M1IF02 - TP1
Rapport d'Analyse d'image

Amine ITJI 12018984
Yanisse FERHAOUI 11909519

1. Introduction

La segmentation d'image est une technique permettant de "séparer" les élements d'une image en différentes sections, c'est une étape primordiale en traitement et analyse d'images.
Ce semestre en cours d'Analyse d'images, nous avons vu les deux principales méthodes de segmentation d'images : le "Region growing" ainsi que le "Split and merge".
La première méthode, le "Region growing" (en français *croissance par région*) consiste a poser plusieurs "graines" dans une image, celles-ci se mettront ensuite à s'étendre autour de leur zone de départ par incorporation des pixels les plus similaires suivant un critère donné. Ces régions sont ensuite fusionnés suivant également des critères de fusion.
La seconde méthode, le "Split and merge" (en français *décomposition/fusion*) consiste à diviser l'image en plusieurs régions et sous-régions de plus en plus petites, ces régions seront ensuite fusionnés grâce à des critères de similarité pour ensuite former des régions plus grandes.
Lors de notre réalisation de ce TP, nous avons choisi d'utiliser le Split and merge

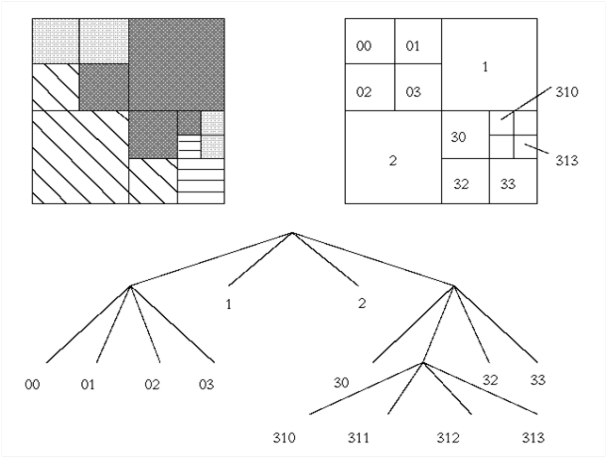

Figure 1: Image représentant un schéma d'une segmentation par décomposition/fusion

2. Split and Merge

2.1. Choix et représentation des structures des données

Dans le contexte du "Split and Merge," le choix des structures de données est crucial pour garantir une implémentation efficace de l'algorithme. Nous devons représenter les régions, les sous-régions, les pixels, et les informations associées. Une structure de données appropriée pour la gestion de l'étape de décomposition (split) pourrait être un arbre, où chaque nœud représente une région ou une sous-région. Nous avons donc opté pour une fonction récursive parcourant chaque sous région, le cas d'arrêt étant l'homogéinité de la région, car l'Arbre étant une structure de données qui fonctionne essentiellement à l'aide de programmes récursifs, nous avons donc besoin de faire un programme récursif. Nous devons bien-sûr rester attentid quand à son utilisation car les récursions infinies sont très facile à réaliser si nous avons la moindre inattention.

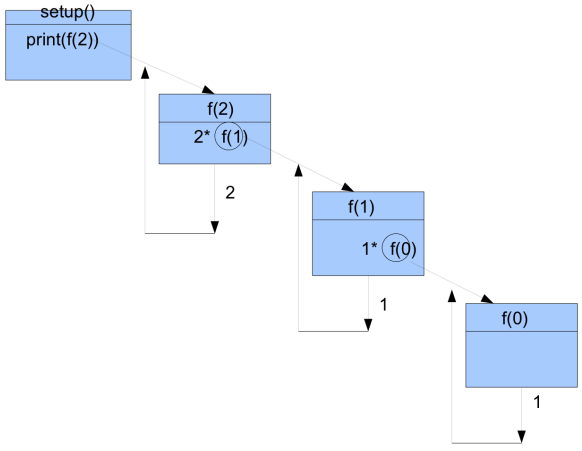

Figure 2: Image représentant un schéma de l'éxécution d'une fonction récursive

Pour l'étape d'agrégation (merge), une représentation des régions fusionnées avec des informations mises à jour est essentielle. Une approche pourrait être d'utiliser une structure de données qui stocke les relations d'adjacence entre les régions fusionnées et maintient les informations de fusion. Pour cela nous avons fait appel à la fonction **bitwise_or** qui réalisera, à chaque récursion, une opération de disjonction bit-a-bit dans la région concernée, permettant de mettre en évidence les pixels adjacents qui se ressemblent.

2.2. Choix des critères de division et de fusion

Dans le cadre du "Split and Merge," les critères de décomposition visent à subdiviser les régions trop grandes, tandis que les critères de fusion sont utilisés pour réunir des sous-régions similaires. Pour la décomposition, nous avons décidé de mettre une marge de tolérance sur la couleur, qui sera éudiée avant chaque division. Si le seuil est dépassé, la région est divisée en sous-régions. Les critères de fusion sont donc basés sur la similarité des propriétés statistiques entre deux régions adjacentes.

Dans notre code, le programme commence a diviser l'image de base en 4, et puis dans chacune des sous-images, une division en 4 autres sous régions est de nouveau faite, et ce tant que cela est nécessaire, ou bien lorsque la région sera de la taille d'un pixel.

Par conséquent, le traitement de l'image est assez long, mais nous avons réussi a optimiser correctement notre programme grâce à l'ajout de **threads**, qui permet de faire du parallélisme au sein de l'algorithme de Split and merge, et donc de traiter plusieurs régions à la fois.

Temps d'éxécution du programme avant multithreading

```
real    0m17,576s
user    0m4,450s
sys     0m14,310s
```

Temps d'éxécution du programme après multithreading

```
real    0m2,621s
user    0m4,170s
sys     0m3,474s
```

2.3. Resultats du Split and Merge

L'utilisation du Split and Merge nous a laissé de résultats assez positifs, la seule imperfection à noter étant que les rectangles des petites régions sont resté affiché sur les images segmentées.

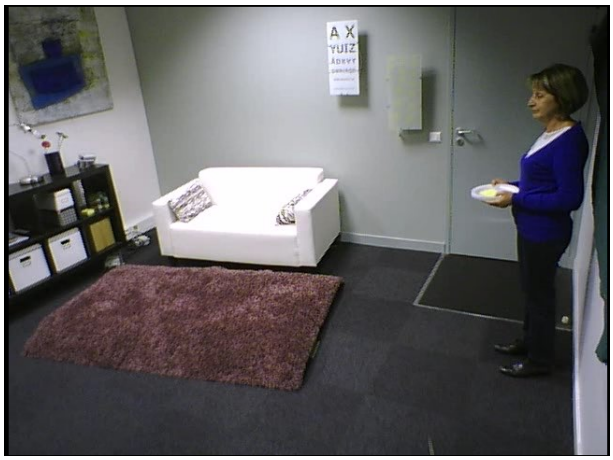

Figure 3: Image initiale

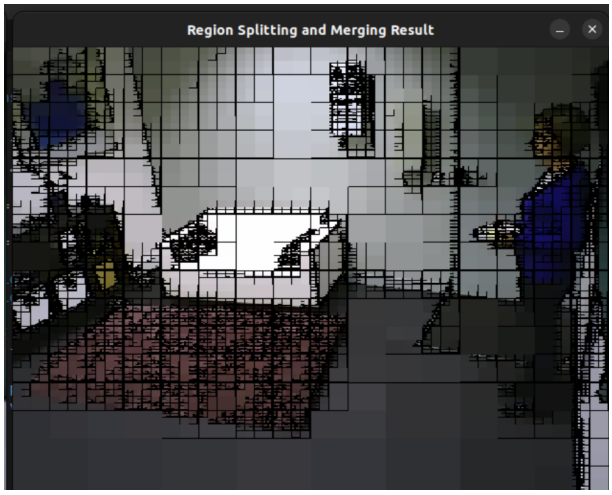

Figure 4: Image après Split and merge

2.4. La répartition des germes

Dans le "Split and Merge," la répartition des germes est liée à l'étape de décomposition. Il s'agit de déterminer où diviser l'image initiale en régions ou sous-régions. Une heuristique pourrait être basée sur la détection des zones de l'image présentant une forte variation des valeurs de pixels, indiquant potentiellement des frontières entre différentes régions. Ensuite, des germes pourraient être placés dans ces zones pour amorcer le processus de décomposition.

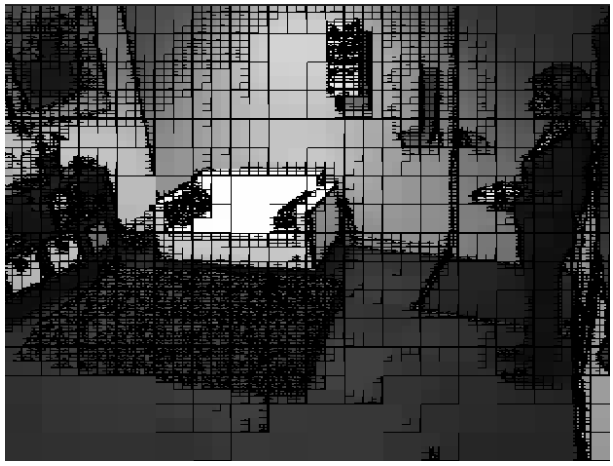

Figure 5: Image après Split and merge en nuances de gris

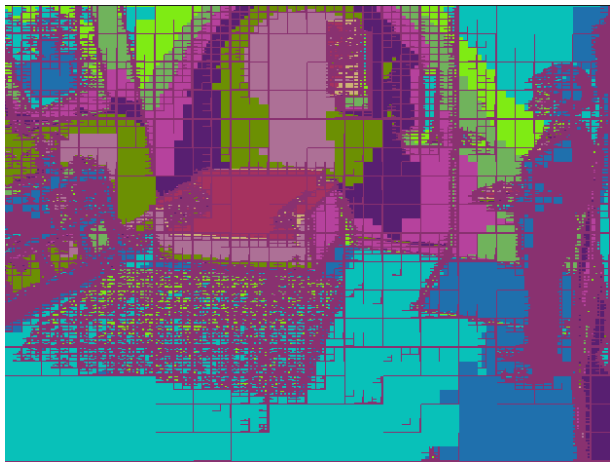

Figure 6: Image après Split and merge colorée

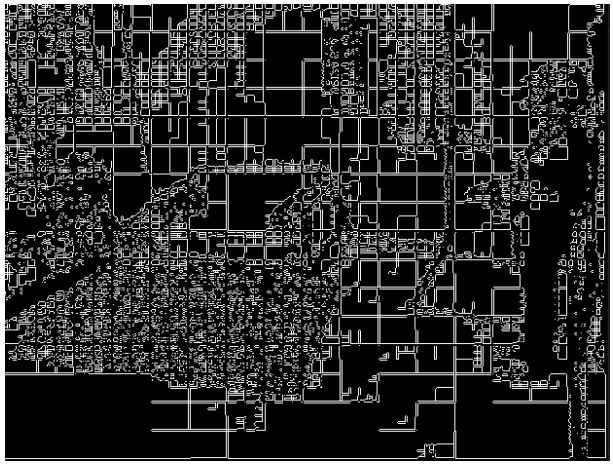

Figure 7: Image des contours de chaque région

L'objectif est d'assurer un recouvrement efficace des germes pour couvrir toute l'image tout en évitant un nombre excessif de germes, ce qui pourrait entraîner une segmentation trop fine (comme ce qui se passe dans certains endroits de nos images segmentées). Cette heuristique devrait prendre en compte la distribution des valeurs des pixels dans l'image.

En résumé, le "Split and Merge" nécessite une conception minutieuse des structures de données, des critères de décomposition et de fusion, ainsi qu'une stratégie réfléchie pour la répartition des germes afin d'obtenir une segmentation précise et efficiente.