TP2 - Opérations morphologiques sur des images

1 Seuillage d'une image et érosion de l'image binaire

À partir des programmes `test_grey.cpp` et `image_ppm.h` téléchargés :

a) Choisir une image (validée par l'enseignement) issue d'un appareil photographique (pas une image de synthèse) et l'enregistrer au format pgm. Réduire la taille de l'image (soit 256x256 pixels, soit 128x128 pixels).

b) Seuiller cette image en testant plusieurs valeurs de seuils. Modifier le programme `test_grey.cpp` afin que le **fond** de l'image (pixels $<$ seuil) soit en **blanc** (255) et que les pixels des **objets** soient en **noir** (0). Comparer l'image seuillée avec au moins 3 valeurs différentes et en déduire le seuil le plus pertinent.

c) À partir de l'image seuillée la plus intéressante, écrire le programme `erosion.cpp`, qui va permettre de supprimer les points objets isolés. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

a) L'image choisie pour la suite du TP est l'image de bateaux 256x256, redimensionné grâce à l'outil `mogrify` (le ! est pour forcer le redimensionnement en carré) :

```
$ mogrify -resize 256x256! boat.pgm boat.pgm
```

b) Comme les bateaux sont foncés et le fond est clair, et que nous voulons faire ressortir les objets (bateaux ici), pour cette partie, on utilise le même programme que pour seuiller au TP1. Si c'était une autre image, on aurait dû inverser le seuillage pour faire ressortir l'objet au premier plan en noir. C'est l'image seuillée à 128 qui va être utilisée pour la suite du TP :

Image de base | Image seuillée à 92 | Image seuillée à 110 | Image seuillée à 128 | Image seuillée à 150

c) Pour éroder une image, on utilise un élément structurant de k pixels. C'est à dire qu'en prenant le pixel actuel comme pixel central (on ne compte pas le pixel actuel comme faisant partie de l'élément structurant), on prend tous les pixels voisins dans un rayon de $\frac{k-1}{2}$, et le pixel sortant sera :

— blanc si on trouve un pixel blanc dans l'élément structurant.

— noir si on ne trouve pas de pixel blanc dans l'élément structurant.

Ici, on choisi un élément structurant de taille 3 (0 correspond à noir et 1 à blanc) :

0	0	0
0	0	0
0	0	0

Le pixel sortant sera noir

0	1	0
1	0	1
0	0	1

Le pixel sortant sera blanc

Image de base

Image érodée

On remarque bien que les points isolés ont disparu, enlevant toutes les vagues de l'image ainsi que les pixels noirs flottant en l'air, un peu les mats et le nom du bateau.

2 Seuillage d'une image et dilatation de l'image binaire

> À partir de l'image seuillée la plus intéressante obtenue à la question 1.b, écrire le programme `dilatation.cpp`, qui va permettre de boucher des petits trous isolés dans les objets de l'image. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

Dilater, c'est l'inverse d'éroder. Ainsi, toujours en prenant un élément structurant de k pixels, le pixel sortant sera :

— blanc is on ne trouve pas de pixel noir dans l'élément structurant.

— noir si on trouve un seul pixel noir dans l'élément structurant.

0	1	0
1	0	1
0	0	1

Le pixel sortant sera noir

1	1	1
1	1	1
1	1	1

Le pixel sortant sera blanc

Image de base

Image dilatée

3 Fermeture et ouverture d'une image de l'image binaire

La fermeture d'une image binaire consiste à enchainer une dilatation et une érosion sur l'image binaire. Cela permet de boucher des trous dans les objets contenus dans l'image binaire. L'ouverture d'une image binaire consiste à enchainer une érosion et une dilatation sur l'image binaire. Cela permet de supprimer des points parasites du fond de l'image binaire.

a) À partir de l'image seuillée la plus intéressante obtenue à la question 1.a, écrire le programme `fermeture.cpp`, qui va permettre de boucher des petits trous isolés dans les objets de l'image. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

b) À partir de l'image seuillée la plus intéressante obtenue à la question 1.b, écrire le programme `ouverture.cpp`, qui va permettre de supprimer des points parasites du fond de l'image binaire. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

c) Enchainer la fermeture et l'ouverture sur la même image, c-à-d que l'image de sortie du premier programme sera utilisée comme image d'entrée du second programme. Que constatez-vous ?

d) Enchainer l'ouverture et la fermeture sur la même image, c-à-d que l'image de sortie du premier programme sera utilisée comme image d'entrée du second programme. Que constatez-vous ?

e) Afin d'avoir plus d'impact, nous vous proposons d'amplifier les effets fermeture et ouverture sur l'image binaire. L'idée est d'appliquer séquentiellement à l'image binaire, 3 dilatations, 6 érosions et enfin 3 dilatations. Que constatez-vous ?

a) Application d'une dilatation et d'une érosion pour boucher les trous : la fermeture

Image de base

Image fermée

b) Application d'une érosion puis d'une dilatation pour supprimer les tâches : l'ouverture

Image de base

Image ouverte

c) Application de la fermeture puis de l'ouverture :

Image de base

Image fermée puis ouverte

d) Application de l'ouverture puis de la fermeture :

Image de base

Image ouverte puis fermée

e) Application de 3 dilatations, 6 érosions puis à nouveau 3 dilatations :

Image de base

Image 12 viandes

4 Segmentation d'une image

Le but de cette question est de développer une approche permettant de visualiser les contours d'une image. À partir de l'image seuillée la plus intéressante obtenue à la question 1.b et de l'image dilatée obtenue à la question 2, écrire un programme `difference.cpp` qui va nous permettre de visualiser les contours des objets contenus dans l'image :

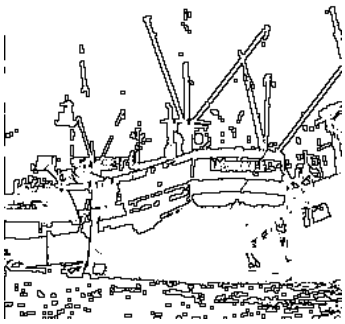
```
if (imageSeuillée[i][j] == imageDilatée[i][j]) {
    imageOut[i][j] = 255;
}
else {
    imageOut[i][j] = 0;
}
```

Écrire le programme `difference.cpp`, compiler ce programme et l'exécuter en utilisant l'image binaire et l'image dilatée obtenues précédemment avec le seuil le plus intéressant. Pour l'exécution, 4 arguments sont nécessaires, à savoir, le nom du programme, les noms des deux images d'entrée et le nom d'image de sortie.

Un `XOR` est appliqué sur l'image de base et l'image dilatée pour nous donner la différence entre les deux. Cela permet de n'avoir que les contours des objets :


Image de base


Image dilatée


Différence

5 Question bonus : extension aux images en niveau de gris, puis en couleur

Le but de cette question est d'étendre les questions précédentes (érosion, dilatation, fermeture et ouverture) dans un premier temps aux images en niveau de gris (c'est à dire sans seuiller l'image), puis aux images couleur.

Il suffit de reprendre notre élément structurant, et de prendre le pixel maximal pour l'érosion ou minimal pour la dilatation. Le seul changement à prendre en compte entre les niveaux de gris et le RVB est de faire la même chose sur les 3 octets qui composent un pixel !

Seulement l'opération de différence change : il faut prendre l'image érodée et l'image dilatée, et appliquer l'opération $\frac{|D_{[i]}-E_{[i]}|}{2}$ pour chaque pixel d'indice i de l'image. Ce n'est plus une différence, mais un gradient (le gradient positif consisterait à faire dilatation - image et le gradient négatif, érosion - image).

Les résultats sont sur la page suivante.

Image de base

Image érodée

Image dilatée

Image fermée

Image ouverte

Gradient

Image de base

Image érodée

Image dilatée

Image fermée

Image ouverte

Gradient