Compte Rendu #1 : Etat de l'art et choix de l'algorithme

Mathys CHAMPENOIS, Cesar CHARBEY, Mickael CZOLACZ - Super Pixels

Dépôt GitHub : `https://github.com/Mathys3114/ProjetCCMTI`

1 Introduction et Choix de l'Algorithme

Suite à nos recherches, nous avons décidé de baser notre projet de compression sur l'algorithme **SLIC (Simple Linear Iterative Clustering)**. C'est la méthode de référence, car elle est rapide, simple à coder et efficace en mémoire.

Plus précisément, nous allons suivre l'implémentation proposée par R. Gay, P. Monasse et al. (2022). La grande différence avec le SLIC classique est que **cette version travaille directement dans l'espace colorimétrique RGB** au lieu de CIELAB. Cela simplifiera grandement notre code tout en gardant d'excellentes performances.

L'idée fondamentale est de travailler dans un espace à 5 dimensions combinant position et couleur :

$$(x, y, r, g, b)$$

2 L'Algorithme SLIC étape par étape

Notre implémentation s'articulera autour des étapes suivantes, telles que décrites dans le papier de référence :

1. **Initialisation** : On choisit le nombre de superpixels souhaité K. On place K centres sur une grille régulière, espacés de S pixels, avec $S = \lfloor \sqrt{N/K} \rfloor$ (N = nombre total de pixels).
2. **Perturbation** : On déplace légèrement les centres vers la position où le gradient est le plus faible. *Objectif :* Éviter de placer un centre de superpixel exactement sur le contour d'un objet de l'image.
3. **Affectation** : Pour chaque pixel, on cherche à quel centre il appartient.

   L'optimisation K-Means Bilatéral

   Dans un K-Means classique, on compare chaque pixel à *tous* les centres. Dans SLIC, on compare chaque pixel **uniquement** aux centres situés dans une fenêtre locale de $2S \times 2S$ autour de lui. C'est ce qui rend la complexité de l'algorithme linéaire ($O(N)$).
4. **Mise à jour et Répétition** : On recalcule la position spatiale et la couleur moyenne (x, y, r, g, b) de chaque centre en fonction des pixels qui lui ont été affectés.
5. **Post-traitement** : C'est le point fort du papier choisi. SLIC peut générer des pixels isolés. Nous devrons implémenter un algorithme de post-traitement pour identifier le composant connexe principal de chaque superpixel, transformer les pixels restants en "orphelins", puis réaffecter ces orphelins aux superpixels voisins.

3 La Formule de Distance

Pour affecter un pixel à un centre, nous combinons la distance couleur (d_c) et la distance spatiale (d_s). La distance totale D s'exprime ainsi :

$$D = \sqrt{d_c^2 + \left(\frac{m}{S}\right)^2 d_s^2} \tag{1}$$

Avec :
— $d_c = \sqrt{(r_i - r_k)^2 + (g_i - g_k)^2 + (b_i - b_k)^2}$ (Distance couleur RGB).
— $d_s = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}$ (Distance spatiale).
— S : Le pas de la grille de base.
— m **: Le paramètre de compacité**.

Le paramètre m (Compacité)

C'est le paramètre ajustable de notre future interface :
— Si m est grand : La distance spatiale domine. Les superpixels seront très réguliers et compacts (carrés/hexagonaux).
— Si m est petit : La couleur domine. Les superpixels colleront parfaitement aux contours des objets, quitte à avoir des formes moins régulières.

4 Pistes pour la Phase de Compression

Une fois l'algorithme SLIC fonctionnel, le vrai défi du projet sera la compression. Nous avons dégagé trois pistes de réflexion :
— **Couleur unique par région :** Au lieu de stocker tous les pixels d'une zone (ex : le ciel), on ne stockera qu'une seule valeur RGB : la moyenne du superpixel.
— **Compression de la carte des labels :** Il faudra sauvegarder la carte indiquant à quel superpixel appartient chaque coordonnée. Les zones étant fortement contiguës, un algorithme de compression sans perte comme le RLE (Run-Length Encoding) devrait être très efficace.
— **Sous-échantillonnage spatiale :** Ne sauvegarder que les coordonnées des centres et leurs couleurs, puis tenter de reconstruire l'image par interpolation.

5 Plan pour la semaine prochaine

L'objectif à court terme est de finaliser le coeur de l'algorithme :
— Mettre en place la structure du code C++.
— Implémenter la création de la grille initiale et le déplacement par le gradient.
— Coder la boucle principale du K-Means bilatéral (calcul des distances RGB/spatiales et mise à jour des centres).
— *Résultat attendu :* Une version console de l'algorithme capable de traiter une image et de générer une première carte de superpixels (même sans l'étape de connectivité).

6 Plan pour les semaines à suivre

Une fois le noyau dur de l'algorithme fonctionnel, nous nous concentrerons sur la qualité et la compression :
— **Semaine +2 :** Implémenter les algorithmes de connectivité (gestion des composants connexes et réaffectation des pixels orphelins).

— **Semaine +3 :** Obtenir nos premiers résultats visuels propres et mettre en place une fonction d'évaluation (calcul du PSNR entre l'image reconstruite et l'originale).

7 Références

— **Article principal (SLIC RGB & Connectivité)** : R. Gay, J. Lecoutre, N. Menouret, A. Morillon, P. Monasse. *Bilateral K-Means for Superpixel Computation (the SLIC Method)*. Image Processing On Line, 11 (2022).
— **Benchmark des algorithmes** : David Stutz et al., *Superpixels : An evaluation of the state-of-the-art*, Computer Vision and Image Understanding (2018).