemblent donc beaucoup mais dans l'article 1 nous sommes forcé d'avoir une partie de l'image  $X$  pour entraîner le modèle (EM).

En effet, la restauration se fait itérativement dans l'algorithme LINC afin d'estimer successivement les paramètres et les patchs de chaque région et de se rapprocher de l'image optimale. Dans ce cas les problèmes d'optimisations sont tous convexes et faciles à calculer.

- Dans l'article 2, on travaille sur les régions indépendamment des autres, mais elles se chevauchent. Il y a donc un travail supplémentaire à faire lors de la reconstruction de l'image à la fin de la méthode.

- Dans l'article 2, le knn sert surtout ici à éviter de prendre tous les patchs d'une région afin de gagner du temps.

- La différence de performance entre les modèles globaux et locaux est plus importante lorsque  $q$  est plus grand.

## 8 Autres résultats

### 8.1 Comparaison entre les GMM, LMM et GGMM en utilisant l'EPLL

Dans cette partie, on va comparer différent choix de modèle. On va pouvoir comparer le GMM, le GGMM et le LMM (Laplacian Mixture Model) modélisant des laplaciennes à la place des gaussiennes.

Dans le Tableau 11, on peut voir que les différences sont minimes. Le GGMM étant une généralisation du LMM et du GMM, on obtient en général de meilleurs résultats. Cependant, avec une image plus simple, comme "shapes", le fait d'utiliser un modèle plus simpliste permet d'éviter les erreurs.

		q=2	$\tau = 4$	q=3	$\tau = 3$
		PSNR	SSIM	PSNR	SSIM
Lena	GMM	34.617	0.9024	30.77	0.84018
	LMM	34.67	0.90418	30.937	<b>0.84518</b>
	GGMM	<b>34.71</b>	<b>0.90471</b>	<b>30.938</b>	0.84475
Cameraman	GMM	32.202	0.76955	29.21	0.69903
	LMM	32.264	0.77201	29.321	<b>0.70387</b>
	GGMM	<b>32.295</b>	<b>0.77281</b>	<b>29.326</b>	0.70355
Barbara	GMM	25.113	0.76987	24.038	0.68326
	LMM	25.352	0.78995	<b>24.08</b>	<b>0.68713</b>
	GGMM	<b>25.373</b>	<b>0.79054</b>	24.077	0.68655
Hill	GMM	<b>31.249</b>	0.82765	28.619	0.72035
	LMM	31.178	0.82843	<b>28.768</b>	<b>0.72827</b>
	GGMM	31.237	<b>0.83075</b>	28.764	0.7273
Shapes	GMM	<b>29.687</b>	<b>0.97618</b>	<b>27.276</b>	0.95824
	LMM	29.394	0.9761	27.14	<b>0.95953</b>
	GGMM	29.401	0.97529	27.158	0.95896

TABLE 11 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et différents modèles (GMM, LMM, GGMM) utilisant 200 composantes.

## 8.2 Comparaison entre l'EPLL et le FEPLL

Cette fois ci, on va comparer l'EPLL et son extension le FEPLL.

Dans le Tableau 12, on peut voir dans un premier temps que le FEPLL a un avantage certain de rapidité sur l'EPLL (jusqu'à 100 fois plus rapide). Les différences sur le PSNR et le SSIM sont très petites mais on note une légère augmentation des performances sur le FEPLL quelles que soient les images. Cela peut être dû à une meilleure extraction des patchs ou bien à une plus grande simplicité du modèle.

		q=2	$\tau = 4$	q=3	$\tau = 3$
		PSNR	SSIM	PSNR	SSIM
Lena	EPLL	<b>34.494</b>	0.90052	250.38	30.722
	FEPLL	34.482	<b>0.90198</b>	<b>3.2942</b>	<b>30.831</b>
Cameraman	EPLL	32.101	0.76599	237.41	29.194
	FEPLL	<b>32.183</b>	<b>0.77066</b>	<b>3.2028</b>	<b>29.312</b>
Barbara	EPLL	25.053	0.76364	245.75	24.034
	FEPLL	<b>25.155</b>	<b>0.77334</b>	<b>3.3581</b>	<b>24.066</b>
Hill	EPLL	31.091	0.82142	245.97	28.567
	FEPLL	<b>31.219</b>	<b>0.82923</b>	<b>3.2824</b>	<b>28.737</b>
Shapes	EPLL	29.914	<b>0.9758</b>	245.64	27.489
	FEPLL	<b>28.98</b>	0.96742	<b>3.5725</b>	<b>26.592</b>

TABLE 12 – Résultats de la super résolution (PSNR et SSIM) sur différentes images et utilisant un GMM avec 200 composantes. On compare entre nos méthodes (utilisant l'EPLL et le FEPLL).

## 9 Conclusion

Dans ce rapport, nous avons travaillé avec des modèles globaux (sur toute l'image) utilisant des mixtures. Ensuite, avec un modèle local (sur chaque région indépendamment des autres) utilisant une simple gaussienne par région.

Nos modèles globaux de mélange nécessitent le fait que l'on connaisse une partie de l'image en haute résolution. L'apprentissage des données par l'algorithme EM nous permet ensuite de comparer les reconstructions utilisant le MMSE, l'EPLL et le Fast EPLL. Les textures déjà connues sont toujours mieux reconstruites et le plus souvent le MMSE obtient de meilleurs résultats que les EPLL. Le Fast EPLL permet de gagner beaucoup de temps sur l'EPLL et tous deux ont l'avantage de bien reconstruire l'image même lorsque la partie connue ne contient pas un ensemble assez diversifié de textures. Le choix de la partie connue (ou des clichés en haute résolution à réaliser) pour l'apprentissage avec l'EM est donc important. En science des matériaux, cela peut avoir une très grande influence.

Le MMSE peut prendre en compte l'ensemble des composantes du modèle de mélange. La reconstruction est une moyenne pondérée entre les différentes estimations selon chacune des composantes mais il arrive parfois que plusieurs distributions significatives donnent des reconstructions variées. C'est pourquoi, il est selon moi préférable de ne pas supposer l'ensemble des composantes mais seulement la plus vraisemblable afin de garder un maximum de détails.

Le modèle GGMM qui généralisent les distributions gaussiennes et laplaciennes est un modèle plus flexible qui peut être plus intéressant pour la modélisation des propriétés statistiques de nombreuses images.

Pour le modèle local, la méthode proposée est performante pour le débruitage et l'interpolation mais pour la super résolution elle reste moins efficace que les modèles globaux. L'avantage réside dans le fait que l'apprentissage des données ne se fait que sur l'image en basse résolution. On a donc pas besoin d'avoir une partie connue en haute résolution. De plus, la complexité en mémoire est très faible car les régions sont traitées les unes après les autres.

Les résultats de nos expérimentations nous poussent à nous poser quelques questions :

- Tester une autre minimisation de l'erreur pour la reconstruction des patchs de l'image (comme le MMAE minimum mean absolute error) sur un modèle global pourrait donner de meilleurs résultats. La difficulté réside dans ce calcul dont il est difficile d'avoir une solution exacte. Il faudrait l'approcher.

- Utiliser une base de données plus grande de patchs pour l'apprentissage avec l'EM pourrait être plus intéressant pour la suite. Cette base de données serait construite à partir de plusieurs images.

## A Annexe MMSE

Supposons  $\hat{x}(y)$  l'estimation du vecteur  $x \in \mathbb{R}^n$  grâce au vecteur  $y \in \mathbb{R}^m$ .

Le calcul du MSE (Mean Square Error) consiste à faire :

$$MSE(\hat{x}(y)|y) = \mathbb{E}\{(x - \hat{x}(y))^2|y\} \quad (88)$$

L'estimateur MMSE (Minimum Mean Square Error) est donc :

$$\begin{aligned} \hat{x}_{MMSE}(y) &= \operatorname{argmin}_{\hat{x}(y)} MSE(\hat{x}(y)|y) \\ &= \operatorname{argmin}_{\hat{x}(y)} \int_{\mathbb{R}^n} p(x|y)(x - \hat{x}(y))^2 dx \end{aligned} \quad (89)$$

On calcule :

$$\begin{aligned} \frac{\partial MSE}{\partial \hat{x}} &= 0 \\ 2 \int_{\mathbb{R}^n} p(x|y)(x - \hat{x}(y))dx &= 0 \\ 2 \int_{\mathbb{R}^n} xp(x|y)dx - 2\hat{x}(y) \underbrace{\int_{\mathbb{R}^n} p(x|y)dx}_{=1} &= 0 \\ \hat{x}(y) &= \int_{\mathbb{R}^n} xp(x|y)dx \\ \hat{x}(y) &= \mathbb{E}[x|y] \end{aligned}$$

Sachant que  $\int_{\mathbb{R}^n} p(x|y)(x - \hat{x}(y))^2 dx$  est convexe, on a donc :

$$\hat{x}_{MMSE}(y) = \mathbb{E}[x|y] \quad (90)$$

Dans une espace de Hilbert, la projection orthogonale de  $X$  sur un sous-espace vectoriel  $H = Vect(e_1, \dots, e_d)$  avec les  $e_i$  orthogonaux est :

$$\pi_H(X) = \sum_{i=1}^d \frac{\mathbb{E}[X \cdot e_i]}{\mathbb{E}[e_i^2]} e_i \quad (91)$$

L'espérance conditionnelle  $\mathbb{E}[X|Y]$  peut être vue comme une projection orthogonale de  $X$  sur  $H = Vect(1, Y)$  un sous-espace de  $L^2(Y)$ . On prend une base orthogonale  $(1, Y - \mathbb{E}[Y])$ . La projection de  $X$  est :

$$\begin{aligned} \mathbb{E}[X|Y] &= \frac{\mathbb{E}[X \cdot 1]}{\mathbb{E}[1^2]} \cdot 1 + \frac{\mathbb{E}[X \cdot (Y - \mathbb{E}[Y])]}{\mathbb{E}[(Y - \mathbb{E}[Y])^2]} (Y - \mathbb{E}[Y]) \\ &= \mathbb{E}[X] + \frac{COV(X, Y)}{VAR(X)} (Y - \mathbb{E}[Y]) \end{aligned}$$

Avec  $X = (X_1, \dots, X_n)$  et  $Y = (Y_1, \dots, Y_m)$ , on peut généraliser sur l'espace  $H = Vect(1, Y_1, \dots, Y_m)$  l'espérance conditionnelle de  $X$  sachant  $Y$ , c'est un vecteur aléatoire de taille  $m$  :

$$\mathbb{E}[X|Y] = \mathbb{E}[X] + \Sigma_{X,Y} \Sigma_X^{-1} (Y - \mathbb{E}[Y]) \quad (92)$$

avec  $\Sigma_{X,Y} = COV(X, Y)$  et  $\Sigma_X = VAR(X)$

## B Annexe Approximations

La fonction de contradiction pour  $\nu > 0, \sigma > 0$  et  $\lambda > 0$  est :

$$f_{\sigma,\lambda}^\nu(x) = -\log \frac{1}{\sigma\sqrt{2\pi}} \frac{\nu}{2\lambda_\nu \Gamma(1/\nu)} - \log \int_{-\infty}^{\infty} \exp\left(-\frac{(x-t)^2}{2\sigma^2}\right) \exp\left(-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right) dt \quad (93)$$

Analyse théorique :

**B.0.1 Proposition.** Soit  $\nu > 0, \sigma > 0$  et  $\lambda > 0$  les relations suivantes sont vraies :

$$i) \quad f_{\sigma,\lambda}^\nu(x) = \log \sigma + f_{1,\lambda/\sigma}^\nu(x/\sigma), \quad (94)$$

$$ii) \quad f_{\sigma,\lambda}^\nu(x) = f_{\sigma,\lambda}^\nu(-x), \quad (95)$$

$$iii) \quad |x| \leq |y| \Leftrightarrow f_{\sigma,\lambda}^\nu(|x|) \leq f_{\sigma,\lambda}^\nu(|y|), \quad (96)$$

$$iv) \quad \min_{x \in \mathbb{R}} f_{\sigma,\lambda}^\nu(x) = f_{\sigma,\lambda}^\nu(0) > -\infty \quad (97)$$

Maintenant, on peut écrire la fonction de contradiction  $f_{\sigma,\lambda}^\nu(x) : \mathbb{R} \rightarrow \mathbb{R}$  grâce à un terme constant  $\gamma_\lambda^\nu$  et une autre fonction  $\phi_\lambda^\nu : \mathbb{R}_+^* \rightarrow \mathbb{R}$  telle que pour  $\lambda > 0$  et  $\nu > 0$  :

$$f_{\sigma,\lambda}^\nu(x) = \log \sigma + \gamma_{\lambda/\sigma}^\nu + \begin{cases} e^{\phi_{\lambda/\sigma}^\nu(|x/\sigma|)} & \text{si } x \neq 0 \\ 0 & \text{sinon.} \end{cases}$$

$$\phi_\lambda^\nu(x) = \log[f_{1,\lambda}^\nu(x) - \gamma_\lambda^\nu] \text{ et } \gamma_\lambda^\nu = f_{1,\lambda}^\nu(0) \quad (98)$$

On appelle  $\phi_\lambda^\nu$  la fonction de log-contradiction.

Approximation numérique :

On va maintenant décrire l'approximation de la fonction de contradiction  $f_{1,\lambda}^\nu(x)$  proposée par Charles-Alban Deledalle. Pour cela il faut approcher la fonction de log-contradiction  $\hat{\phi}_\lambda^\nu$  pour que :

$$\hat{f}_{1,\lambda}^\nu(x) = \gamma_\lambda^\nu + \exp \hat{\phi}_\lambda^\nu(x) \text{ où } \gamma_\lambda^\nu = f_{1,\lambda}^\nu(0) \quad (99)$$

En se basant sur l'analyse théorique précédente, une solution qui préserve les comportements (concave, asymptotique et croissant) de la fonction de log-contradiction peut être définie en utilisant les approximations suivantes :

$$\hat{\phi}_\lambda^\nu(x) = \alpha_1 \log|x| + \beta_1 - \mathbf{rec}(\alpha_1 \log|x| + \beta_1 - \alpha_2 \log|x| - \beta_2) \quad (100)$$

où  $\mathbf{rec}$  est une fonction de redressement positive, croissante, convexe et qui satisfait :

$$\lim_{x \rightarrow -\infty} \mathbf{rec}(x) = 0 \text{ et } \mathbf{rec}(x) \underset{x \rightarrow \infty}{\sim} x \quad (101)$$

$\alpha_1, \alpha_2, \beta_1$  et  $\beta_2$  sont des paramètres dépendant de  $\lambda$  et  $\nu$ .

Deux fonctions de redressement possibles sont :

$$\mathbf{relu}(x) = \max(0, x) \text{ et } \mathbf{sofrplus}(x) = h \log[1 + \exp(\frac{x}{h})], h > 0 \quad (102)$$

L'approximation de  $\hat{f}_{1,\lambda}^\nu(x)$  est donc paramétrée par six scalaires :  $\gamma_\lambda^\nu, \alpha_1, \beta_1, \alpha_2, \beta_2$  et  $h$ . En pratique, on peut les pré-calculer en testant beaucoup de combinaisons sur  $\lambda$  et  $\nu$ .

## Références

- [1] M. Saïd Allili. Wavelet modeling using finite mixtures of generalized gaussian distributions : Application to texture discrimination and retrieval. *IEEE Trans. Image Processing*, 21(4) :1452–1464, 2012.
- [2] J.-Y. Tourneret F. Pascal, L. Bombrun and Y. Berthoumieu. Parameter estimation for multivariate generalized gaussian distributions. *IEEE Trans. Signal Processing*, 61(23) :5960–5971, 2013.
- [3] G. Sapiro G. Yu and S. Mallat. Image modeling and enhancement via structured sparse model selection. *IEEE International Conference on Image Processing*, page 1641–1644, 2010.
- [4] A. Houdard J. Delon. Gaussian priors for image denoising. *Bartalmío, Marcelo. Denoising of Photographic Images ans Video : Fundamentals, Open Challenges and New Trends*, 2018, 978-3-319-96029-6. (hal-01800758).
- [5] X. Yuan P. Llull D. J. Brady G. Sapiro L. Carin J. Yang, X. Liao. Compressive sensing by learning a gaussian mixture model from measurements. *IEEE Transactions on Image Processing*, 24(1) :106–119, January 2015.
- [6] H. Rabbani M. Niknejad and M. Babaie-Zadeh. Image restoration using gaussian mixture models with spatially constrained patch clustering. *IEEE Transactions on Image Processing*, 24(11), June 2015.
- [7] K. Nasrollahi and T.B. Moeslun. Super-resolution : a comprehensive survey. *Machine Vision and Application*, 25(6) :1423–1468, 2014.
- [8] L. Denis T. Q. Nguyen S. Parameswaran, C. Deledalle. Accelerating gmm-based patch priors for image restoration : Three ingredients for a 100 speed-up. 2017. (hal-01617722v1).
- [9] T. Q. Nguyen S. Parameswaran, C. Deledalle. Image denoising with generalized gaussian mixture model patch priors. 2018. (hal-01700082v2).
- [10] P. Sandeep and T. Jacob. Single image super-resolution using a joint gmm method. *IEEE Transactions on Image Processing*, 25(9) :4233–4244, 2016.
- [11] L. Bombrun Y. Berthoumieu Z. Boukouvalas, S. Said and T. Adali. A new riemannian averaged fixed-point algorithm for mggd parameter estimation. *IEEE Signal Processing Letters*, 22(12) :2314–2318, December 2015.
- [12] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. *International Conference on Computer Vision*, November 2011.