TP Morphologie mathématiques : les opérateurs élémentaires et leurs applications

November 21, 2014

1 introduction

Rappel Avant tout, nous vous rappelons, même si ce n'est pas nécessaire, que la documentation et des exemples consernant les fonctions que vous allez utiliser est disponnible soit dans le logiciel via la commande *help* soit en cherchant sur le web via votre moteur de recherche préférer.

1.1 Objectif

Ce premier TP du module de TSI vous permet de vous familiariser avec :

- la manipulation des données images via les fonctions de chargement comme *imread*, l'affichage *imshow* ou histogramme comme *hist* et bien d'autres.
- les opérateurs élémentaire de morphologie mathématiques
- des applications un peu plus avancées se basant sur les opérateurs dérosion, dilatation, ouverture et fermeture.

1.2 À rendre

De votre part, nous attendons deux éléments :

- Un compte rendu expliquant votre démarche, vos codes : il faut que vous montriez que vous avez compris de quoi il s'agit de faọn concise (8 pages maximum sans les figures)
- Des script : un pour chaque question avec des figures si nécessaire (toujours commentées)

1.3 Les moyens

Vous trouverez toutes les ressources (résume de cours, énoncé de TP et des morceaux de script à compléter) à l'adresse suivante : `https://github.com/matthewozon/TPmorphoMath4ETI`
Consigne à observer Le TP doit se dérouler sous OCTAVE que vous pouvez lancer soit en mode terminal via la commande *octave* soit en démarrant l'interface graphique *QtOctave*.

conseil pour les figures Utiliser la commande *print('-depsc','nom_du_fichier')* qui "imprime" ce que vous voyez dans la figure sans en diminuer la qualité (contrairement au capture décran). Pour plus d'information sur cette commande, nous vous invitons a taper dans votre prompt octave *help print*.

autre conseil Vous allez utiliser des images binaires, pensez a vous servir des opérations logique comme : *and*, *or*, *not*, *xor*, etc.

2 Prise en main

Dans cette section vous allez voir comment faire pour accéder aux données d'une image et la transformer. Commencez par créer un fichier "TP_moprho_init.m" qui commencera par les trois lignes suivantes :

clc

clear all

close all

pour éviter d'avoir des conflit de définition et des affichages multiple en navigant entre les fichiers.

2.1 Affichage d'une image

Testez le script suivant et décrivez ce qu'il fait. Apportez des modifications si nécessaire.

```
clear all;
close all;
clc;

%chargement de l'image dans un tableau
imRGB = imread('accessoire_poker_heart.jpg');

%affichage de l'image
figure(1)
imshow(imRGB)
title('mon titre')

%autre methode d'affichage
figure(2)
imagesc(imRGB)
title('mon autre titre')
%colormap jet %ligne a decommenter pour tester
%colorbar

%transformation de l'image en niveau de gris methode 1
class(imRGB(1)) %affiche le type de variable
imGray1 = imRGB(:,:,1) + imRGB(:,:,2) + imRGB(:,:,3); %somme des trois composantes... mais est-ce

%transformation de l'image en niveau de gris methode 2
imGray2 = rgb2gray(imRGB);

figure(3)
subplot(121)
imshow(imGray1)
```

```
title('methode de transformation : somme des composantes')
subplot(122)
imshow(imGray2)
title('methode de transformation : commande rgb2gray')
```

2.2 Binarisation

Maintenant que vous savez charger une image en mémoire et que vous pouvez la transformer en niveaux de gris, vous allez chercher à binariser une image et à extraire des contours. Vous utiliserez le script suivant en apportant les modifications nécessaires et vous expliquerez ce que fait le script.

```
clear all;
close all;
clc;

%chargement de l'image dans un tableau
imRGB = imread('accessoire_poker_heart.jpg'); %faut il changer de type...

%transformation de l'image en niveau de gris
imGray = ...;

%detection des contours par seuillage de la norme du gradient (estimation par filtre de sobel)
    %definition des filtre pour l'estimation du gradient
Fl = (1/4)*[-1 -2 -1;
         0   0   0;
         1   2   1];
Fc = Fl';
    %estimation des composantes du gradient
imGradl = conv2(imGray, Fl,'same');
imGradc = conv2(imGray, Fc,'same');

    %calcul de la norme du gradient
imGradNorm = ...;

    %seuillage de la norme du gradient pour extraction des contours
        %pour savoir a quelle valeur seuiller, on affiche l'histogramme de l'image a l'aide de la
imhist...
        %pour fixer le seuil, on choisit de demander a l'utilisateur le taper dans le prompt
seuil = input("nom text\n")
        %seuillage de la norme du gradient (pas de boucle autorisee)
imGradBin = ...;

%affichage des resultats
figure(1)
...
```

3 Observation des opérateurs de base

3.1 Somme et soustraction de Minkowski

En vous aidant des notes de cours (mopho_resume_cours.pdf) et des fichier minkowskiSum.m et minkowskiSub.m, vous implémenterez les opérations d'addition et de soustraction Minkowskiennes. Attention, votre code ne doit pas contenir plus de deux boucles **for**. Vous testerez vos implémentations sur des éléments simple comme un cercle avec un carre.
Attention : ces deux codes sont très importants car toutes les autres parties reposent dessus, il faut donc que vous prêtiez une attention toute particulière à la validation de cette étape.
Conseil : pour la validation, vous pouvez créer des objet simple comme le disque centre sur l'origine et de rayon 5 que réalise le script suivant.

```
[dC,dL] = meshgrid(-49:50,-49:50);
X = sqrt(dC.^2 + dL.^2)<5;
```

À rendre : explication du script et de sa validation.

3.2 Érosion, dilatation, ouverture et fermeture

Implémentation Implémentez les opérateurs d'érosion et de dilatation a l'aide des somme et soustraction de Minkowski. Implémentez ensuite l'ouverture et la fermeture en utilisant les opérateur que vous venez de coder.
Validation Dans une premier temps, tester vos algorithmes sur les images qui vous sont fournis (certaine sont binaire comme "veau2_moins_petit.bmp" et peuvent être utilisées sans prétraitement). Vous pouvez aussi comparer vos résultats à ceux des fonctions *imerode*, *imdilate*, *imopen* et *imclose*.
À rendre : Test de plusieurs éléments structurants sur une image de votre choix dans le jeu d'images fournis. Mise en évidence des propriétés de composition et de dualité pour érosion/dilatation et propriétés d'idempotence et de dualité pour ouverture/fermeture. Faire le lien entre la dilatation et le seuillage de la carte de distance. Proposez solution similaire pour l'érosion.

3.3 Hit or Miss

Implémentation En utilisant les opérateurs précédemment codés, implémentez la transformée Tout-ou-Rien.
Validation vous testerez sur l'image "bin.png" en ajoutant du bruit (des points aléatoirement distribues pour lesquels l'état du pixel est complémenté) puis en la débruitant. Indication : utilisez

```
N = randn(300,400)>2.0;
```

pour générer le bruit et la commande *xor* pour ajouter le bruit a l'image. Vous pourrez comparer votre résultat à celui de **bwhitmiss**.
À rendre : la preuve en image que votre implémentation fonctionne avec une explication de la méthode que vous utilisez pour débruiter l'image.

4 Applications

L'idée de cette partie est de mettre en oeuvre votre travail a des situations un peu plus complexe.

4.1 Détection de période

Consigne : utilisez l'image "rayure.jpg" et un élément structurant forme d'une paire de points pour déterminer la période du motif.
À rendre : estimation de la période, explication de votre méthode et la preuve en image.

4.2 Granulométrie

Consigne : utilisez l'image (tout ou partie) "NGC6960_24Juillet2006_LD.jpg" et tracez la courbe granulometrique pour deux éléments structurants de votre choix.
À rendre : explication de la méthode et illustration.

4.3 Comptage de trou

Consigne : A partir du script suivant qui génère un disque perce de plusieurs trou, compter le nombre de trou. Indication : servez vous du nombre de connexités.

```
%creation d'un grand disque centre sur le centre de l'image
[dC,dL] = meshgrid(-49:50,-49:50);
imBin = sqrt(dC.^2 + dL.^2)<43;

%creation de composantes a enlever du disque
composanteConnex = (sqrt((dC+5).^2 + dL.^2)<5) | (sqrt(dC.^2 + (dL+10).^2)<5) | (sqrt((dC-15).^2

%ajout des composantes
imBin = and(imBin,not(composanteConnex));
```

Essayer de compter le nombre de trou sur le disque de frein "disque_frein_bin.tif".
À rendre : expliquer votre méthode.

4.4 Gradient... comparaison avec Hit-or-Miss

Consigne : Implémentez le gradient morphologique et le tester sur un image de votre choix. Proposez une alternative avec la transformée Tout-ou-Rien. Comparer les deux résultats.
À rendre : expliquer votre méthode et commenter vos résultats.

4.5 Squelette

Consigne : squelettisez le petit veau! (A l'aide de votre propre méthode)
À rendre : Le squelette et son tueur!

3) Reconstruction par marqueur.

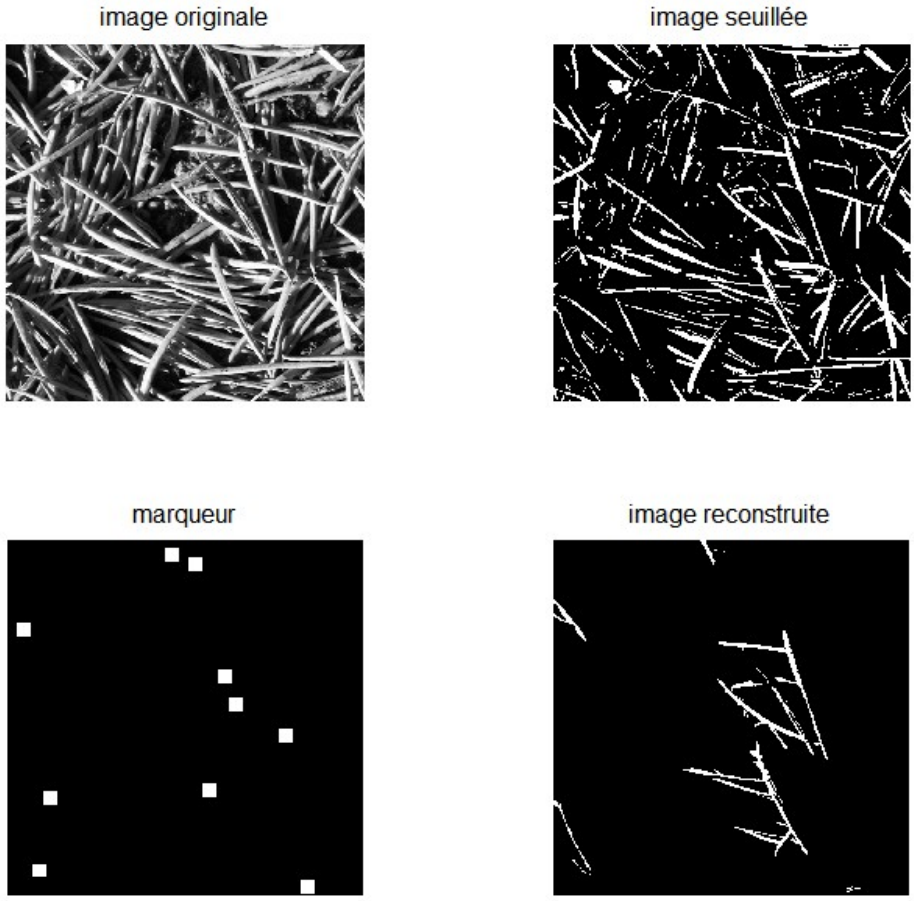

FIGURE 1 – *Image d'un tapis d'épines de pin, seuillée pour en faire une image binaire. On construit une image "marqueur" constituée de 10 "points" indiquant des zones à reconstruire. On reconstruit une nouvelle image constituée des zones blanches de l'image seuillée qui avaient au moins un pixel commun avec un pixel blanc de l'image marqueur.*

Pour comprendre la reconstruction par marqueurs, tapez le programme Matlab ci-dessous dont une trace d'exécution est donnée par la Fig. 1. Comprenez-en les différentes étapes et commentez-les.

```
clear all;close all;clc;
set(gcf,'color','w');%fond blanc
f=imread('aiguilles_pin.jpg');%à remplacer par une image de votre choix
f=f(1:512,1:512);
[a,b]=size(f);
subplot(2,2,1);imshow(f);title('image originale');
f=(f>200);
whos f
f=im2double(f);
subplot(2,2,2);imshow(f);title('image seuillée');
marq=zeros(a,b);% équivalent à : m=zeros(size(f));
for k=1:10;
    marq(ceil(a*rand),ceil(b*rand))=1;
end;
marq=imdilate(marq,ones(20),'same');
subplot(2,2,3);imshow(marq);title('marqueur');
g=imreconstruct(marq,f);
subplot(2,2,4);imshow(g);title('image reconstruite');
```

Travail personnel : Comment pourrait-on se servir de la reconstruction par marqueur pour éliminer les aiguilles qui touchent le bord de l'image (indication : prendre comme marqueur le bord de l'image)