

# Compte Rendu Semaine 3 - Projet IMAGE

## HAI809I - Master 1 Imagine - Université de Montpellier

Laporte Laëtitia - Chaudillon Luca

17 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

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Informations supplémentaires . . . . .	2
<b>2</b>	<b>Étude de l'algorithme de Felzenszwalb</b>	<b>2</b>
2.1	Segmentation d'image basée sur un graphe . . . . .	2
2.2	Mesure de l'affinité . . . . .	2
<b>3</b>	<b>Implémentation de l'algorithme SLIC</b>	<b>3</b>
3.1	Résultats . . . . .	3
<b>4</b>	<b>Prévisions pour la semaine prochaine</b>	<b>5</b>
<b>5</b>	<b>Références</b>	<b>5</b>
5.1	Algorithme de Felzenszwalb . . . . .	5
5.2	Implémentation . . . . .	5

# 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 Étude de l'algorithme de Felzenszwalb

## 2.1 Segmentation d'image basée sur un graphe

L'algorithme de Felzenszwalb utilise la segmentation d'image basée sur un graphe. Cette technique de traitement d'image consiste à représenter l'image sous forme de graphe, où les pixels sont des noeuds et les relations entre ces pixels sont des arêtes.

On note le graphe  $G$  d'une image de la façon suivante :

$$G = (V, E)$$

Où  $V$  représente les sommets (pixels) et  $E$  les arêtes (relations) de l'image.

Chaque arête est pondérée par affinité ou par similitude entre ses deux sommets. Ce qui permet de regrouper les pixels en régions, de manière à ce que les différences entre les pixels voisins et appartenant à des régions distinctes soient supérieures aux différences entre les pixels appartenant à une même région. Voici un schéma qui illustre nos propos.

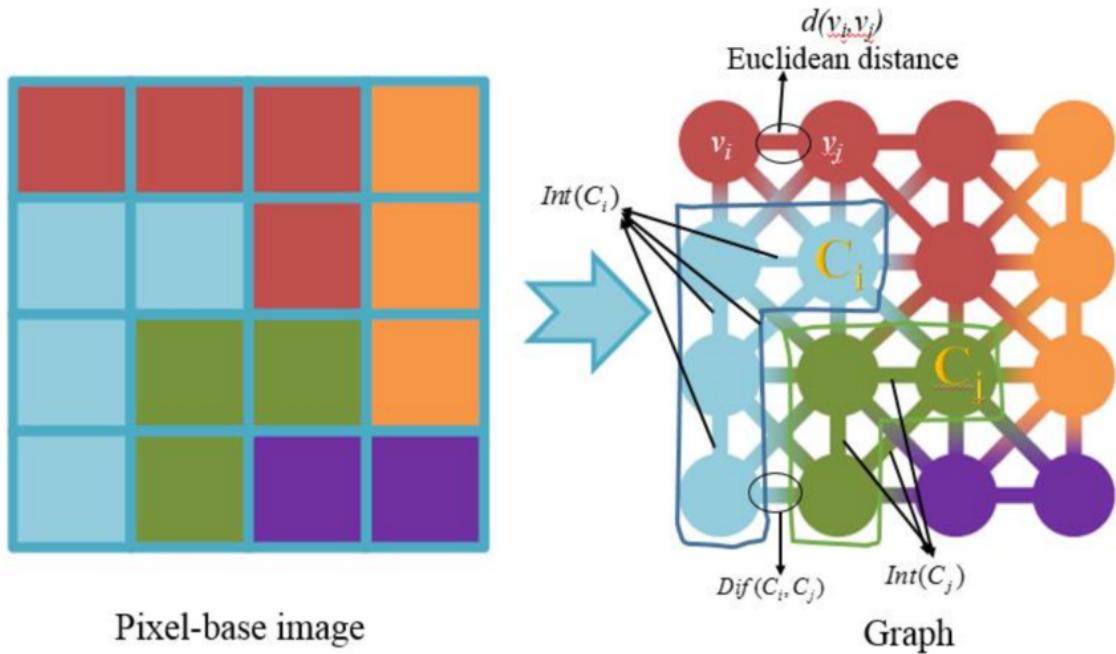


FIGURE 1 – Schéma de la segmentation d'image basée sur un graphe

## 2.2 Mesure de l'affinité

La formule, qui est couramment utilisée pour calculer l'affinité entre deux pixels, se base sur la similitude de couleur entre ces pixels. Cette similitude peut être calculée à l'aide de la distance

euclidienne dans un espace colorimétrique.

La formule de l'affinité entre deux pixels  $p_1$  et  $p_2$  peut être exprimée de la manière suivante :

$$affinity(p_1, p_2) = \exp\left(-\frac{distance(p_1, p_2)^2}{2\sigma^2}\right)$$

Où sigma est un paramètre de normalisation qui contrôle l'ampleur de l'affinité. Une valeur plus grande de sigma augmente la tolérance à la différence entre les pixels, tandis qu'une valeur plus petite la diminue.

### 3 Implémentation de l'algorithme SLIC

Nous avons implémenté un programme terminal qui permet de segmenter une image en superpixel, et qui crée une nouvelle image permettant de visualiser cette segmentation. Pour l'instant, le programme est assez rudimentaire, il est conseillé d'utiliser une image couleur (au format .ppm), de taille 512 par 512, et de choisir un nombre de superpixel qui est une puissance de 4. Il se peut qu'il y ait des bugs que nous n'avons pas rencontré lors des petits tests que nous avons effectué. L'algorithme implémenté prend trois paramètres en entrée : une image, le nombre de super pixels

souhaité et la distance maximale entre deux couleurs. Ce dernier paramètre permet d'influencer la compacité des superpixels, par exemple en mettant ce paramètre à 1, on obtient des superpixels amorphes, tandis qu'en le mettant à 40, on obtient des superpixels compacts et plutôt carrés.

#### 3.1 Résultats

L'image utilisée ci-dessous provient des images mises à disposition pour les TP de Codage et Compression, et Analyse et Traitement des Images ([Source](#)).

Pour des raisons pratiques, le nombre de superpixels sera appelé  $K$  et la distance maximale entre deux couleurs sera appelé  $m$ .

Voici maintenant quelques résultats obtenus :



(a)  $K = 256, m = 20$

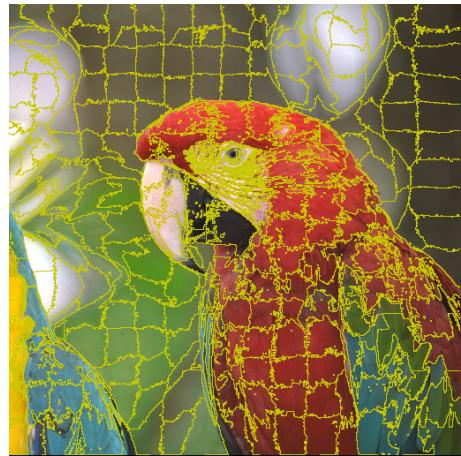


(b)  $K = 256, m = 30$

FIGURE 2 – 256 superpixels, avec une compacité suffisante

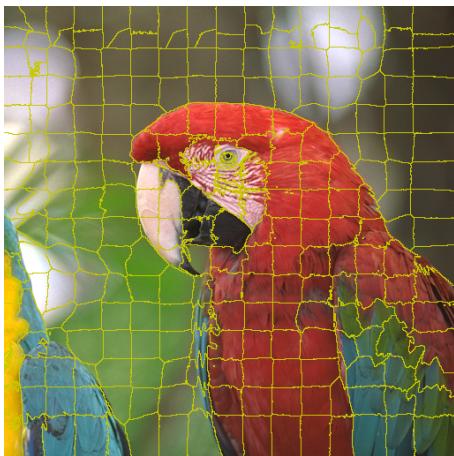


(a)  $K = 256, m = 1$

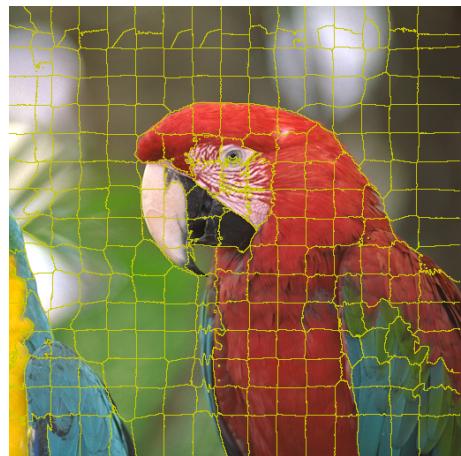


(b)  $K = 256, m = 10$

FIGURE 3 – 256 superpixels, avec une compacité faible



(a)  $K = 256, m = 40$



(b)  $K = 256, m = 50$

FIGURE 4 – 256 superpixels, avec une compacité forte

On peut observer que la distance maximale entre deux couleurs influence beaucoup la compacité et l'efficacité de la segmentation, une distance maximale trop forte maximise la compacité aux dépens d'une segmentation correcte, et une distance trop faible donne des superpixels amorphes, comme l'on pourrait avoir avec l'algorithme de Felzenszwalb. On verra plus tard qu'il existe des variantes de l'algorithme SLIC qui permettent de calculer la distance maximale entre deux couleurs optimale selon l'image, ou qui utilise de meilleures mesures de distance.

Maintenant, voyons ce qu'il se passe lorsque l'on réduit le nombre de superpixels :

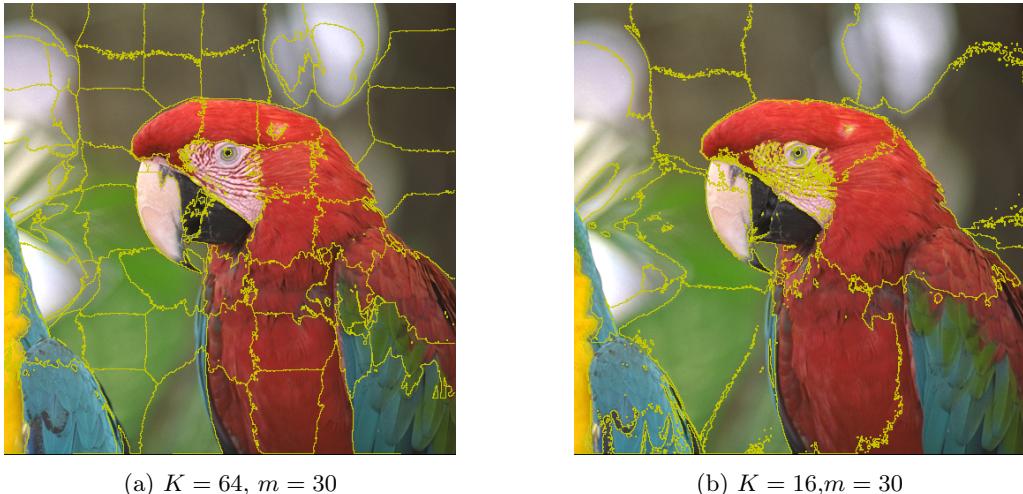


FIGURE 5 – Différents nombres de superpixels

## 4 Prévisions pour la semaine prochaine

Pour la semaine prochaine, nous prévoyons de trouver un algorithme de compression qui pourrait exploiter la segmentation en superpixels, et de l'implémenter. Nous prévoyons également de continuer l'état de l'art en parallèle de l'implémentation (notamment de continuer l'explication de l'algorithme de Felzenszwalb).

## 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://www.iro.umontreal.ca/~mignotte/IFT6150/Articles/SLIC_Superpixels.pdf)
- <https://infoscience.epfl.ch/record/177415>