2022-2023

INFORMATIQUE GRAPHIQUE ET IMAGE

LARIBI Iliesse P1911241 – VINCENT Yann P1906701

Table des matières

Introduction 2
Split and merge 2
Split 2
Merge 3
Nos résultats et implémentations 3
Nos implémentations 3
Les critères 3
Affichage : 4
Résultats 4
Conclusion 6
Lien git 6

1- Image représentant le fonctionnement de la division d'une région 3
2 - Image source 4
3 - Split sans fusion 4
4 - Split sans fusion avec l'affichage des bordures 4
5 - SplitandMerge 5
6 - SplitandMerge avec l'affichage des bordures 5
7 - SplitandMerge avec l'affichage des bordures et les régions coloriés aléatoirement 5

Introduction

Explication générale de segmentation d'une image.

La segmentation d'une image est un processus de partition d'une image afin de regrouper les pixels selon un critère d'homogénéité. Nous allons voir ici comment nous avons réalisé cela en utilisant l'algorithme « Split and merge ».

L'algorithme « Split and merge » est une des principales techniques de traitement d'image pour segmenter une image. Celui-ci peut être séparé en 2 parties « la division » et « la fusion » qui seront développées plus tard ci-dessous. L'algorithme prend en paramètre une image et crée à partir de celle-ci une nouvelle image dite segmentée selon un critère choisi indépendamment par l'utilisateur.

Nous verrons dans un premier temps les différentes structures de données que l'on a mise en place pour utiliser cet algorithme. Puis nous verrons le fonctionnement de l'algorithme étape par étape.

Split and merge

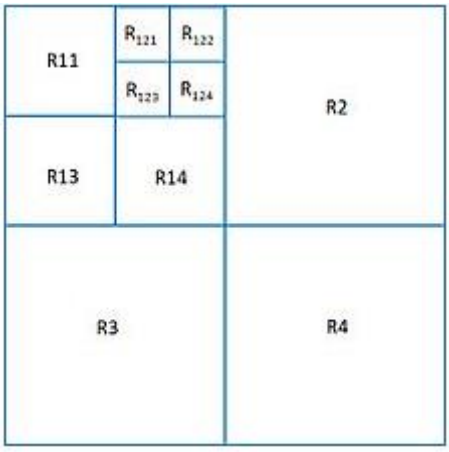
Pour réaliser cet algorithme, nous avons utilisé 2 structures de donnée. La première est une structure « Region » pour représenter les différentes régions, c'est-à-dire les parties composant une image. Une région est composée d'un point de coordonnées (X, Y), d'une largeur, d'une hauteur et d'une couleur.

La deuxième structure de donnée utilisée est un vecteur de région qui nous permet de stocker les régions lors des différentes étapes de la segmentation.

Comme dit dans l'introduction, l'algorithme « Split and merge » est composé de deux étapes.

Split

La première est la partie division (split) qui consiste à diviser les régions stockées dans le vecteur « VectorRegion » en 4 tant que le critère d'homogénéité utilisé n'est pas validé. Pour réaliser cette division, nous avons divisé la région initiale qui représente toute l'image puis nous lançons la fonction de division «splitRegion » sur les sous-régions, tant que le cas d'arrêt c'est-à-dire le critère d'homogénéité n'est pas respecté. Lorsque qu'une division est effectuée nous supprimons la région divisée du vecteur « VectorRegion » puis nous ajoutons les 4 sous-régions à la suite à la place de la région divisée. Dans notre programme, le critère d'homogénéité est la différence entre le pixel qui a la plus grande nuance de gris dans la région et le pixel qui a la plus faible. Si ce critère est inférieur à 10 alors la région est considérée comme homogène et donc n'est pas divisé.



1- Image représentant le fonctionnement de la division d'une région

Merge

La partie fusion (merge) consiste à fusionner les régions adjacentes qui ont un caractère défini par l'utilisateur en commun qui peut être le même que le critère d'homogénéité ou non. La fusion se déclenche uniquement quand il n'y a plus de division possible. Dans notre cas, les régions fusionnent lorsqu'elles sont adjacentes et que leur différence de couleur est inférieure à 10.

Concernant l'affichage, on parcourt le vecteur qui stocke les régions et l'on affiche chaque région une à une avec la couleur moyenne dans la région.

Nos résultats et implémentations

Nos implémentations

Les critères

Homogénéité : Critère que l'on va utiliser pour savoir si l'on divise la région ou pas. Si celle-ci n'est pas homogène alors on relance la fonction split sur la région.

Dans notre programme, le critère d'homogénéité est la différence entre le pixel avec l'intensité de gris maximum et le pixel avec l'intensité de gris minimum. Si ce critère est supérieur à 50 alors on divise la région.

Similarité : Critère que l'on va utiliser pour savoir si deux régions sont considérées comme similaires et si celles-ci vont fusionner entre elles.

Dans notre programme, le critère de fusion est la différence de couleur moyenne entre les 2 régions qui sont comparées. Si la différence est inférieure à 15 alors on considère que les régions sont similaires et on les fusionne sinon non.

Affichage :

Il y a 3 fenêtres qui s'ouvrent lors de l'exécution du programme.

Normal : l'affichage de l'image de base non modifié.

Split : L'affichage de l'image juste après avoir été divisé selon le critère d'homogénéité.

Split and Merge : L'affichage après les fusions sur la version divisé selon les critères de similarité.

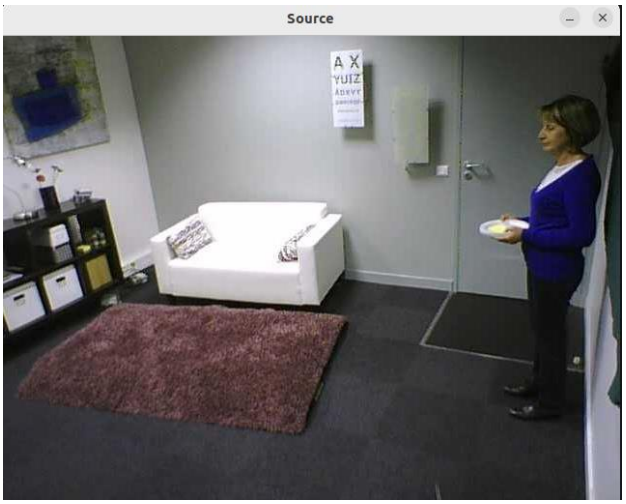
On dispose ensuite de 2 options pour l'affichage de la version Split et la version Split and Merge :

Version couleur : On attribue à chaque région une couleur aléatoire pour bien voir les différentes régions présente.

Bordure : Chaque région est délimitée par une bordure de couleur rouge pour voir la séparation clairement.

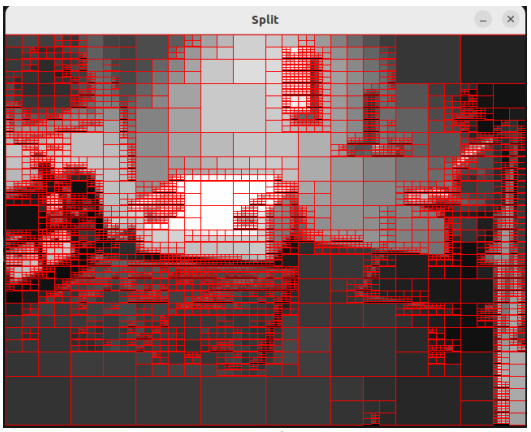
Résultats

Les mesures sont faits avec un critère d'homogénéité = 50 et un critère de similarité = 15



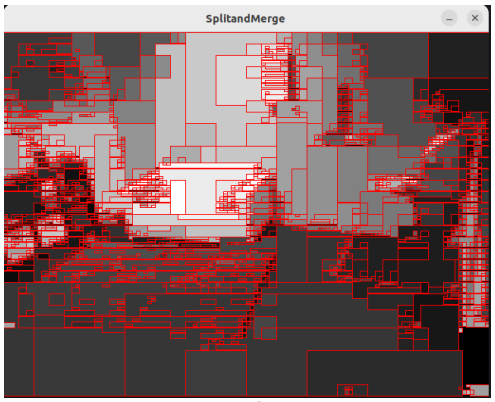
2 - Image source

3 - Split sans fusion



4 - Split sans fusion avec l'affichage des bordures

5 - SplitandMerge



6 - SplitandMerge avec l'affichage des bordures

7 - SplitandMerge avec l'affichage des bordures et les régions coloriés aléatoirement

Conclusion

Nous pensons avoir bien compris les principes de cette méthode, nous sommes maintenant capables de modifier les paramètres de notre algorithme pour avoir les résultats que nous désirons par exemple de plus grande région, etc. L'algorithme que nous avons conçu fonctionne et respecte bien les 2 étapes de split et merge. Nous avons aussi réalisé les fonctionnalités concernant les régions en couleur et les bordures de régions. Concernant nos choix nous avons décidé d'utiliser en tant que critères d'homogénéité la différence entre la couleur la plus sombre et la plus claire d'une région et pour le critère de similarité nous avons fait la différence entre les moyennes de niveau de gris. Pour stocker les différentes régions, nous avons fait le choix de les stocker dans un vecteur. Après coup nous pensons que ce n'est pas la façon la plus optimale de traiter le stockage de nos régions, mais la gestion par un vecteur nous paraissait plus simple à gérer.

Nous pensons qu'il est possible d'optimiser notre programme pour améliorer la complexité en temps et en espace. Notre façon de vérifier si nous devons fusionner n'est pas optimal car nous testons toutes les régions même les plus éloignés pour savoir si elles sont adjacentes. La gestion du stockage des régions n'est sûrement pas optimale.

Nous avons repéré un bug dans notre programme : si le critère de similarité est à 255 en toute logique toutes les régions devraient fusionner, mais ce n'est pas le cas.

Concernant les améliorations possibles, nous aurions pu afficher différents résultats en même temps avec différents critères d'homogénéité et de similarité pour les comparer efficacement.

Lien git

https://forge.univ-lyon1.fr/p1911241/informatique-graphique-et-image-tp1