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

24 mars 2024

1 Introduction

Cette semaine, nous avons implémenté trois méthodes de compression bénéficiant de la structure des superpixels d'une image segmentée, pour lesquelles nous avons analysé les résultats. Il s'agit notamment de la compression par palette, par codage prédictif suivi d'un codage par plage ainsi que de l'encodage des contours des superpixels.

2 Méthodes de compression

2.1 Compression par palette

La compression par utilisation de palette de couleur dans notre cas est très intuitive. En effet la palette de couleur est tout simplement l'ensemble des couleurs des superpixels. Nous utiliserons en guise de palette une carte ayant pour clé la couleur et un certain indice entier donné est retourné par la palette pour cette couleur. Ainsi on peut passer d'une image rgb à une image en niveau de gris permettant ainsi de gagner en compression. Prenons l'image suivante et appliquons lui notre algorithme SLIC avec 500 cluster et un seuil de 5 000 et sa version compressée :

(a) Image originale 1 Mo

(b) Image segmentée 348 ko

(c) Image compressée 69 ko

Dans cet exemple le taux de compression est donc de 14,49. Cependant l'image compressée ici ne possède pas la palette en entête. Si nous calculons le poids de rajout de l'entête nous devons ajouter 3 octets par superpixels et l'indice correspondant. Cela donne donc la formule suivante :

$$P_h = 24 \times s + i \times s$$

où P_h est le poids en bits de l'entête, s est le nombre de superpixels, et i est le poids en bits d'un indice.

Dans notre cas on utilise un int comme indice donc i vaut 32 bits, et on a une entête faisant 28 000 bits à rajouter, soit 3,5 Mo ce qui au final augmente la taille de l'image. La palette pourrait être intéressante si on utilisait

la même pour plusieurs images ou si l'on arrivait à la compresser. Il existe des algorithme tels que DEFLATE qui permettent de compresser des fichiers d'entête ayant des taux allant jusqu'à 5. Si l'on avait théoriquement ce taux nous aurions une entête de 700 ko et donc 769 ko sur l'image finale, donnant un taux de compression de 1,3 pour la méthode totale.

2.2 Compression par codage prédictif et RLE

Cette méthode de compression RLE se base sur la prédiction d'un pixel possédant dans notre cas la capacité à faire ressortir un grand nombre de valeurs nulles. L'algorithme RLE est tel qu'employé par JPEG, et chaque symbole correspond à une paire (c, l) où c représente la couleur du pixel non nul et l la taille de la chaîne de couleur nulle subséquente, encodée sur le plus petit nombre de bits permettant d'encoder la plus longue de ces chaînes, et cette valeur est définie dans l'entête des données compressées sur 6 bits. Le codage prédictif a été évalué au moyen de trois approches différentes afin d'en déterminer celle qui produit de meilleurs résultats : la moyenne du pixel haut et gauche, DPCM et MED. Sur la segmentation de Felzenszwalb-Hutterlocher à 30 dB, les résultats sont les suivants :

	Moyenne	DPCM	MED
Débit	14,949 bits/pixel	8,397 bits/pixel	6,789 bits/pixel
Taux de compression	1,605	2,858	3,535

D'après notre évaluation, MED produit de meilleurs résultats de compression, et possède également la capacité de limiter les erreurs d'arithmétique entière telles que produites par la première méthode en produit à cause de la division.

(a) Segmentation de Felzeszwalb à 30 dB

(b) Codage prédictif MED (+128)

FIGURE 2 – Codage prédictif d'une image segmentée

Ainsi, l'analyse du débit en fonction de la distorsion a été menée à l'aide de l'algorithme MED sur différentes segmentations de l'image côte.

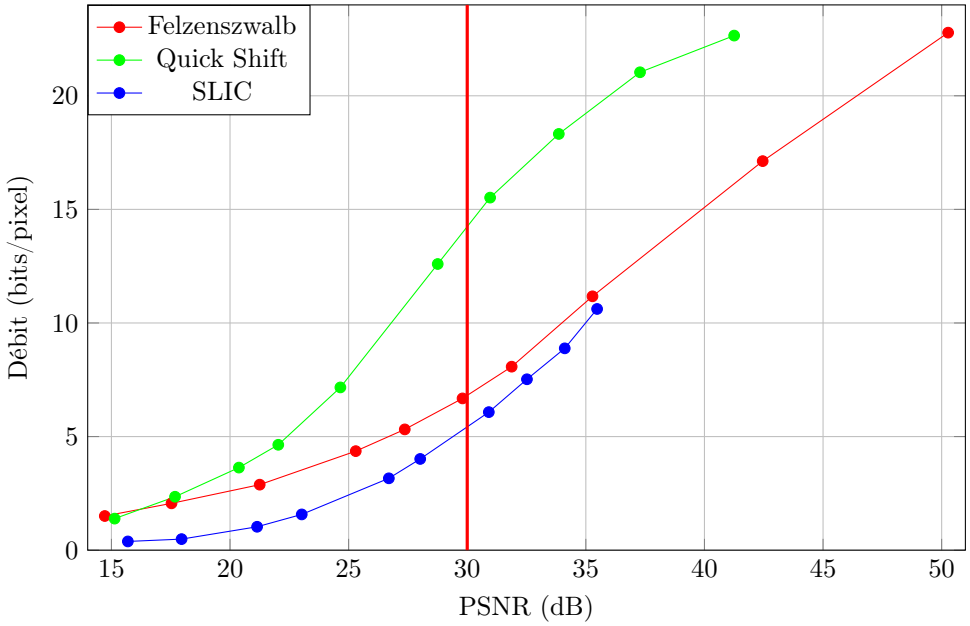
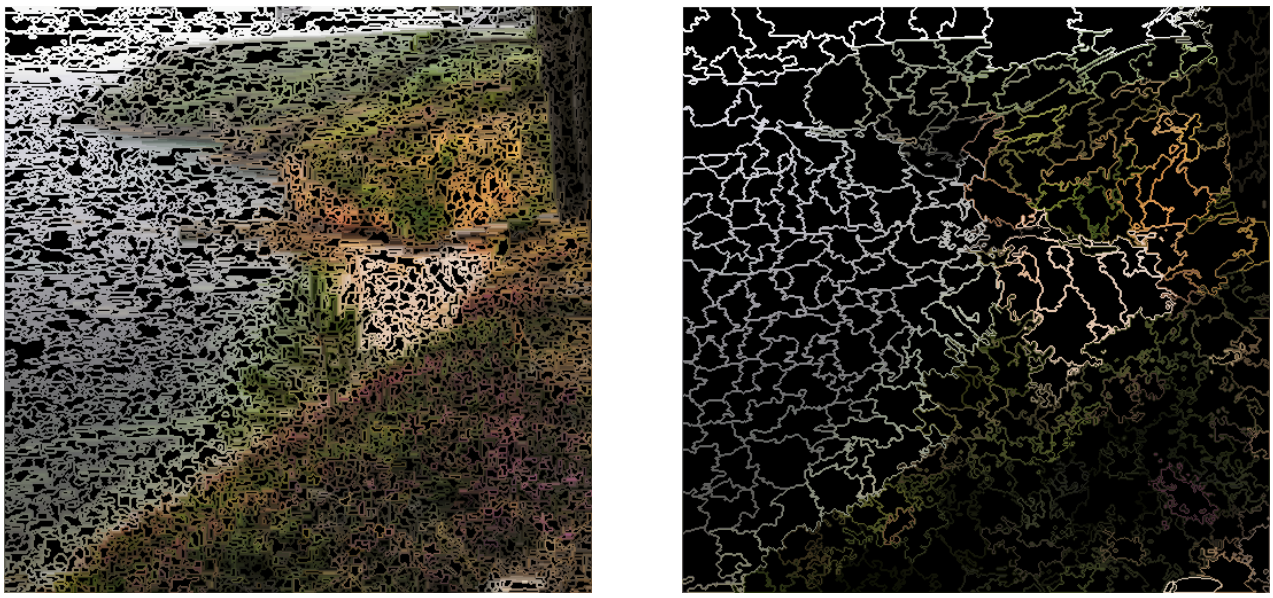


FIGURE 3 – Graphe débit/distorsion de la compression par prédiction et RLE

L'algorithme possède un très bon taux de compression pour des images aussi distordues que de très bonne qualité. De meilleurs résultats de compression sont obtenus par l'algorithme SLIC et par celui de Felzenszwalb, ce qui suggère que leur capacité à générer de plus gros clusters au maintient d'une bonne reconstruction permet de faciliter leur encodage par plage. Ce qui est par ailleurs très probablement en raison du calcul moyen des couleurs pour Felzenszwalb. En effet, avec cet algorithme, nous obtenons environ 2 octets par pixel au lieu de 3 rien qu'au-delà d'une distorsion de 40 dB. Il semblerait aussi que SLIC atteigne le débit de Felzenszwalb pour un même PSNR, mais il est compliqué de le déterminer car l'algorithme ne dépasse que très difficilement 36 dB sur l'image côte. En revanche, cet algorithme n'est pas parfait, car des hautes valeurs de PSNR conduisent l'algorithme RLE a considérer un grand nombre de valeurs non-nulles et des plages plus réduites, ce qui peut entraîner un taux de compression inférieur à 1.

2.3 Compression par contour de région

Le principe de cette segmentation de l'image est de construire des superpixels dont on estime que leur taille représente une région uniforme de l'image plus vaste que dans l'image originale. Cette idée nous permet d'en déduire intuitivement que chacune de ces régions peuvent simplement être délimitées par leur contour dans le but de compresser cette segmentation sans induire de perte d'information sur cette décomposition en superpixels. Il s'agit donc d'encoder chaque région par leur couleur et un ensemble de pixels qui forment les contours de la région, qui peut être remplie à la décompression par un simple algorithme de remplissage. Le fonctionnement est assez similaire à ce qui a été précédemment implémenté avec l'algorithme de Felzenszwalb et Hutterlocher où nous regroupons chacun des pixels dans leur composante respective selon un critère de similarité. Ici, ce critère est fort puisqu'il servire de critère de fusion des régions de même couleur. Dès que ces régions sont construites, seuls les pixels qui ne bordent pas le cluster, c'est à dire qui sont contenus entre quatre pixels de ce même cluster, sont ignorés, et donc le reste forment le contour du superpixel. Chaque région est alors encodée par leur couleur (24 bits) puis par l'indice de chacun des pixels de contour sur le nombre de bits minimum pour les représenter ($\lceil \log_2(w \times h) \rceil$ où w et h sont les dimensions de l'image).



(a) Felzenszwalb avec $k = 1000$ et $N_m in = 10$

(b) Quick Shift avec $\sigma = 0.1$ et $k = 12$

FIGURE 4 – Visualisation des contours

Le calcul du débit en fonction de la distorsion a été tracé sur l'image de côte, qui reste l'image de référence, sur nos trois algorithmes de segmentation.

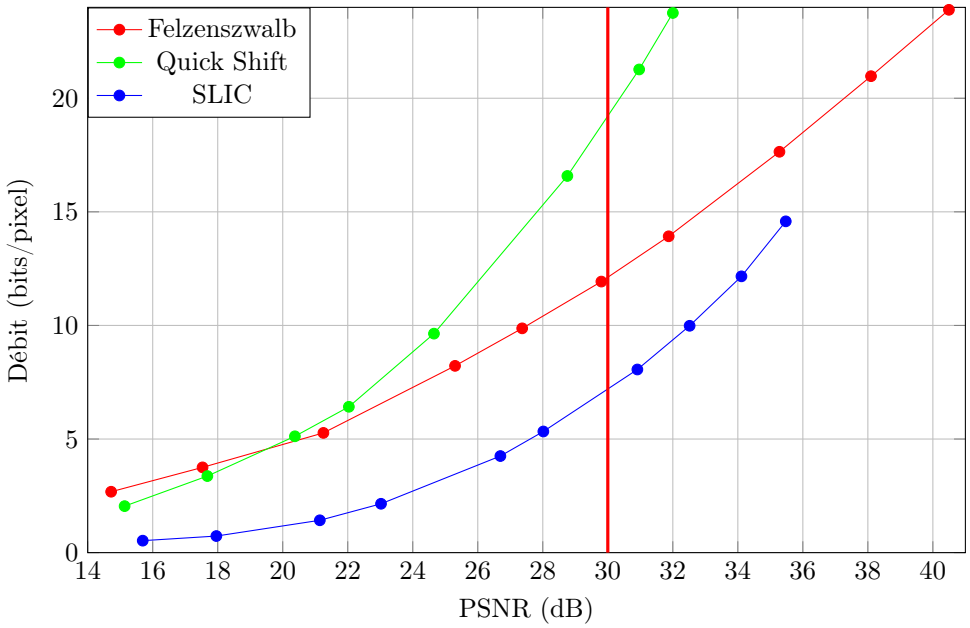


FIGURE 5 – Graphe débit/distorsion de la compression par contour

Cet algorithme ne permet pas mieux que les autres de compresser l'image afin d'obtenir un bon PSNR puisque le débit augmente fortement en fonction de la distorsion, et ne compresse que très peu les reconstructions de bonne qualité. Couplée avec Quick Shift, cette méthode n'est pas complètement efficace pour obtenir un bon taux de compression à un débit supérieur à 30 dB. Or pour des paramètres Quick Shift k plus elevés, cette approche permet d'obtenir un meilleur débit que l'algorithme de Felzenszwalb, au coût d'un PSNR médiocre. En revanche, cette méthode de compression sur SLIC propose de très bons résultats, qui suit à peu près la même tendance obtenue que par l'approche prédictive, mais ne permettant toujours pas d'obtenir des résultats au-delà de 36 dB sans sacrifier un grand nombre de ressources computationnelles et de temps.

Le problème est que cet algorithme aura tendance a stocker non seulement la couleur des régions à très faible pixels, mais l'indice des pixels eux-mêmes, ce qui peut en théorie atteindre un débit de $24 + l$ bits/pixel où l est

le nombre de bits minimum pour encoder l'indice de tous les pixels de l'image. En outre, une amélioration de cet algorithme est possible, ne permettant pas entièrement de pallier ce problème mais d'obtenir de meilleurs résultats pouvant compenser cette erreur. Il s'agit ainsi de réduire le nombre de pixels utilisés pour encoder un contour, puisqu'un segment droit ou diagonal peut être encodé seulement par un premier point et un second plutôt qu'une succession de pixels adjacents, et pour gérer les superpixels contenus dans (ou en grande majorité dans) un autre superpixel, il serait possible de trier ces régions par ordre de dessin, permettant de représenter chaque superpixel par un polygone ne comportant pas de trou dans sa structure. Par ailleurs, nous pouvons également omettre les pixels aux bords de l'image.