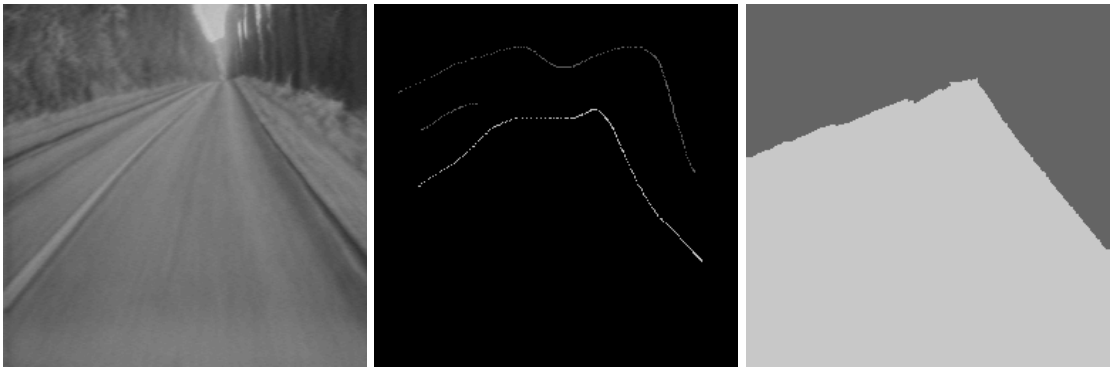
Projet : Ligne de partage des eaux

Pour ce projet, vous devrez rendre un (ou plusieurs) fichier(s) de code C dans un premier temps, ainsi qu'un rapport en PDF dans un deuxième temps. Dans le rapport, vous devrez mettre une introduction au problème, décrire les parties importantes de votre code (pas besoin de mettre tout le code...) ainsi que les éléments qui vous semblent nécessaires de mettre en valeur, répondre aux questions qui vous sont posées, et conclure sur le projet.

Le projet se fera par groupes de deux au maximum, tout plagiat entre groupes sera sanctionné par un 0 POUR LES DEUX GROUPES (les copieurs et les copiés). Votre travail est personnel, si vous le donnez à un autre groupe, c'est à vos risques et périls !

1 Introduction à la ligne de partage des eaux

La ligne de partage des eaux est un algorithme permettant de *segmenter* un objet dans une image, c'est à dire trouver avec précision les contours d'un objet précis dans une image. Pour fonctionner, il faut tou d'abord transformer l'image et en calculer le gradient : on obtient alors une image où les contours de tous les objets ressortent. Ensuite, en indiquant à l'algorithme quelques points dans l'objet d'intérêt, et quelques points hors de l'objet d'intérêt, ce dernier tentera de trouver les limites précises de l'objet souhaité.



(a) (b) (c)

FIGURE 1 – a) Une image d'une route. b) Une image de marqueurs, avec trois couleurs de pixels : les pixels à 0, ceux à 100 (hors de la route) et ceux à 200 (dans la route). c) Le résultat de la ligne de partage des eaux : tous les pixels de la route sont à la valeur 200, et ceux hors de la route sont à la valeur 100.

L'algorithme peut fonctionner en toute dimension (nous nous limiterons à des images 2d classiques dans ce travail), sur des images couleur (nous nous limiterons à des images en niveau de gris dans ce travail) et détecter plusieurs objets si nécessaires (nous nous limiterons à seulement deux objets).

2 Première étape : ouvrir une image et lire les pixels

Nous utiliserons, dans ce projet, la librairie lodepng afin de lire des images au format PNG. Cette librairie avait déjà été utilisée dans le cadre des TPs d'Informatique de base, dont voici un extrait de l'énoncé :

"Le C ne possède pas, nativement, de fonction permettant de gérer facilement les images. Il est tout à fait possible d'utiliser la fonction **fopen** puis de réaliser sa propre routine de lecture, mais une image est rarement écrite telle quelle sur le disque : elle est souvent compressée (JPEG, PNG, ...), et il faut alors implémenter une routine de décompression pour pouvoir la lire.

Pour éviter d'avoir à faire cela, nous allons utiliser une librairie extérieure de lecture d'images appelée LodePNG (`http://lodev.org/lodepng/`). Les fichiers *lodepng.c*, *lodepng.h*, *example_decode.c* et *example_encode.c* sont inclus parmi les fichiers de ce TP.

Lorsque vous utilisez LodePNG pour ouvrir une image, vous obtiendrez un tableau de **unsigned char** de type RVBa :

— La première case du tableau indique la quantité de rouge du premier pixel de l'image (0 pour signifier que le rouge est absent, 255 pour signifier qu'il est très intense),

— La seconde case du tableau indique la quantité de vert du premier pixel de l'image (0 pour signifier que le vert est absent, 255 pour signifier qu'il est très intense),

— La troisième case du tableau indique la quantité de bleu du premier pixel de l'image (0 pour signifier que le bleu est absent, 255 pour signifier qu'il est très intense),

— item La quatrième case du tableau indique la transparence du premier pixel de l'image (255 pour signifier que le pixel est complètement opaque, 0 pour signifier qu'il est complètement transparent),

— La cinquième case du tableau indique la quantité de rouge du second pixel de l'image,

— Etc..."

Questions

1. Proposez une structure de données permettant de stocker les données relatives à l'image, à savoir sa hauteur, sa largeur, et un tableaux 2d représentant le canal rouge de l'image. L'image étant en noir et blanc, la valeur de rouge de chaque pixel est égale à sa valeur de bleu ainsi qu'à sa valeur de bleu : il est donc inutile de stocker les informations des trois canaux couleur, car ils sont égaux. De plus, on considèrera que les images ne sont pas transparentes : le niveau alpha de chaque pixel est égal à 255. Nous insistons ici sur le fait que les valeurs des pixels sont des entiers entre 0 et 255.

2. En vous inpirant du corrigé du TP3 d'Informatique de base (groupe avancé) disponible sur l'ENT, proposez une fonction **LireImage(char* nom_fichier)** permettant de lire une image PNG et de la stocker dans votre structure qui sera renvoyée en sortie.

   A la compilation, vous devrez compiler votre programme en même temps que le fichier *lodepng.c* :

   ```
   gcc mon_prog.c lodepng.c -o mon_prog.exe
   ```

3. Proposez une fonction **void EcrireImage(Im, char* nom_fichier)** permettant d'écrire l'image *Im* dans le fichier *nom_fichier*.

4. Proposez une fonction **AllouerImage(hauteur, largeur)**, qui construit et renvoie une structure de données représentant une image vide de hauteur *hauteur* et largeur *largeur*.

5. Proposez une fonction **LibererImage(Im)**, qui libère de la mémoire la structure de données *Im* représentant une image.

3 Seconde étape : calculer l'image de gradient

Pour calculer l'image de gradient de l'image Im, il faut construire une nouvelle image vide nommée *Gradient*, ainsi que choisir un rayon entier r (en général, on aura $r = 1$), et appliquer l'algorithme suivant :

pour chaque *pixel* (i, j) *de l'image* Im **faire**
  Gradient(i,j) $= \max_{\substack{k\in[i-r;i+r]\\ l\in[j-r;j+r]}} Im(k,l) - \min_{\substack{k\in[i-r;i+r]\\ l\in[j-r;j+r]}} Im(k,l)$
fin

Algorithme 1 : Calcul de l'image de gradient

Questions

Proposez une fonction **CalculerGradient(Im, rayon)** qui calcule et renvoie l'image de gradient de l'image *Im* calculée à l'aide d'un rayon *rayon*. Dans l'algorithme **??**, si un pixel (k, l) "sort" de l'image Im, on l'ignorera simplement du calcul.

Sur la figure **??**, on montre le résultat du gradient obtenu sur l'image de la route de la figure **??**, avec différentes valeurs pour le rayon.

(a) (b) (c)

FIGURE 2 – a) Le gradient de l'image de route calculé avec une valeur de rayon de 1, b) avec un rayon de 5, c) avec un rayon de 10.

4 Dernière étape : calculer la ligne de partage des eaux

Pour appliquer l'algorithme de ligne de partage des eaux et extraire un objet précis de l'image, il faut ouvrir "l'image de marqueurs" associée à l'image. Dans cette image, que nous appellerons M par la suite, les pixels valant 100 représentent les pixels hors de l'objet, les pixels à 200 représentent les pixels dans l'objet, et les pixels à 0 sont ceux dont nous ne savons pas s'ils sont à l'intérieur ou à l'extérieur de l'objet. Le but de l'algorithme de ligne de partage des eaux est d'attribuer la valeur 100 ou 200 à tous les pixels de M valant 0 (selon qu'ils soient considérés comme à l'intérieur ou à l'extérieur de l'objet).

L'algorithme de la ligne de partage des eaux se déroule ainsi :

Soit L une structure de données vide au départ
pour chaque *pixel* (i, j) *de l'image* M **faire**
  si $M(i, j) \neq 0$ **alors**
    $L = L \cup \{(i, j)\}$
  fin
fin
tant que $L \neq \emptyset$ **faire**
  Soit $(i, j) \in L$ tel que, $\forall (x, y) \in L, Gradient(i, j) \leq Gradient(x, y)$
  $L = L \backslash \{(i, j)\}$
  pour chaque $(k, l) \in [i - 1; i + 1] \times [j - 1; j + 1]$ **faire**
    si $M(k, l) = 0$ **alors**
      $M(k, l) = M(i, j)$
      $L = L \cup \{(k, l)\}$
    fin
  fin
fin

Algorithme 2 : Calcul de la ligne de partage des eaux

Le résultat de l'algorithme est obtenu dans l'image M, où tous les pixels ont, à la fin, une valeur différente de 0 : soit 100 s'ils ont été identifiée en dehors de l'objet, soit 200 s'ils ont été identifés dedans.

Questions

Proposez une fonction **CalculerLPE(Gradient, M)** qui calcule et renvoie la ligne de partage des eaux calculée sur l'image *Gradient* et l'image de marqueurs *M*. Dans l'algorithme **??**, si un pixel (k, l) "sort" de l'image *Gradient*, on l'ignorera simplement du calcul.

Réfléchissez à l'implémentation de votre algorithme, et particulièrement au choix de la structure de données *L*. Une idée pourrait être d'en faire une liste triée sur la valeur de *Gradient* de chaque pixel (k, l), mais un meilleur choix qui permet d'accélérer l'algorithme existe... N'oubliez pas que les valeurs de l'image *Gradient* sont des entiers allant de 0 à 255.

La figure **??** montre un exemple de résultat de la ligne de partage des eaux obtenu avec un gradient calculé avec un rayon de 1.

5 Aller plus loin

Pour aller plus loin, vous pouvez vous interroger sur ces éléments :

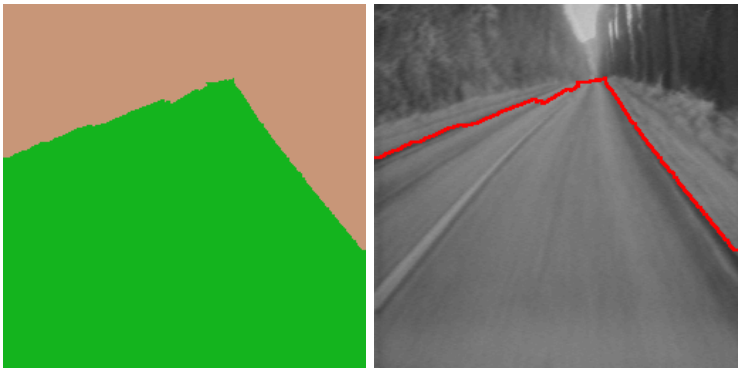
— Comment mettre en place plusieurs objets à détecter dans l'image ?

— Comment représenter les résultats de l'algorithme non pas en niveaux de gris, mais en couleur (avec des couleurs sélectionnées aléatoirement), comme montré sur la figure **??** ?

— Faites vos propres expériences avec vos propres images afin de les publier sur votre rapport.

— Pour le rapport, analysez le temps de calcul de votre algorithme en fonction de la taille de l'image.

— Comment représenter les résultats en faisant ressortir le contour des objets en rouge sur l'image résultat, comme montré sur la figure **??** ?



(a) (b)

FIGURE 3 – a) Les résultats de la ligne de partage des eaux représentés en mettant en couleur la route et le reste de l'image. b) Les résultats de la ligne de partage des eaux représentés en mettant en valeur la frontière entre la route et le reste de l'image.