Spécifications projet Split & Merge

tom.daguerre zoe.coudray

October 2023

Contents

1 Étapes du Programme

1. En entrée on retrouve:
   (a) Critère d'homogénéité pour un gris entre 0 et 1(à moduler en fonction du codage en niveau de gris)
   (b) Volume minimum d'un cube(en nombre de pixel)
   (c) Valeur de choix pour la méthode de construction du graphe 0 pour la méthode 1, n'importe quelle autre valeur lancera la méthode 2
   (d) Nom de fichier/chemin de l'image de départ
   (e) Nom de la nouvelle image donnée par le split and merge
2. On tente de récupérer l'image grâce au fichier
3. On crée un tableau 3D contenant la couleur des pixels de l'image
4. Opération de Split
   (a) Recherche de min max dans un cube
   (b) Si critère d'homogénéité respecté : appliquer un numéro de groupe(créer un groupe)
   (c) Sinon réitérer sur 8 sous-groupes(sous critère de volume minimum)
5. Création d'un graphe de voisinage(un tableau d'arcs)
6. Opération Merge
7. Fin: réécriture de la nouvelle image et affichage du nombre de groupe

2 Spécifs des classes

2.1 Classes objets

2.1.1 Classe annexe - Classe Coordinates

1. Attributs:
   - private int x : coordonnée du point selon l'axe x
   - private int y : coordonnée du point selon l'axe y
   - private int z : coordonnée du point selon l'axe z
2. Méthodes:
   - public Coordinates(int x, int y, int z) : Constructeur avec les coordonnées
   - public void setX(int x) : setter sur x
   - public void setY(int y) : setter sur y
   - public void setZ(int z) : setter sur z
   - public int getX() : getter sur x
   - public int getY() : getter sur y
   - public int getZ() : getter sur z

2.1.2 Classe Cube

1. Attributs:
   - private Coordinates start : coordonnées de départ du cube
   - private Coordinates end : coordonnées de fin du cube
2. Méthodes:
   - public Cube(int x0, int y0, int z0, int x1, int y1, int z1) : Constructeur avec les coordonnées
   - public Coordinates getStart() : getter sur start
   - public Coordinates getEnd() : getter sur end
   - public Bool IsNeighbourWith(Cube secondcube) : regarde si les deux cubes sont voisins, les coordonnées des deux cubes doivent être identiques sur deux axes et différentes de +-1 sur le dernier axe. Deux coordonnées sont identiques sur un axe s'il existe au moins une valeur sur cet axe sur laquelle les deux cubes sont présents.
   - public int getCubeVolume() renvoie en nombre de pixel la taille du cube

2.1.3 Classe Group

- Attributs:
  - private int groupnumber : numero du groupe (devenu inutile et supprimé)
  - private int colormin : valeur minimum de couleur dans le groupe
  - private int colormax : valeur maximale de couleur dans le groupe

- private static int nbofgroup=0 : stocke le dernier numéro de groupe attribué pour permettre la future attribution des numéros de groupe. A l'étape de Merge celui-ci stockera le nombre total de groupe diminuant à chaque regroupement.

- Méthodes:
  - public Group() : constructeur de la classe par défaut attribut un numéro de groupe en fonction de nbofgroup /méthode codé mais ne donne pas de numéro de groupe il augmente seulement nbofgroup de 1 et donne des valeurs extrèmes à colormin et max
  - public Group(int colormin, int colormax) : constructeur de la classe, prenant en paramètre des valeurs pour les attributs de couleur
  - public int getGroupnumber() : getter de l'attribut groupnumber /supprimé avec l'attribut
  - public int getColormin() : getter de l'attribut colormin
  - public void setColormin(int colorMin) : setter de l'attribut colormin
  - public int getColormax() : getter de l'attribut colormax
  - public void setColormax(int colorMax) : setter de l'attribut colormax
  - public void seekingMinMax(Cube cube, int[][][] image) : méthode recherchant la couleur minimum et maximum dans un cube pour les attribuer au groupe
  - public static int getNbofGroup() : getter de l'attribut statique nbofgroup
  - public static void lessNbofgroup() : -1 sur l'attribut Nbofgroup, correspond à la suppression d'un groupe lors de l'étape de Merge

2.1.4 Classe Arc

- Attributs:
  - private int start : numéro d'un groupe, l'origine de l'arc
  - private int end : numéro d'un groupe, la destination de l'arc

- Méthodes:
  - public Arc(int start, int end) : constructeur de la classe, prenant en paramètre des valeurs pour l'origine et la destination /Modifie l'origine et la destination si la destination est plus petite que l'origine
  - public int getStart() : getter sur start
  - public int getEnd() : getter sur end

2.2 Classes fonctionnelles et méthodes diverses

Les méthodes suivantes sont regroupés dans des classes fonctionnelles qui ne contiennent pas d'attributs. Les classes n'ont pas été réfléchis lors de la première spécification tout était regroupé dans une classe unique SplitAndMerge ce qui a été modifié plus tard mais cela ne modifie pas le fond des algorithmes présentés ci-après.

2.2.1 Split

public static void Split(float homogeneityC, int volumeMin, float[][][] image, Cube cube, Arraylist<Cube> cubelist, Arraylist<Group> grouplist) : méthode prenant en entrée le critère d'homogénéité, le volume minimum d'un cube et le cube étudié.

En sortie on a une liste de Cube et une liste des groupes associées.
Le but de la fonction est de tester l'homogénéité.
Si le cube d'entrée n'est pas homogène et que son volume est supérieur au volume minimal, il sera séparé en 8 et la méthode sera appelée à nouveau sur chacun de ces 8 cubes.
Sinon, on attribuera à celui-ci un numéro de groupe, on créer un objet Group à ajouter dans la liste, et l'objet Cube étudié sera ajouté dans sa liste associée.

Algorithme:

1. Bool test = HomogeneityTest(cube,image, homogeneityC)
2. Si test==false:
3. Si getCubeVolume(cube)¿volumeMin:
4. Pour chaque sous groupe:
5. Split(homogeneityC,volumeMin,image, nouveaucube,cubelist,grouplist)
6. Sinon:
7. cubelist.add(cube)
8. Group group= new Group()
9. grouplist.add(group)
10. group.seekingMinMax(cube,image)
11. Fin Si
12. Sinon:
13. cubelist.add(cube)
14. Group group= new Group()
15. grouplist.add(group)
16. group.seekingMinMax(cube,image)
17. Fin Si

2.2.2 fonctions TestHomogénéité

public static Bool HomogeneityTest(Cube cube, float[][][] image, float homogeneityC): teste l'homogénéité dans un cube, utilisé dans le split

public static Bool HomogeneityTest(Group groupe1,Group groupe2, float homogeneityC): teste l'homogénéité entre deux groupes, utilisé dans le merge

1. int maxcolor = max(groupe1.getColormax(),groupe2.getColormax())
2. int mincolor = min(groupe1.getColormin(),groupe2.getColormin())
3. Si maxcolor-mincolor<homogeneityC
4. alors retourne vrai
5. Sinon retourne faux
6. Fin Si

2.2.3 Création de Graphe de voisinage

public static Arraylist<Arc> MakeNeighbourGraph(Arraylist<Cube> cubelist)

1. Arraylist<Arc> arcs = new Arraylist<Arc>
2. Pour i itérant sur chaque cube:
3. Pour j itérant sur chaque cube à partir de i:
4. Si cubelist.get(i).IsNeighbourWith(cubelist.get(j)):
5. Arc arc = new Arc(i,j);//le cube i est associé au groupe i
6. arcs.add(arc)
7. Fin Si
8. Fin Pour
9. Fin Pour
10. return arcs

2.2.4 Merge

public static int[] Merge(float homogeneityC, Arraylist<Arc> arcs, Arraylist<Group> groups)

1. int listSize = arcs.length()
2. int tabSize =groups.lenght
3. int[tabSize] associatedGroup
4. Pour i allant de 0 à tabSize-1
5. int associatedGroup[i] =i

6. Fin Pour
7. Pour i allant de 0 à listSize-1
8. Si les groupes associés à l'origine et à la destination de l'arc sont différents :
9. Si HomogeneityTest(groupe 1, groupe 2 , homogeneityC)
10. Alors Merge2Groups(associatedGroup, numéro groupe 1,numéro groupe 2, groups)
11. Fin si
12. Fin si

2.2.5 Merge2Groups

public static void Merge2Groups(Arraylist<Group> groups, int firstGroup, int secondGroup, int[] associatedGroup)

1. int newGroup = associatedGroup[firstGroup]
2. int modifiedGroup = associatedGroup[secondGroup]
3. Si modifiedGroup<newGroup
4. newGroup = modifiedGroup
5. modifiedGroup=associatedGroup[firstGroup]
6. Fin Si
7. Pour i allant de 0 à associatedGroup.lenght-1
8. Si associatedGroup[i]= modifiedGroup :
9. associatedGroup[i]=newGroup;
10. Fin Si
11. Fin Pour
12. ChangeMinMax(groups, newGroup, modifiedGroup)
13. Group.lessNbofgroup() //On retire 1 au nombre total de groupes comme on vient d'en fusionner deux

2.2.6 ChangeMinMax

public static void ChangeMinMax(Arraylist<Group> groups, int newGroup, int modifiedGroup) : va modifier les valeurs de colormin et colormax dans le groupe associé le plus petit afin de concerver la cohérence dans la suite du programme

3 Sorties

Nous avons défini 4 sorties possible pour notre programme:

- 0 : le programme s'est terminé sans problème évident
- 1 : Il manque un ou des arguments
- 2 : Le critère d'homogénéité n'est pas compris entre 0 et 1
- 3 : L'image d'origine n'a pas été trouvée ou n'est pas au bon format
- 4 : L'image a une dimension différente de 3

4 Modélisation UML