Compte Rendu Semaine 4 - Projet IMAGE

HAI809I - Master 1 Imagine - Université de Montpellier

Laporte Laëtitia - Chaudillon Luca

24 Mars 2024

Résumé

L'objectif de ce projet est de compresser des images réelles (issues d'APN) à partir d'une approche de superpixels.

Table des matières

1 Introduction

1.1 Informations supplémentaires

Les codes produits lors de ce projet ainsi que les comptes rendus et la base de données d'images que nous allons utiliser sont retrouvables sur un répertoire github à l'adresse suivante : https://github.com/cygne1110/projet-IMAGE-M1

2 Suite de l'étude de l'algorithme de Felzenszwalb

2.1 Rappels

Le but de cette approche est d'avoir des éléments plus similaires dans une même région et plus différents s'ils appartiennent à des régions différentes. Cela signifie que les arêtes (relations) entre deux nœuds d'une même région devraient avoir un poids faible (le poids d'une arête correspond à la dissimilarité entre les deux pixels de celle-ci), et les arêtes entre les nœuds de différentes composantes devraient avoir un poids plus fort.

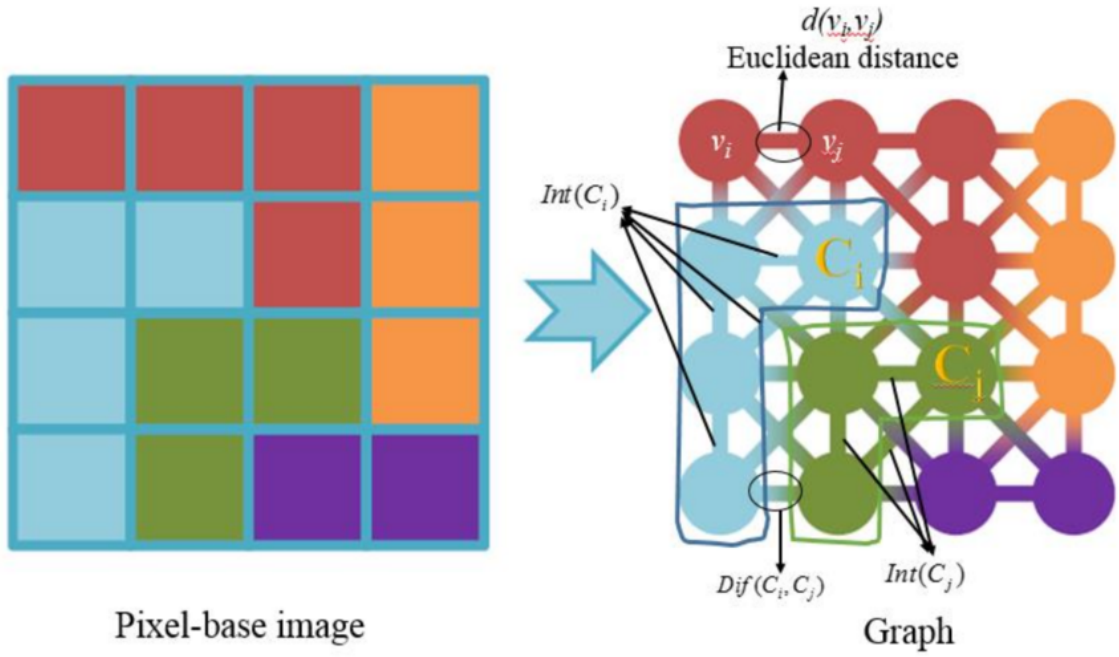


FIGURE 1 – Schéma de la segmentation d'image basée sur un graphe

2.2 Fusion des régions

Initialement, chaque pixel est considéré comme une région distincte. L'algorithme procède à la fusion des régions de manière itérative. À chaque itération, il sélectionne la paire de régions voisines ayant la plus faible dissimilarité selon un **critère de fusion** défini. La dissimilarité entre deux régions est calculée en fonction de la similarité entre les pixels des deux régions et de leur proximité spatiale. Les régions sont fusionnées si leur dissimilarité est inférieure à un seuil prédéfini. Cela garantit que les régions fusionnées sont suffisamment similaires tout en contrôlant la taille des régions produites.

2.2.1 Critère de fusion

Le critère de fusion est généralement basé sur une mesure de la similarité entre deux régions. Cette mesure peut combiner plusieurs facteurs tels que la différence de couleur, la texture, la distance spatiale, etc. Les régions ayant une faible différence de couleur et une proximité spatiale sont plus susceptibles d'être fusionnées. Le choix du critère de fusion et des paramètres associés peut influencer le résultat de la segmentation.

Jusqu'à présent, nous avons exploré un aspect du critère de fusion, à savoir la proximité spatiale, représentée par la formule d'affinité entre deux pixels. Voici un rappel de la formule d'affinité spatiale entre deux pixels p1 et p2 :

$$spatialAffinity(p1, p2) = exp(-\frac{distance(p1, p2)^2}{2\sigma^2})$$

Maintenant, nous allons aborder deux autres facteurs importants du critère de fusion : la couleur et la texture.

Le facteur couleur est calculé de la manière suivante :

$$colorAffinity(c1, c2) = exp(-\frac{colorDiff(c1, c2)^2}{2\sigma^2})$$

Où :

- c1 et c2 sont les vecteurs de couleur moyens des points étudiés (ou régions étudiées)
- sigma est un paramètre de contrôle pour la similarité de couleur
- $$colorDiff(c1, c2) = \sqrt{c1 - c2}^2$$

Le facteur texture est calculé de la manière suivante :

$$textureAffinity(t1, t2) = exp(-\frac{textureDiff(t1, t2)^2}{2\sigma^2})$$

Où :

- c1 et c2 sont les caractéristiques de texture moyennes des points étudiés (ou régions étudiées)
- sigma est un paramètre de contrôle pour la similarité de texture
- $$textureDiff(t1, t2) = \sqrt{t1 - t2}^2$$

2.2.2 Combiner les facteurs du critère de fusion

Pour réunir ces facteurs, nous pourrions utiliser une approche pondérée où chaque critère contribue à la mesure globale de similarité entre deux régions.

Tout d'abord, il faut attribuer des poids à chaque facteur en fonction de son importance relative dans la segmentation souhaitée. Par exemple, nous pourrions classer les facteurs dans l'ordre du plus important ou moins important : couleur, distance, texture. L'ordre n'est pas encore fixe.

Ensuite, il faut s'assurer que les différentes mesures de similarité (différence de couleur, texture, distance) sont normalisées pour avoir la même échelle. Cela garantit que les poids attribués aux critères ont un sens comparable.

Troisième étape, on peut combiner ces facteurs en utilisant la formule suivante qui mesure la similarité combinée entre deux régions R1 et R2 :

$$simCombine(R1, R2) = Wc * colorAffinity(R1, R2) + Wt * textureAffinity(R1, R2) + Ws * spatialAffinity(R1, R2)$$

Ici, Wc, Wt, Ws correspondent aux poids attribués à chaque facteur respectif.

Enfin, une fois que l'on a calculé la similarité combinée entre deux régions, on peut utiliser cette mesure pour décider si les régions doivent être fusionnées ou non. On pourra définir un seuil de similarité en dessous duquel les régions ne seront pas fusionnées, et au-dessus duquel elles seront fusionnées.

Voici une comparaison entre le résultat attendu de l'algorithme de Felzenszwalb et celui de l'algorithme SLIC :

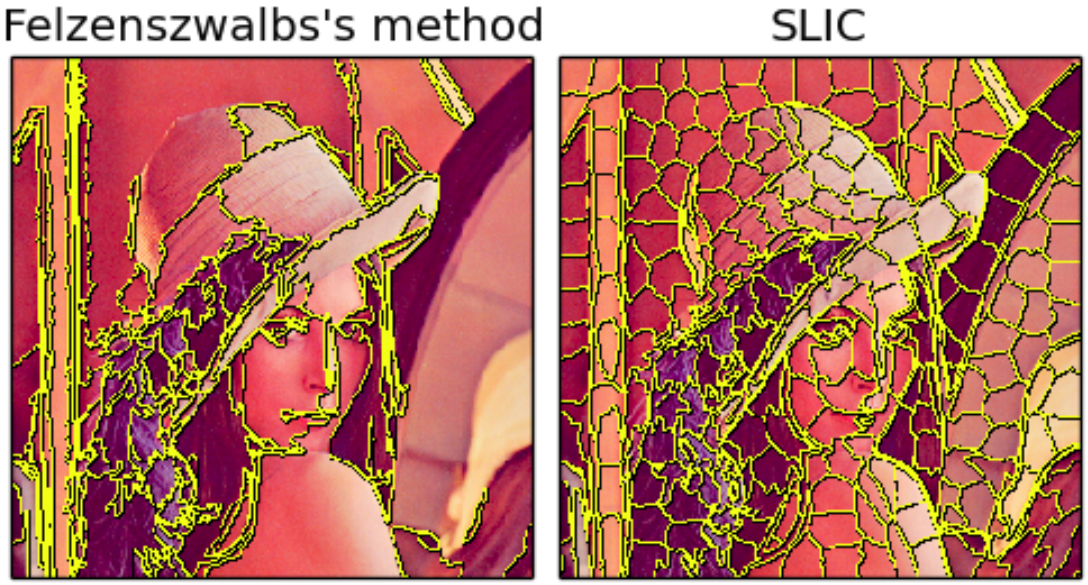


FIGURE 2 – Comparaison Felzenszwalb et SLIC

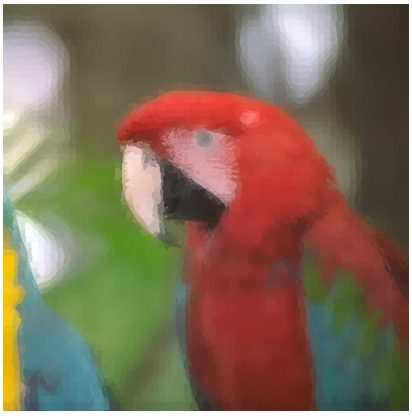
3 Résultats de l'implémentation SLIC

Depuis la semaine dernière, nous avons corrigé quelques erreurs dans notre implémentation SLIC, notamment il est maintenant possible de choisir un nombre de superpixels entre 1 et le nombre de pixels de l'image. Cependant, nous avons observé des comportements étranges lorsque que l'on s'approche de ces extrêmes, ainsi qu'une baisse non négligeable de la mesure du PSNR lorsque l'on utilise un nombre de superpixels pour segmenter qui n'est pas puissance de 4 pour une image 512 par 512.

Pour l'instant, nous n'appliquons pas d'algorithme de compression, juste un algorithme très simple qui affecte à un pixel la valeur moyenne du superpixels dans lequel il se trouve, et par la suite, nous pourrons stocker la valeur moyenne de tous ces superpixels sous forme de palette, et affecter à chaque pixel un indice correspondant à la couleur dans la palette.

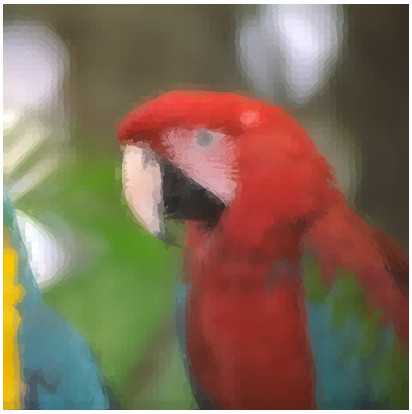
Voici quelques résultats (pour rappel, l'algorithme SLIC prend en entrée un paramètre m représentant la distance maximale entre deux couleurs, qui est un poids lors de la mesure de distance entre deux pixels. Une distance maximale proche de 1 donnera des superpixels amorphes, tandis que des distances maximales supérieures à 30 donneront des superpixels compacts) :

(a) *Superpixels* : 1024, $m = 1$

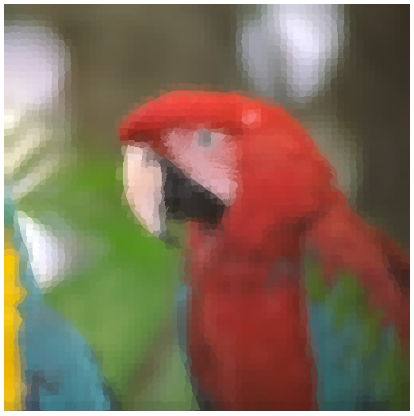


(b) *Superpixels* : 1024, $m = 5$

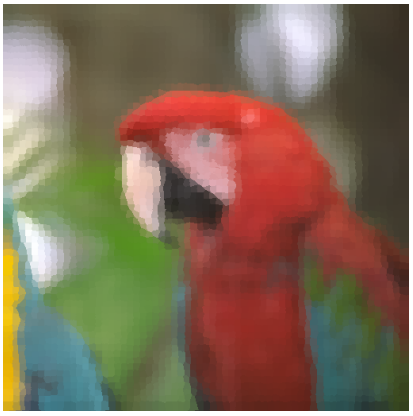
(c) *Superpixels* : 1024, $m = 40$



(d) *Superpixels* : 4096, $m = 5$



(a) *Superpixels* : 4096, $m = 20$



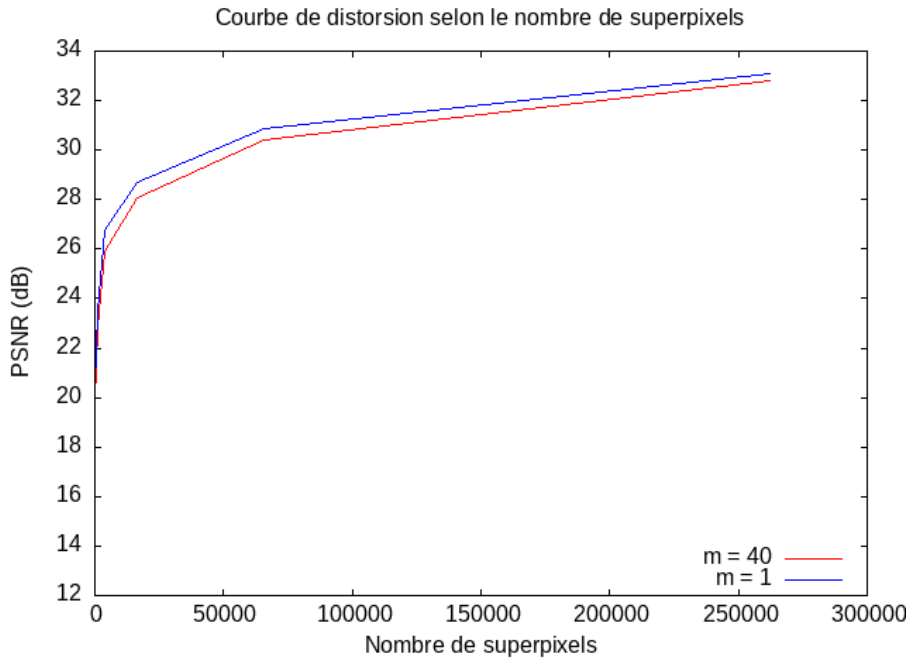
(b) *Superpixels* : 4096, $m = 40$

(c) *Superpixels* : 65536, $m = 1$

(d) *Superpixels* : 65536, $m = 40$

On peut observer que pour 65536 superpixels, on obtient une image proche de l'original à l'œil nu. L'image étant de taille 512 par 512, cela fait un superpixel pour 16 pixels, un facteur non négligeable pour la compression. Nous n'avons pas encore fait de mesures de taille de fichier, mais en théorie, si l'on stocke 512 par 512 pixels sur 24 bits, la taille des informations sur ces pixels est de 6291456 bits. Si l'on stocke 65536 couleurs dans une palette, on peut affecter à chaque pixel une valeur codée sur 16 bits ($2^{16} = 65536$), et stocker 65536 couleurs sur 24 bits, la taille des informations sur ces pixels est de 5767168 bits. Cela nous fait déjà un **taux de compression de 1.09**, ce qui est peu, mais qui a du potentiel.

Nous avons réalisé une mesure de distorsion selon le nombre de superpixels et la distance maximale entre deux couleurs, voici nos résultats :



Nous pouvons observer qu'affecter la valeur moyenne d'un superpixel à tous les pixels qui le compose engendre des pertes d'information non négligeables, mais que l'on arrive quand même à avoir des images de moyenne qualité (au-dessus de 30 dB). On observe aussi que la distance maximale entre deux couleurs influence peu la distorsion, on observe qu'elle ne décale que la courbe de quelques dB. Il est donc préférable d'utiliser, pour SLIC classique, des superpixels moins compacts pour compresser l'image. Nous pouvons même hypothéser que l'algorithme de Felzenszwalb permettra de meilleurs résultats sur ce point-là, car il permet de faire des groupes de pixel amorphes plus grands que les superpixels de l'algorithme SLIC. Nous avons aussi réfléchi à un moyen d'améliorer SLIC en dehors des variantes déjà explicitées dans les papiers précédents, où nous pourrions fusionner les superpixels qui sont proches spatialement et colorimétriquement.

4 Prévisions pour la semaine prochaine

Nous prévoyons de continuer l'implémentation d'un algorithme de compression palette sur SLIC. Nous voulons aussi essayer d'améliorer notre algorithme SLIC en fusionnant les superpixels qui se ressemblent.

5 Références

5.1 Algorithme de Felzenszwalb

— Vidéo youtube : Graph Based Segmentation | Image Segmentation
— https://davidstutz.de/implementation-of-felzenswalb-and-huttenlochers-graph-based-image-segmentation/
— https://www.iro.umontreal.ca/ mignotte/ThesisReports/RachidHedjamMSc.pdf

5.2 Implémentation

— https://www.iro.umontreal.ca/ mignotte/IFT6150/Articles/SLIC_Superpixels.pdf
— https://infoscience.epfl.ch/record/177415