FUCHS Steve
HAHN Philippe
SCHEFFKNECHT François

Tuteur : Mme A. Deruyver

Projet :

Traitement d'images

Watershed et fusion de régions

Application de l'algorithme de Watershed

I] Préparer le terrain

L'image devra être, avant tout, convertie en noir & blanc et elle aura été, cela est toutefois facultatif, soumis soit à un filtre médian, soit à un filtre gradient, soit aux deux.

Dans le cadre de l'application de l'algorithme de Watershed, a chaque point de l'image, on doit associer un label. Il nous faut donc recopier l'image dans un tableau de structure définit comme suit :

- Le tableau doit avoir pour hauteur le nombre de pixels de hauteur de l'image,
- Le tableau doit avoir pour largeur le nombre de pixels de largeur de l'image,
- Le tableau doit contenir des objets définit comme suit :

```
class WPoint
{
    int x;
    int y;
    int nGris;
    long label;
}
```

Chaque pixel de l'image de coordonnées (x, y) sera converti en un objet de type WPoint qui sera placé dans le tableau à l'emplacement [x] [y].

Il faudra, pour cela :

1. Recopier l'intensité de gris (de 0 à 255) du pixel dans la variable nGris,
2. Initialiser label à NULL (non affecté).

Les variables x et y de l'objet sont facultatif, elles représenteront les coordonnées du WPoint dans le tableau et leurs présence dépendra de leurs utilité.

Ensuite, il nous faudra initialiser une liste définie comme suit :

- La liste contiendra des éléments de type File <WPoint>,
- Elle contiendra autant de files que de niveau de gris dans l'image,
- Les files seront ordonnées du plus petit niveau de gris au plus grand,
- Chaque file sera initialisée correctement (vide au départ).

Ces files contiendront en fait des WPoints référençant des WPoints du tableau.

Nous voilà donc avec tout les éléments nécessaires au bon déroulement des opérations…

II] Recherche des minima

La première étape de l'algorithme de lignes de partage des eaux est d'initialiser correctement les files de la liste.

Initialiser les files, d'accord, mais avec quoi ?

En effet, les files doivent être initialisées avec les minima de l'image, c'est à dire les points de niveau de gris les plus bas.

Comment les trouver, deux méthodes s'offrent à nous :

1) Recherche des minima sans l'aide des marqueurs

Comment retrouver les minima sans l'aide de l'utilisateur ?

Cette technique se déroule en trois temps. Dans un premier temps, elle consistera à faire un balayage horizontal, ensuite de faire un balayage vertical et finalement de corriger les résultats.

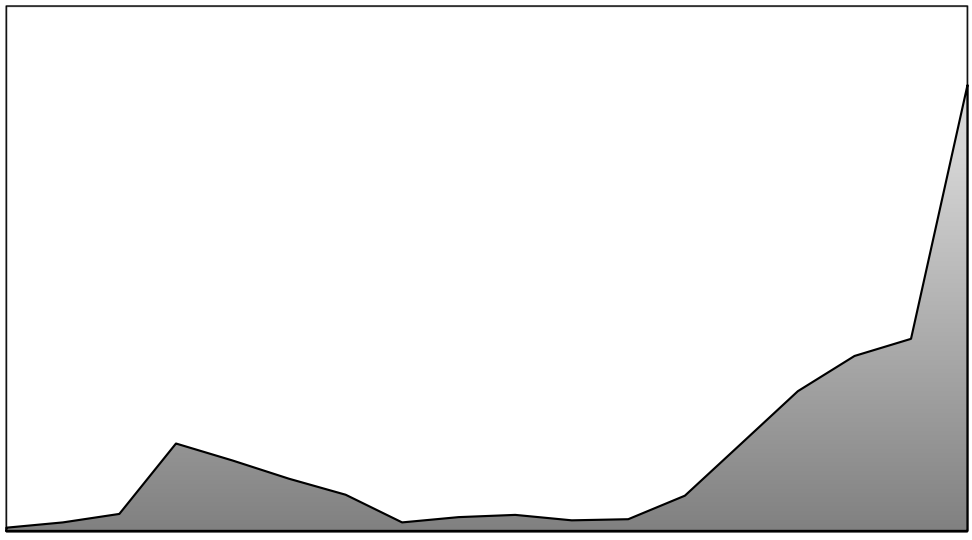
a) Etape 1 : balayage horizontal

Le tableau est parcouru ligne par ligne, de gauche à droite et de bas en haut.

Chaque ligne peut être matérialisée par une topographie ; exemple :

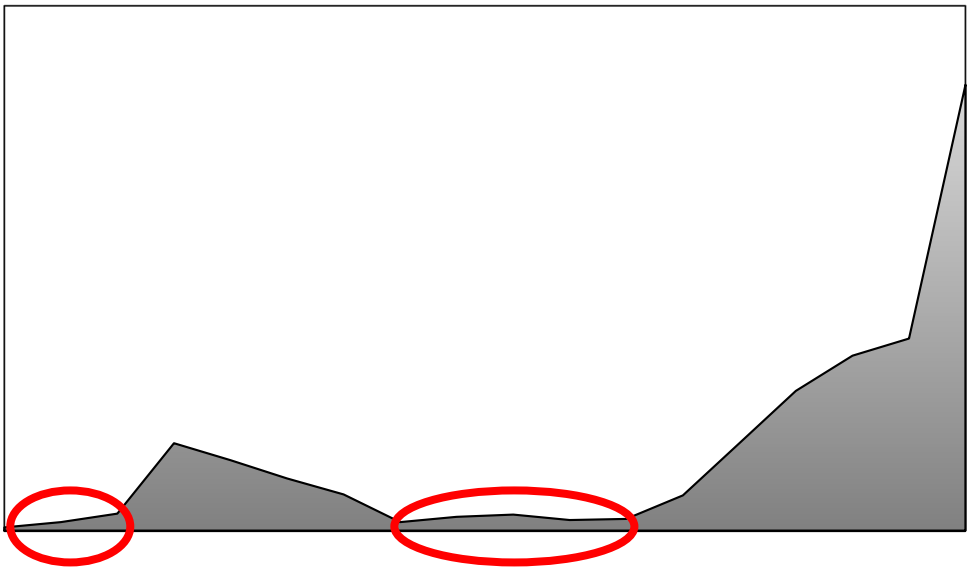
2	5	10	50	40	30	21	5	8	9	6	7	20	50	80	100	110	255

les niveaux de gris des WPoints d'une ligne du tableau



la topographie correspondante

Comme on le remarque, il est alors facile de retrouver les minima locaux sur cette topographie. Je les représente en rouge ci-dessous :



Comment retrouver alors les minima de façon plus informatique ?

C'est simple, il nous suffira d'observer l'évolution des niveaux de gris de la ligne en suivant l'algorithme suivant :

```
pour
   x un WPoint pointant sur le premier élément de la ligne
   y, z des WPoints
tant que x ≠ NULL répéter
   soit y le WPoint suivant x dans la ligne
   si y = NULL
      alors si le WPoint précédent x est de niveau de gris inférieur
             alors labelliser(x)
             sinon rien
            fsi
      sinon si le prochain WPoint après x a un niveau de gris supérieur
             alors labelliser(x)
             sinon rien
            fsi
   fsi
fpour

fonction labelliser(WPoint x)

si un des WPoint voisin y de x est déjà labellisé
   alors x.label = y.label
   sinon x.label = newLabel()
fsi

fonction newLabel()

pré : le tableau de WPoint traité
post : le plus grand numéro de label alloué + 1
```

b) Etape 2 : balayage vertical

Le balayage vertical reprend le principe du balayage horizontal sauf que ce dernier se fait en parcourant le tableau colonne par colonne, de bas en haut et de gauche à droite.

L'algorithme est sensiblement le même, sauf que l'on traite les colonnes au lieu des lignes.

c) Etape 3 : Correction

J'ai remarqué que cette méthode met à jour un dilemme.

Considérons la situation simplifiée suivante :

10	50	20
100	40	255

le tableau initial des WPoints matérialisés par leurs niveaux de gris

Quel sera le résultat de l'algorithme expliqué avant ?

En fait, le résultat est variable suivant l'ordre dans lequel seront traités les différents voisins du WPoint traité dans la fonction « labelliser » ; on peut avoir les deux cas suivants :

10	50	20
100	40	255

ou encore

10	50	20
100	40	255

Les couleurs matérialisent le label affecté au WPoint en question.

Où est le problème me direz-vous ?

Il vient du fait que le label de la valeur 40 est très aléatoire… Trois hypothèses sont alors envisageables :

- On laisse la situation tel quelle, on considère que cette méthode est bonne.
- On effectue un balayage en diagonale en « dé labellisant » les WPoints qui n'ont pas le droit de l'être…
- On effectue une procédure de correction, qui consistera à re labelliser correctement l'ensemble des WPoints déjà marqué. Dans ce cas-ci, nous

aurons alors les WPoints de niveaux de gris 10, 20 et 40 qui auront tous le même label.

Ce choix deviendra concret en fonction des résultats que nous obtiendrons dans l'un ou l'autre cas.

2) Recherche des minima avec des marqueurs

Cette solution est la plus simple d'un point de vue programmation car elle ne nécessite pas de fonction de recherche des minima, c'est l'utilisateur, grâce à ne sorte de pinceau virtuel qui colorera les zones qu'il considère comme des minima.

Cette méthode comporte toutefois un hic, il ne faut pas tomber sur un utilisateur fainéant…

III] Initialiser la liste

Les files doivent être initialisées avec les WPoints considérés comme des minima dans l'image.

Cependant, selon la méthode choisie à l'étape précédente, certains points ne sont pas à insérer dans les files :

- avec la méthode des marqueurs, seuls les points frontière des régions marquées sont mis dans les files,
- avec l'autre méthode, tout les points doivent être mis dans la liste.

Comment insérer les points dans la liste ?

Il suffit de créer pour chaque WPoint à insérer une référence et d'insérer cette dernière dans la liste correspondant au niveau de gris du WPoint.

IV] Déroulement de l'algorithme de Watershed

L'algorithme est le suivant :

```
pour
   rien
tant que chacune des files n'est pas vide répéter
   soit x un WPoint
   soit a, b, c, d les WPoints de x      // on ne considère que 4 voisins
   x <- extraire()
   si a.label = 0
      alors  a.label = x.label
             insérer(a)
      sinon  rien
   fsi
   si b.label = 0
      alors  b.label = x.label
             insérer(b)
      sinon  rien
   fsi
   si c.label = 0
      alors  c.label = x.label
             insérer(c)
      sinon  rien
   fsi
   si d.label = 0
      alors  d.label = x.label
             insérer(d)
      sinon  rien
   fsi
fpour

fonction extraire()

pour
   x <- 0
tant que liste(x).longueur() = 0 répéter
   x <- x + 1
fpour

renvoyer liste(x).prendre()

renvoyer x

fonction insérer(WPoint x)

nPriorité <- x.nGris
liste(nPriorité).rajouter(x)
```

V] Résultats obtenus

Qu'obtenons nous alors après ces 4 étapes ?

Nous obtenons un tableau, de WPoints, tous labellisées ; c'est à dire, que chaque point de l'image traitée appartient à une région caractérisée par un label.

Ce tableau pourra être ensuite aisément transcrit en une image affichable qui affichera bien chaque régions trouvées.