Compression d'image

Lilian Marey, Simon Mariani, Coni Soret

ENSAE Paris

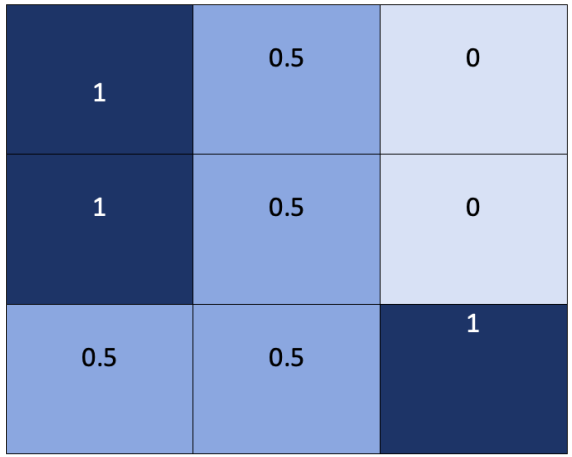
18 Mai 2021

Sommaire

Modèle de Potts

On considère des images aléatoires dont les pixels sont à valeur dans $\{1, ..., K\}$.

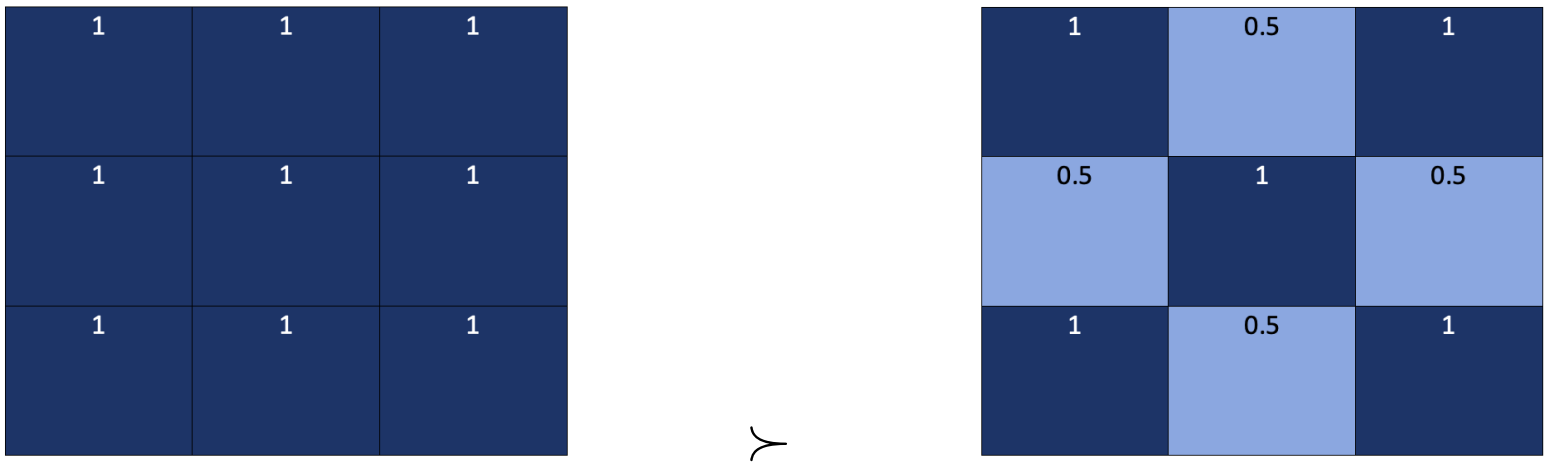
3	2	1
3	2	1
2	2	3



Modèle de Potts

Loi de probabilité sur des images : $\pi(x) = \frac{1}{Z(\beta)} exp(\beta \sum_{i \sim j} \mathbb{1}[x_i = x_j])$

Relation d'équivalence $\sim$: pixels voisins.
Cette loi met plus de poids sur les images avec beaucoup de pixels voisins de même valeur (zone uniforme de gris).

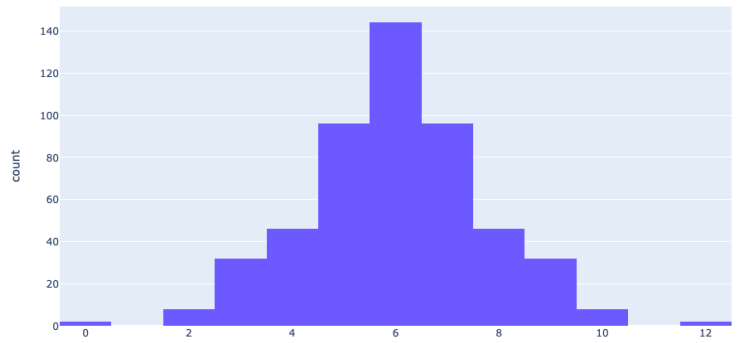


Modèle de Potts

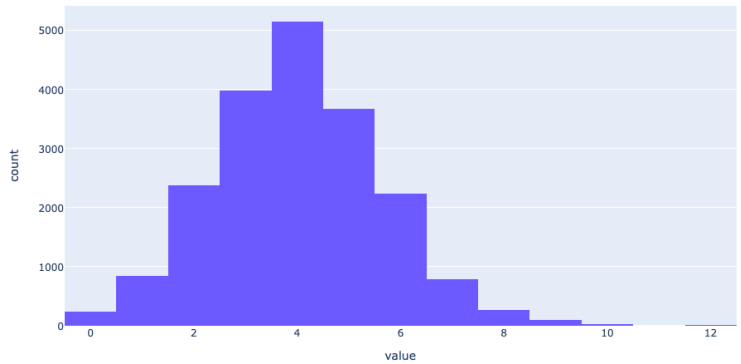
- Rôle de β :
  - β grand $\rightarrow$ images avec zones uniformes très probables
  - β petit $\rightarrow$ probabilité uniforme
- Constante $Z(\beta)$:
  - n^2 la taille de l'image
  - $\gamma(x) = \sum_{i \sim j} \mathbb{1}[x_i = x_j]$ (nombre de paires de pixels en relation dans l'image x)
  - $\pi(x) = \frac{1}{Z(\beta)} exp(\beta\gamma(x))$
  - $Z(\beta)^{-1} = \sum_{x \in \chi} exp(\beta\gamma(x))) = \sum_{k=0}^{f(n)} \sum_{x \in \{x|\gamma(x)=k\}} exp(\beta k) = \sum_{k=0}^{f(n)} e^{\beta k} \#\{x|\gamma(x) = k\}$
    Avec $f(n)$ le nombre maximal de paires en relation dans une image
    $$f(n) = 2n(n-1)$$

Modèle de Potts

Notons $\Gamma(n, k, K) = \#\{x | \gamma(x) = k\}$



Histogramme de $\Gamma(3, k, 2)$



Histogramme de $\Gamma(3, k, 3)$

Ce n'est pas un problème de ne pas avoir une expression de la constante explicite (fonctions de normalisations déjà intégrées dans Python).

Gibbs Sampling

Objectif : simuler la loi π. Exemple ici avec un vecteur de dimension 3.

Initialisation : $x_1 = (x_1^1, x_1^2, x_1^3)$ tiré uniformément

1. x_n^1 tiré selon $\pi_{1|2,3}(\bullet|x_{n-1}^2, x_{n-1}^3)$
2. x_n^2 tiré selon $\pi_{2|1,3}(\bullet|x_n^1, x_{n-1}^3)$
3. x_n^3 tiré selon $\pi_{3|1,2}(\bullet|x_n^1, x_n^2)$

De cette manière, l'algorithme nous permet d'obtenir des vecteurs respectant la distribution de Potts. On évalue la performance en observant l'autocorrélation et la moyenne mobile.

Simulations

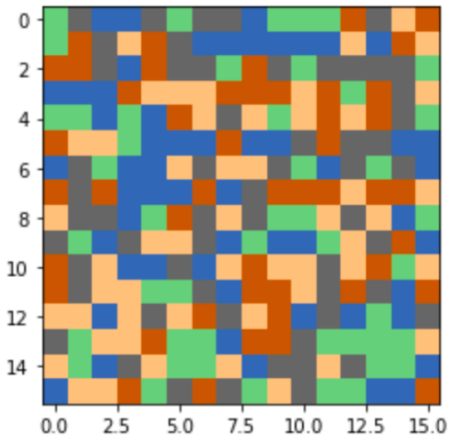
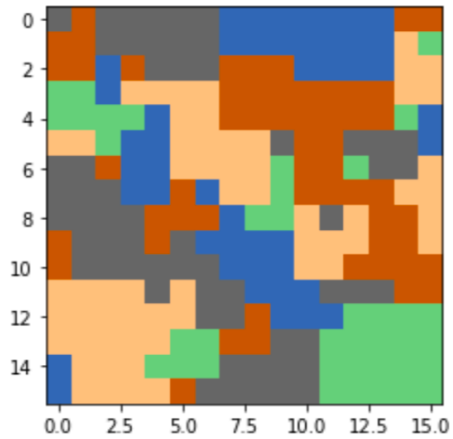
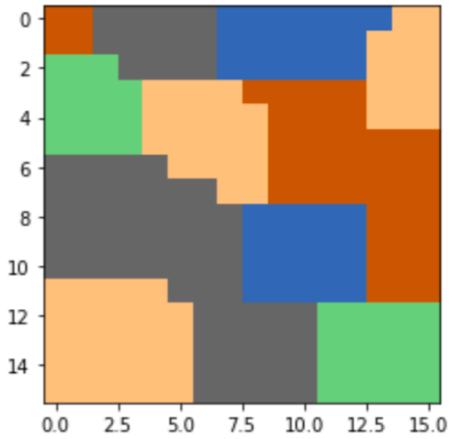


Image initiale



Après 1 itération



Après 10 itérations

$$n = 16, K = 5, \beta = 5$$

Simulations

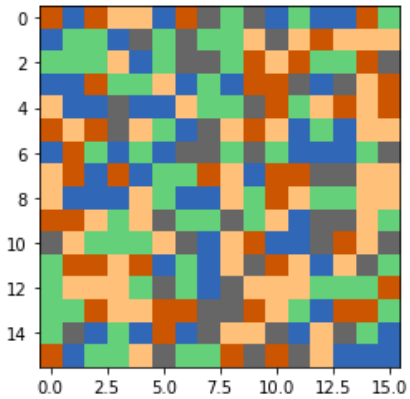
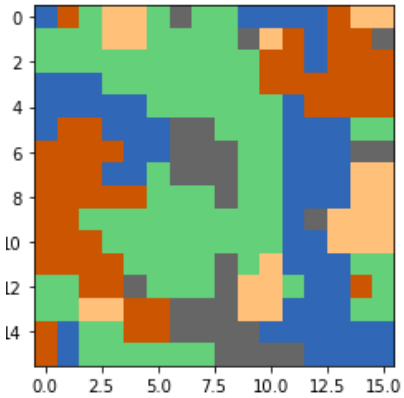
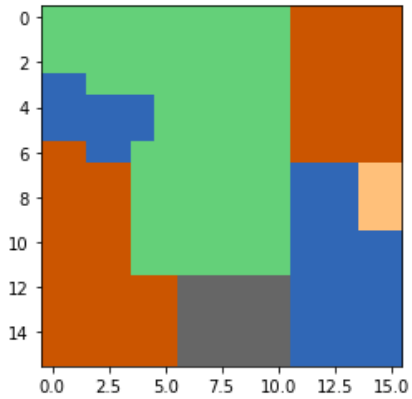


Image initiale



Après 1 itération

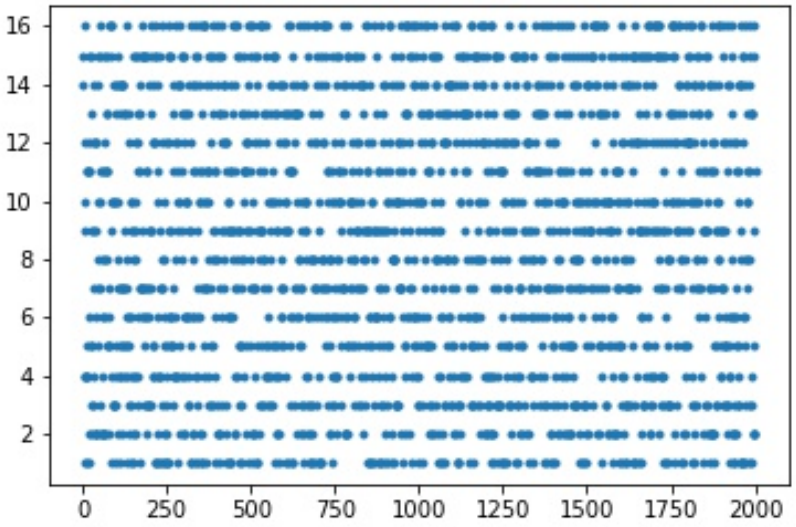


Après 10 itérations

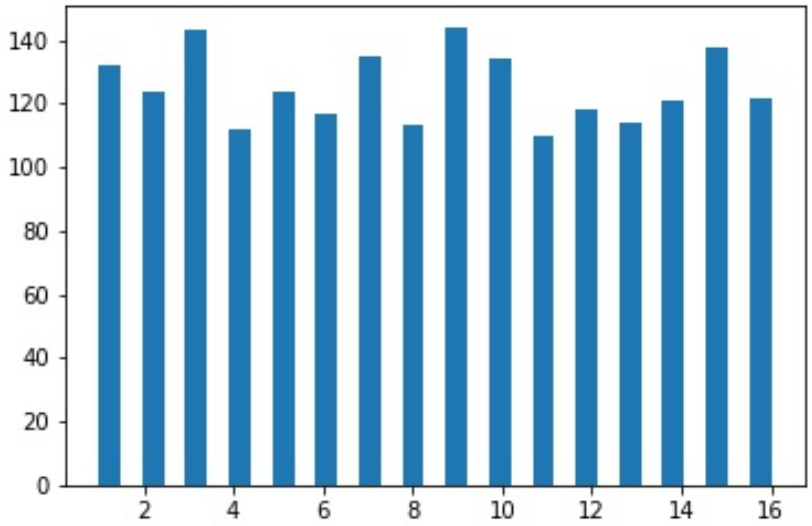
$$n = 16, K = 5, \beta = 15$$

Performance de l'algorithme

On se concentre sur une seule cellule de notre matrice, en prenant $n = 16, K = 10, \beta = 1$. Nous réalisons 2000 itérations du Gibbs Sampler.



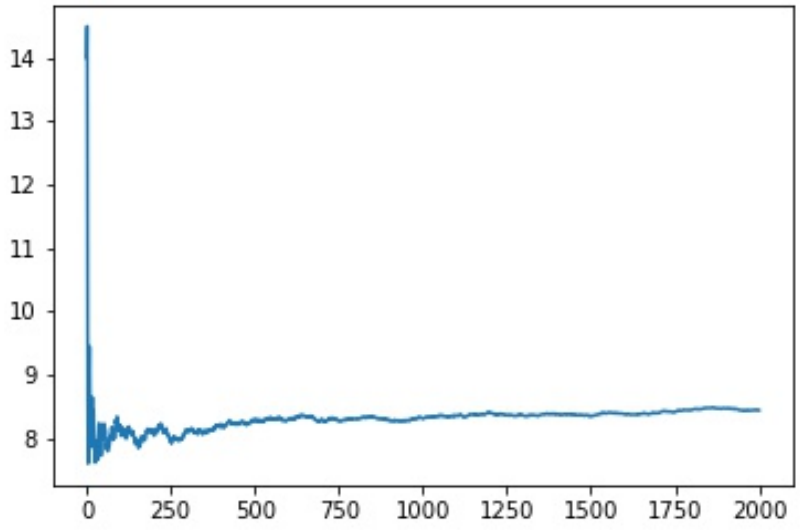
Trace



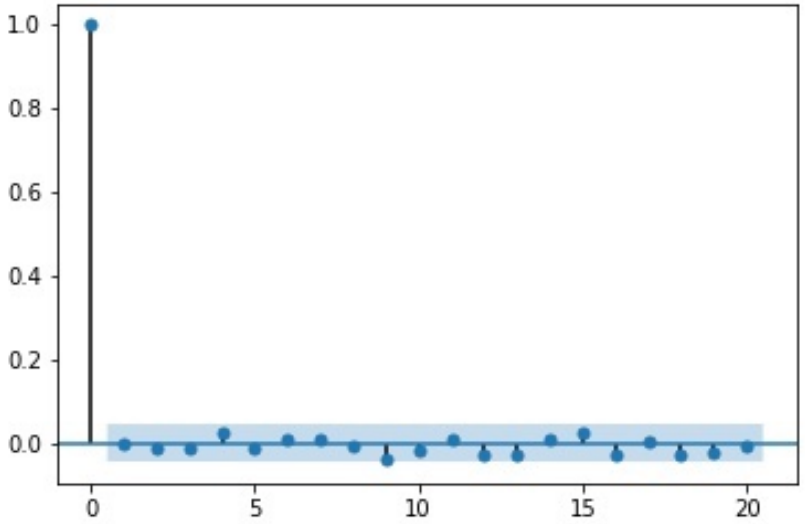
Histogramme

Performance de l'algorithme

Détermination du burn-in :



Moyenne mobile



Autocorrélation

Compression d'image

On suppose que chaque pixel d'une image a une valeur en niveau de gris $y_i|x_i = k \sim \mathcal{N}(\mu_k, \sigma_k^2)$ où les x_i forment un modèle de Potts.

Objectif : simuler les $(x_i), (\mu_k), (\sigma_k^2)$ à l'aide d'un Gibbs-Sampler pour obtenir des bonnes variables latentes et de bons paramètres.

Loi à posteriori

Formule de Bayes : $\pi(\theta|Y) \propto f_{Y|\theta=\theta}(y) * \pi(\theta)$

Variable	Loi à priori	Loi à posteriori conditionnelle
x_i	Potts(β)	$\mathcal{M}((\pi(x_i = k) * f_{Y_i\|x_i=k}(y))_{k=1..K})$
μ_k	$\mathcal{N}(m_k, s_k)$	$\mathcal{N}(\Sigma_k^2 F_k, \Sigma_k^2)$
σ_k^2	$\mathrm{I\Gamma}(\alpha_k, \beta_k)$	$\mathrm{I\Gamma}(\alpha_k + \frac{n_k}{2}, \beta_k + \sum_{x_i=k} \frac{(y_i-\mu_k)^2}{2})$

où $n_k = \#\{i : x_i = k\}$

$$\Sigma_k^2 = \frac{(\sigma_k s_k)^2}{\sigma_k^2 + s_k^2 n_k}$$

$$\frac{m_k}{s_k^2} + \sum_{x_i=k} \frac{y_i}{\sigma_k^2}$$

avec $\beta, m_k, s_k, \alpha_k, \beta_k$ des hyperparamètres à faire varier.

Application à la compression d'image

Principe : Simulation des paramètres et variables pour compresser l'image. Une image est codée d'autant plus simplement qu'elle présente des zones de couleur uniforme.

Méthode :

1. On simule selon l'algorithme les paramètres μ_k, σ_k et les variables latentes x_i associés aux pixels y_i.
2. Pour décompresser, on associe à chaque pixel sa nouvelle valeur :
   - soit directement μ_k
   - soit on tire selon $\mathcal{N}(\mu_k, {\sigma_k}^2)$: crée un peu plus de bruit mais semble plus réaliste

Gain de stockage

Les valeurs des pixels initiales sont dans $[\![0, 255]\!]$ donc stockées sur 8 bits.
En prenant $K = 16$, on peut maintenant stocker les pixels sur 4 bits.
Il faut quand même stocker les différentes valeurs de μ_k, sur 8 bits.
Mais aussi les valeurs de σ_k^2 dans le cas de la deuxième méthode, sur 16 bits.

Ainsi sur une image de 128x128, elle pèse initialement 16 Ko. A la suite de notre compression elle pèse 8.02 Ko, ou 8.05 Ko selon la méthode.

Application à la roue chromatique

On peut voir l'effet de la compression sur toute la gamme de nuances de gris

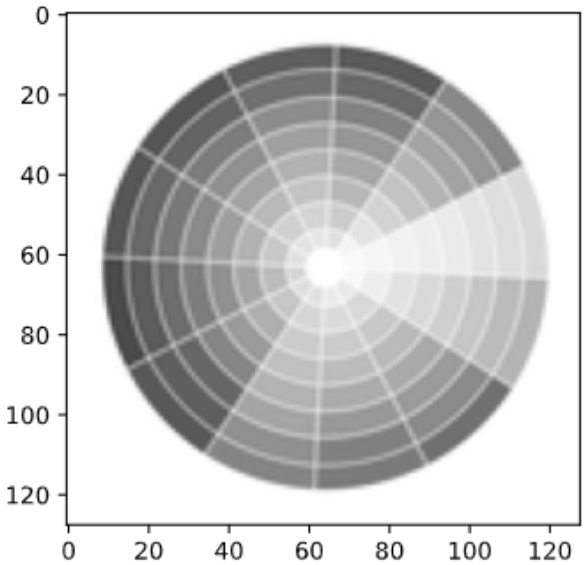
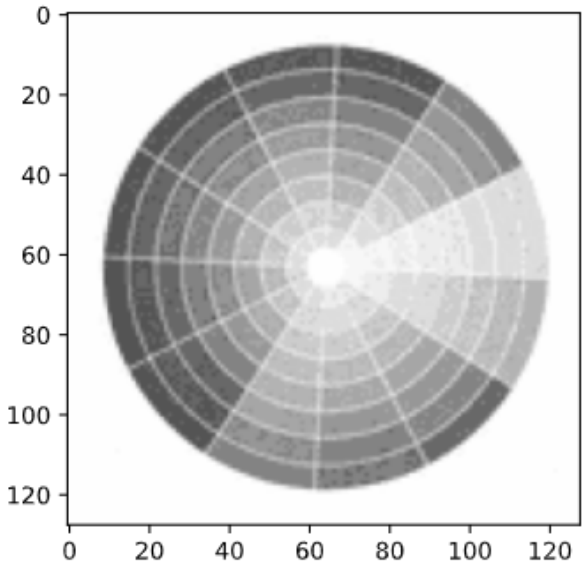
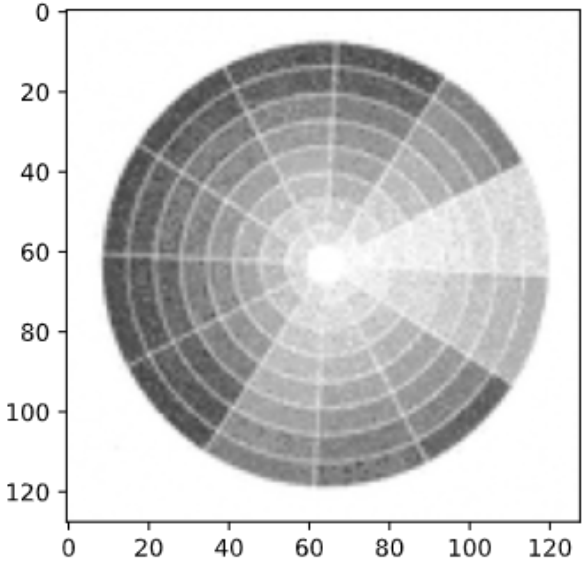


Image initiale



Affectation directe de μ_k



Tirage selon $\mathcal{N}(\mu_k, {\sigma_k}^2)$

Le tirage aléatoire selon une loi normale présente plus de bruit, mais ce *grain* peut donner un aspect plus réaliste à l'image.

Application une photo

Appliquons maintenant la compression à un cas possible.

Image initiale

Affectation directe de μ_k

Tirage selon $\mathcal{N}(\mu_k, {\sigma_k}^2)$

Cas particulier : une compression plus adaptée aux images désequilibrées

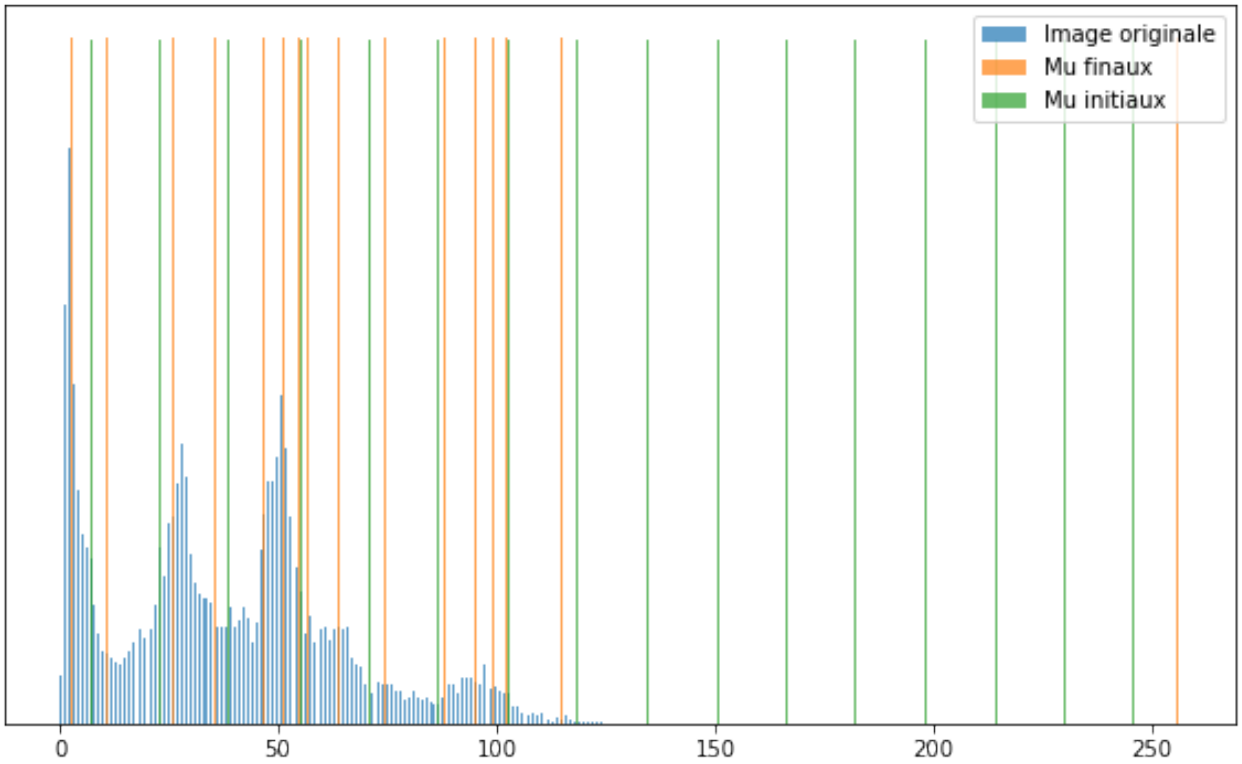
Image initiale

Compression Potts

Compression naïve

Cas particulier : une compression plus adaptée aux images désequilibrées

La compression *naïve* a du mal à s'adapter lorsque les pixels ne décrivent qu'une partie du spectre de luminosité.



Pixels de l'image sombre : aucun n'est éclairé à plus de 120 sur 255 possibles

Variations des paramètre β, K et du nombre d'itérations

On fait varier les paramètres à partir de la photo sombre précédente. La qualité est d'autant plus dégradée que β et K.

1. 20 itérations semblent un bon compromis entre taux de compression et rapidité de l'algorithme.
2. $\beta = 0.1$ semble un bon ordre de grandeur.
3. Un grand K améliore de beaucoup la qualité mais réduit aussi le taux de compression.
4. Les s_k, représentant la variance de la loi à priori de μ_k, permettent la variation plus ou moins forte des μ.

Finalement, il semble intéressant de jouer sur les paramètres en fonction du ratio rapidité de l'algorithme / performance de compression souhaité.

Quelques exemples des variations du paramètres β

$\beta = 0.1$

$\beta = 1$

$\beta = 10$

On visualise l'effet de β qui agit comme un facteur de *lissage* de l'image.

Estimation de β

Une idée pour simuler β pourrait donc être de prendre une loi uniforme comme loi a priori : $\beta \sim \mathcal{U}([a, b])$ avec $[a, b] = [0.1, 10]$ par exemple. Cela laisserait inchangé le reste de l'algorithme de Gibbs-Sampling et ça nous donnerait une nouvelle variable à estimer avec comme densité :

$$f(\beta|x, y, \mu, \sigma^2) = f(\beta|x) \propto f(x|\beta) * f(\beta)$$

Problème : on ne connaît pas explicitement $Z(\beta)$ qui intervient dans $f(x|\beta)$