Projet : Compression basée super-pixels
Codage et compression multimédia [HAI809I]
Analyse et traitement des images [HAI804I]

Université de Montpellier - FDS
1ère année Master IMAGINE
Jean-Baptiste BES - Thomas CARO - Valentin NOYÉ

17 mars 2024

1 Introduction

Durant cette semaine nous avons réglé quelque problèmes qu'il pouvait y avoir sur nos implémentations précédentes et analysé plus précisément nos résultats.

2 Ajout et modification d'algorithmes

2.1 Modification de l'algorithme de Felzenszwalb-Hutterlocher

L'algorithme de Felzenszwalb-Hutterlocher a été amélioré afin de rajouter une seconde passe qui a été omise dans la première implémentation. Cette seconde passe permet de fusionner les petits segments en vérifiant pour chacun d'entre eux si leur taille est supérieure à N_{min}. Afin d'être également conforme à l'implémentation définie dans l'article de recherche, le paramètre k est divisé par 255, c'est à dire que l'algorithme génère maintenant des résultats avec un k espéré plus grand.

2.2 Implémentation du Quick Shift

Le Quick Shift est un algorithme de segmentation rapide basé sur un algorithme de déplacement moyen (mean shift) dont le but est de regrouper des points de données vers les zones de densité plus élevée. Ces points de données 5-dimensionnels représentent l'information spatiale et colorimétrique, dans l'espace CIELab, de chacun des pixels de l'image $p_{x,y} = (x, y, L, a, b)$.
Cet algorithme repose sur la densité des pixels de l'image qui dépendent de ceux de leur entourage, pour ce faire, il s'appuie sur l'estimation de la densité de Parzen et de l'utilisation d'un noyau Gaussien de taille $n_k \times n_k$ avec $n_k = 2k + 1$, défini à partir de k, et d'un écart-type σ, soit les deux paramètres de cet algorithme. Pour chaque pixel $p_{x,y}$, sa densité est représentée par la formule suivante.

$$D(p_{x,y}) = \frac{1}{\sqrt{(2\pi\sigma^2)^5}} \sum_{i=-k}^{k} \sum_{j=-k}^{k} \exp\left(-\frac{1}{2\sigma^2} \left\| p_{x,y} - p_{x+i,y+j} \right\|^2\right)$$

En pratique, le calcul de cette formule est plutôt lent, mais la nature de l'algorithme, basé sur les comparaisons de densités, nous permet d'omettre le premier facteur. Quant à la fonction $\exp(-x)$, il n'est pas obligatoire d'obtenir des résultats parfaits, et donc une approximation est largement suffisante. Par exemple,

$$\exp(-x) \approx \frac{2}{(0.465x + 1)^4 + 1}, \forall x \geq 0$$

Lorsque la densité est calculée pour chaque pixel, une deuxième passe est effectuée sur l'ensemble de ces derniers avec cette information supplémentaire permettant de déterminer tous les voisins $p_{x',y'}$ à une distance inférieure à k du pixel $p_{x,y}$ qui satisfont le prédicat $D(p_{x',y'}) > D(p_{x,y})$. Parmi ces pixels de densité plus grande, $p_{x,y}$ sera donc connecté à son parent dont la distance est minimale, soit

$$\min_{D(p_{x',y'})>D(p_{x,y})} \left\{ \|p_{x,y} - p_{x',y'}\|^2 \right\}$$

Si aucun pixel voisin ne satisfait le prédicat, alors ce pixel p est un maximum local de densité, et tous les pixels rattachés à celui-ci (et récursivement pour chaque sous-arbre) forment un superpixel de la couleur de p.

3 Analyse des résultats

3.1 Mesure SSIM

Tout d'abord nous souhaitions utiliser une métrique supplémentaire pour comparer les performances des méthodes. Le PSNR mesure la différence absolue pixel par pixel entre 2 images mais ne prend pas en compte la structure de l'image. Sachant que l'oeil humain à tendance à bien détecter les différences de structures dans une image, la mesure SSIM nous a semblé pertinente à utiliser.

Le principe de la mesure SSIM est de se déplacer de grille en grille dans l'image et de les analyser. La taille de la grille peut varier mais il semble que dans la littérature des grilles 8 par 8 ou bien 11 par 11 sont souvent utilisées. Nous avons choisi des grilles 8 par 8. Pour chacune de ces grilles nous allons calculer son SSIM avec la formule suivante, x et y étant les grilles dans les 2 images :

$$\mathrm{SSIM}(x,y) = l(x,y) \cdot c(x,y) \cdot s(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_x\sigma_y + c_2)(\mathrm{cov}_{xy} + c_3)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)(\sigma_x\sigma_y + c_3)} \tag{1}$$

où μ_x, μ_y sont les moyennes des intensités des pixels dans les fenêtres x et y, σ_x, σ_y sont les écarts-types des intensités des pixels dans les fenêtres x et y, cov_{xy} est la covariance entre les intensités des pixels dans les fenêtres x et

$$c_1 = (k_1 L)^2, \quad c_2 = (k_2 L)^2, \quad c_3 = \frac{c_2}{2}$$

avec L la dynamique des valeurs des pixels (ici 255 pour les images couleurs en 8 bits), $k_1 = 0.01$ et $k_2 = 0.03$ par défaut.
Nous calculons ce SSIM pour chaque canal de couleur et à la fin effectuons la moyenne de ces SSIM sur toutes les grilles calculées.

3.2 SLIC

Passons à l'implémentation de SLIC précédente qui était très lente. Nous avons recompilé avec l'option de compilation -Ofast et obtenu de bien meilleures performances , presque instantanné pour de faibles nombres de cluster.

Il y avait ensuite un endroit dans le code qui pouvait engendrer une division par 0 lorsquil y avait un grand nombre de cluster demandé. On peut donc maintenant lancer l'algorithme sur par exemple 5400 superpixels, environ le nombre de superpixel utilisé par Felzenszwalb, ce qui n'était pas possible avant. Cependant chaque itération à 5400 superpixels dure environ 5 secondes ce qui avec un seuil bas peut donner des temps de traitement assez long. A ce niveau la Felzenszwalb surclasse largement SLIC qui fonctionne sur le principe d'un K-means.

Nous avons ensuite décidé de prendre la même image que précedemment :

FIGURE 1 – Image Cote

Nous allons utiliser SLIC en demandant 5400 clusters et faire varier le seuil utilisé :

FIGURE 2 – Seuil = 100 000

FIGURE 3 – Seuil = 50 000

FIGURE 4 – Seuil = 30 000

FIGURE 5 – Seuil = 10 000

FIGURE 6 – Seuil = 5 000

FIGURE 7 – Seuil = 2 000

Les images ont visuellement peu de différences malgré le fait que les temps de traitement soient très différents. L'image avec le seuil de 100 000 pixel changeant de classe par itération , ne tourne en fait que pendant 2 itérations. L'image avec un seuil de 2 000 tourne pendant plus de 20 itérations.

Afin de mesurer plus finement les différences nous avons mesuré le PSNR entre l'image de départ et l'image issue de SLIC. Voici le graphe des valeurs de PSNR selon la valeur du seuil à $k = 5400$:

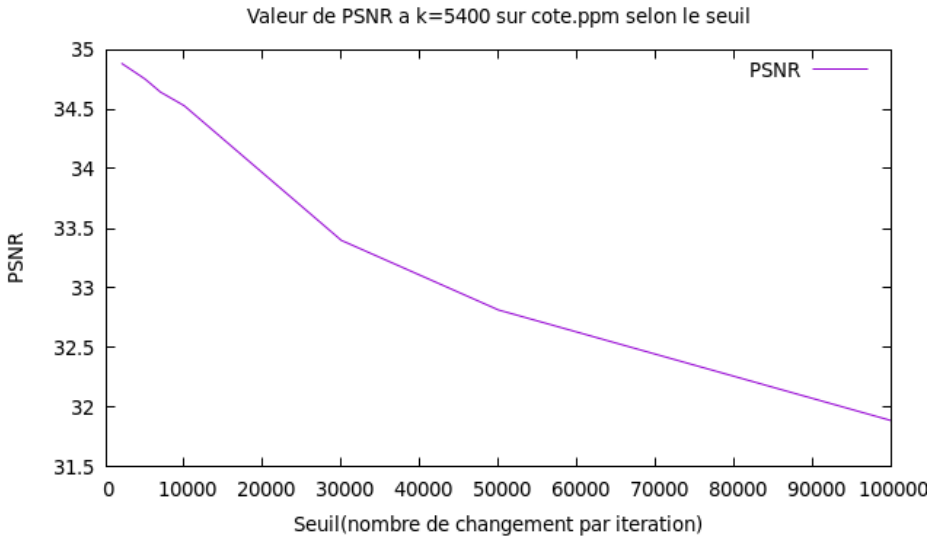


FIGURE 8 – Courbe PSNR selon seuil

Le PSNR atteint son maximum lorsque le seuil est au plus bas avec environ 35 dB et décroit jusqu'aux alentour de 31.5 dB pour un seuil de 100 000.

Nous avons aussi utilisé la mesure SSIM, qui permet de voir la similitude en structure dans l'image comparé au PSNR qui mesure la différence absolue pixel par pixel :

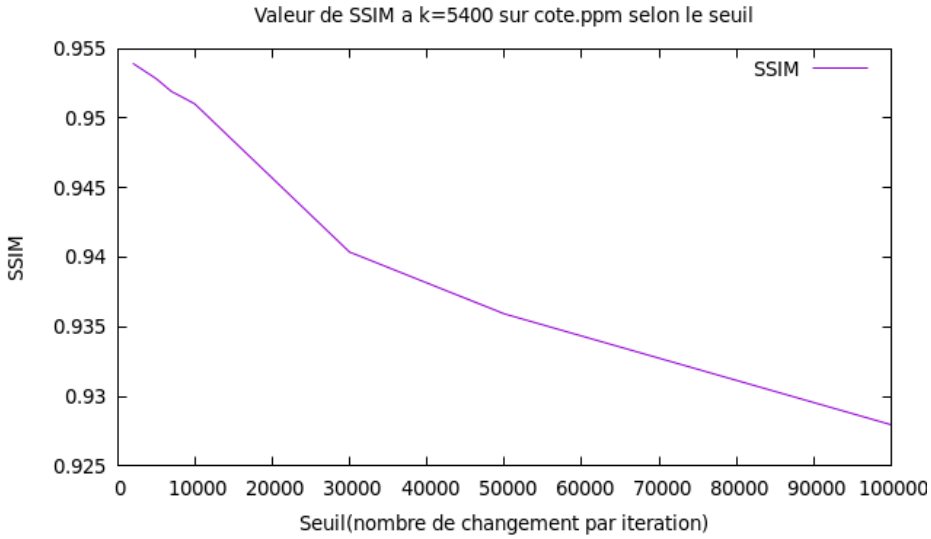


FIGURE 9 – Courbe SSIM selon seuil

Les 2 courbes sont extrememem similaires. Le SSIM, qui si il est à 1 indique une copie parfaite, est au mieux à 0.955 et pour un seuil de 100 000 descend en dessous de 0.93.

On peut donc se demander si il y a moyen de trouver le seuil optimal au préalable avant de lancer SLIC. En effet le gain de qualité en réduisant le seuil est assez négligeable et visuellement ne se voit presque pas. Commencer avec un seuil à un tier de la quantité totale de pixel semble être un bon compromis , sachant que dans notre cas nous avions 262144 pixels et que l'image faite avec un seuil de 100 000 n'avait pas une si mauvaise qualité que cela. Nous allons tester l'algorithme sur une image plus petite de taille 200 par 300 , soit 60 000 pixels :

FIGURE 10 – Image Pirate

Et voici les résultats avec SLIC avec k = 2 000 :

FIGURE 11 – Seuil = 20 000

FIGURE 12 – Seuil = 15 000

FIGURE 13 – Seuil = 10 000

FIGURE 14 – Seuil = 5 000

FIGURE 15 – Seuil = 2 000

FIGURE 16 – Seuil = 1 000

Des différences peuvent se voir, notamment au niveau des nuages. Nous allons faire la même analyse que précemment avec PSNR et SSIM :

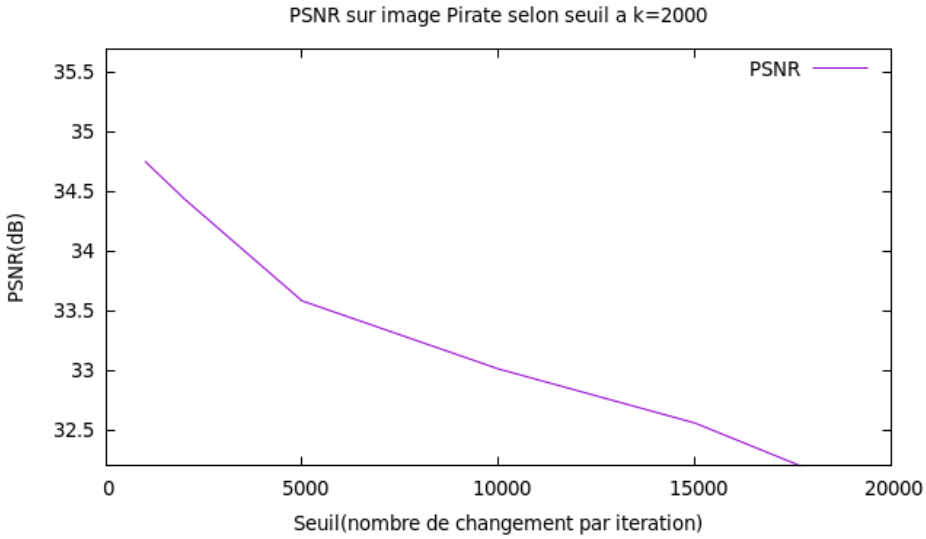


FIGURE 17 – Courbe PSNR selon seuil

Encore une fois le PSNR semble se rapprocher de 35 au mieux puis diminue. C'est intéressant de noter que la valeur maximum dans nos 2 exemples sont approximativement les mêmes.

Le SSIM :

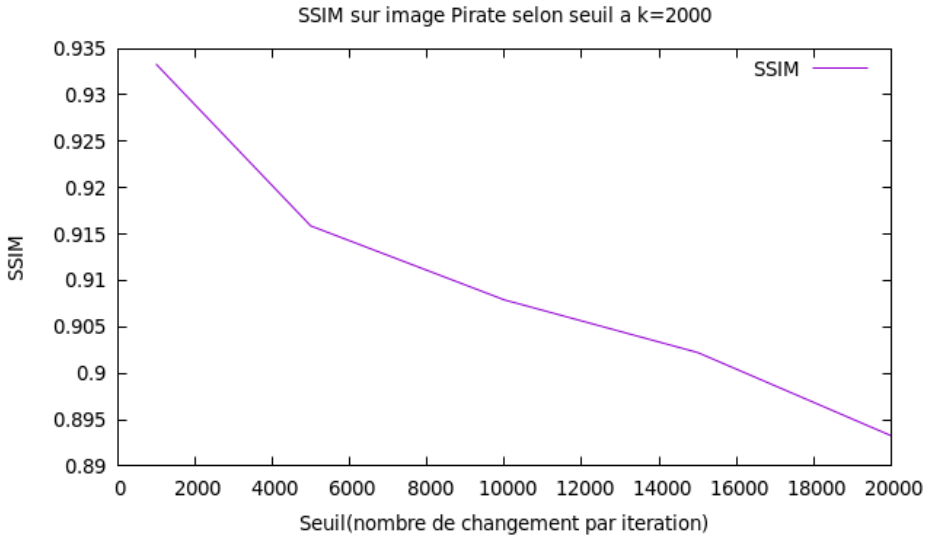
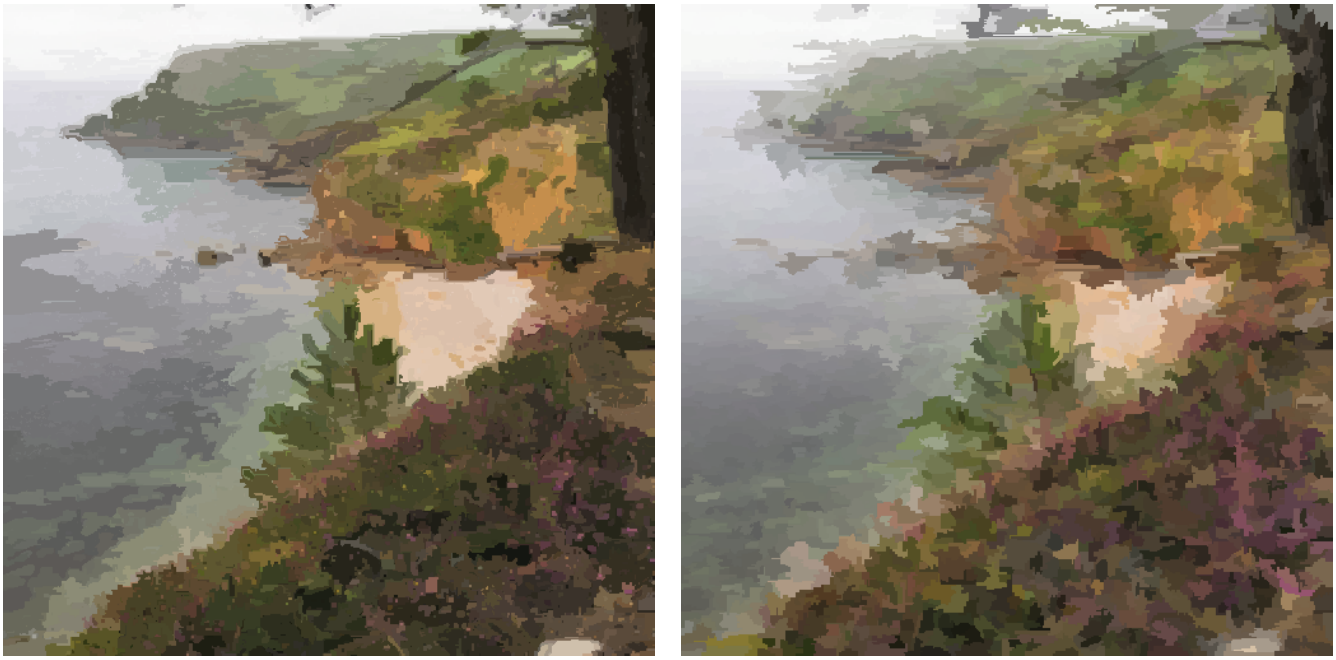


FIGURE 18 – Courbe SSIM selon seuil

La courbe SSIM ressemble encore une fois à la courbe PSNR mais la valeur maximum atteinte cette fois-ci est aux environ de 0.935.

3.3 Algorithme de Felzenszwalb

L'algorithme de Felzenszwalb dépend de deux paramètres qui déterminent le nombre et la répartition des superpixels dans l'image, k contrôlant la segmentation des composantes (segments) de l'image en fonction de leur taille, et N_{min} la taille minimale d'un segment. Ici, k possède un contrôle plus direct sur la manière donc la segmentation prend place, tandis que le second paramètre permet d'améliorer les résultats de cette segmentation afin de réduire la portion de superpixels plus petits, et donc de fragmenter l'image plus efficacement. Or, il n'est pas toujours évident de reconnaître le paramètre ayant la plus forte influence sur nos résultats. (Voir Figure 23)



(a) $k = 8000, N_{min} = 1$ (b) $k = 0, N_{min} = 30$

FIGURE 19 – Importance des paramètres sur la segmentation

En partant du principe que le PSNR de l'image segmentée doit être supérieur à $30dB$, on estime dans nos tests le nombre minimal de pixels par cluster $N_{min} = 3$, ce que nous approximons à 1 du fait de sa très faible représentation visuelle mais qui agit tout de même fortement sur le PSNR. Cela nous permet de faire uniquement varier k et d'en déterminer son influence sur le résultat final de l'image. Nous avons également inclus la mesure de la similarité structurel (SSIM) entre l'image d'origine et l'image segmentée qui donne un léger indicatif de la ressemblance entre les deux images.

(a) $k = 1000$
PSNR = 35,208 dB, SSIM = 0,939

(b) $k = 2000$
PSNR = 29,802 dB, SSIM = 0,851

(c) $k = 4000$
PSNR = 26,022 dB, SSIM = 0,732

FIGURE 20 – Influence de k dans la segmentation de l'image

Visuellement, la différence n'est pas extrême, mais elle représente tout de même une forte perte d'information pour des k plus elevés. Elle se remarque notamment à l'étude de la distortion (PSNR) et du SSIM en fonction de ce paramètre.

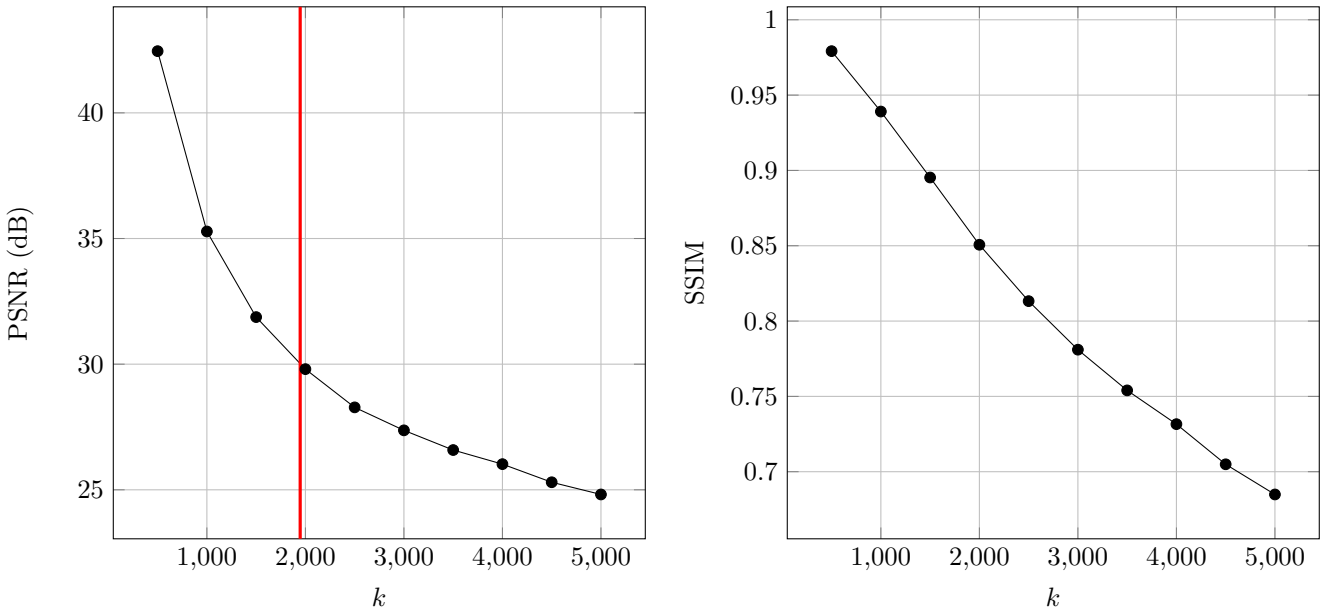
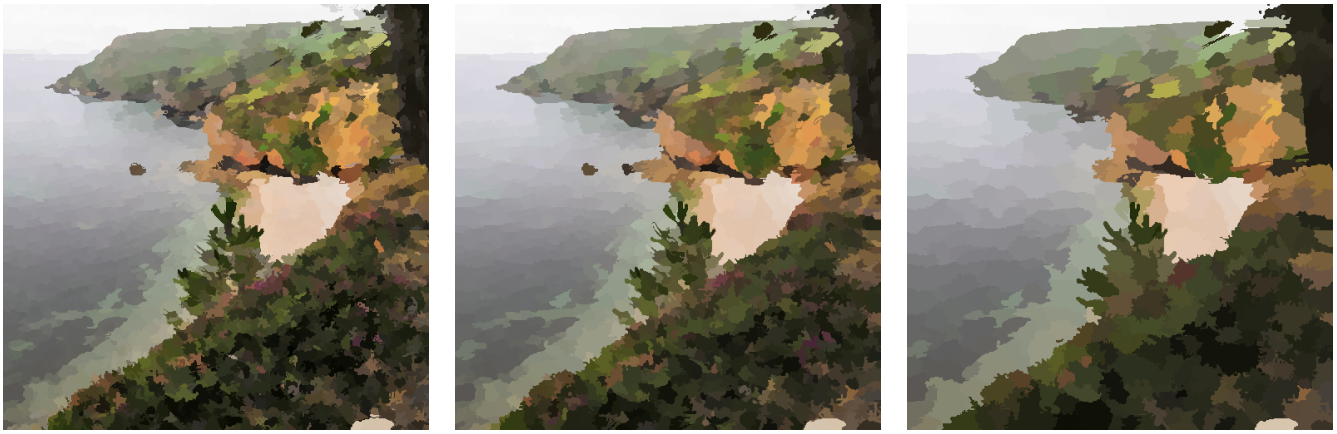


FIGURE 21 – Mesure du PSNR et du SSIM en fonction de k

Tandis que le PSNR décroît plus rapidement, le SSIM décroît de manière linéaire puisque la segmentation d'une image en superpixels assure que la qualité de la structure générale de l'image décroît visuellement de manière plus linéaire. On remarque tout de même que sur cette image, nous estimons $k \approx 1947$ segmentant l'image à un PSNR minimum de 30 dB.

3.4 Quick Shift

À nouveau, l'algorithme Quick Shift dépend de deux paramètres, la taille du noyau Gaussien k et l'écart-type de ce dernier σ. Il est clair cette fois-ci que k contrôle plus directement le nombre de superpixels présents dans l'image, tandis que σ contrôle notamment le taux de variation entre deux superpixels.



(a) $k = 4, \sigma = 1$
PSNR = 22,133 dB, SSIM = 0,639

(b) $k = 4, \sigma = 4$
PSNR = 22,837 dB, SSIM = 0,651

(c) $k = 8, \sigma = 4$
PSNR = 20,961 dB, SSIM = 0,567

FIGURE 22 – Influence de k dans la segmentation de l'image

Or, nous ne nous attendons pas à obtenir un fort PSNR en faisant varier k, car il influe fortement sur le PSNR de l'image, et uniquement des valeurs inférieures à $\sigma = 0,00694$ semblent être les seules possibilités afin d'obtenir au moins une qualité supérieure à 30 dB. Heureusement, la segmentation prend tout de même place et ne destructure pas entièrement l'image (SSIM de 0,905).

(a) Image segmentée (b) Image segmentée (+ zoom)

FIGURE 23 – Segmentation de l'image à un PSNR de 30 dB

Nous pouvons à nouveau visualiser les courbes de la distortion et du SSIM en fonction de σ, et notamment autour des écarts-types relativement bas.

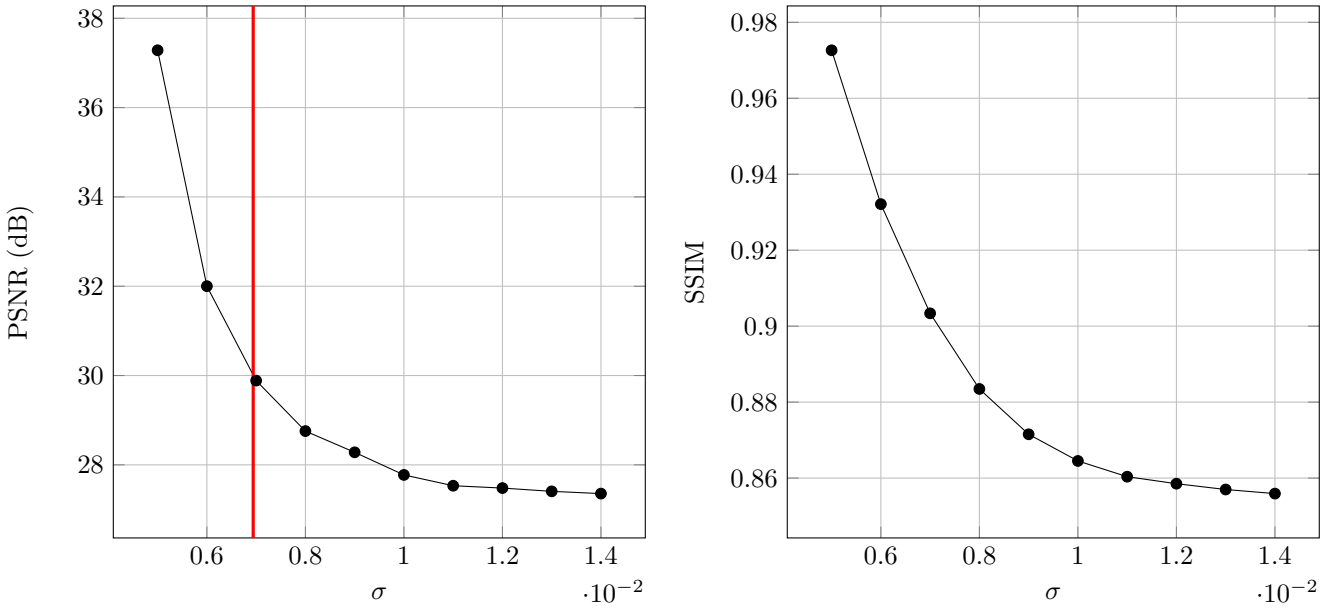


FIGURE 24 – Mesure du PSNR et du SSIM en fonction de σ

Il est possible de remarquer que pour des valeurs très basses de σ, la qualité de l'image augmente très vite, et ne décroît que très faiblement ensuite, pour enfin stagner autour des 27,35 dB, ce que nous pouvons également voir avec le SSIM, autour de 0,85.