TP Morphologie mathématiques : les opérateurs élémentaires et leurs applications

November 25, 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 faon concise (8 pages maximum sans les figures)
- Des script : un pour chaque question avec des figures si nécessaire (toujours commentées)

vous disposez d'une semaine pour rendre votre TP sous format électronique. Un depot sera ouvert sur le e-campus a partir du milieu de la semaine. Vous devrez y deposer une archive (zip ou tar) contenant :

- deux dossiers : un dossier contenant votre résumé et vos figures, un autre contenant vos script
- les script doivent être clairement identifiable. On doit pouvoir savoir de quelle question il s'agit dans le nom des fichiers ".m".
- Les script doivent montrer les éléments de réponse que ce soit en figure ou en chiffre. Chaque morceau de code doit être commenté lorsqu'il s'agit d'une opération "importante".

- Votre résumé (au format pdf), qui ne doit pas nécessairement être long (on ne note pas au kilo), doit comporter les éléments de bases pour être identifier : noms et prénoms du binôme ainsi que le nom du TP. SI vous faite référence a du cours, il faut écrire de façon complète la propriété (pas de cf...). Les figures ne sont pas des justifications par elles meêmes. Elles doivent être commentées et expliques. Nous vous conseillons de faire une petit résumé par question comportant objectif, méthode, résultat et conclusion (cela peut être aussi court que deux ou trois phrases).
- Pour tout autres questions, nous somme disposes à vous répondre soit par mail ou a nos bureau respectifs.
- Vous disposez de 10 jours après le TP pour nous rendre votre compte rendu. Nous nous attendons à ce que vous aillez le temps de faire les parties 2 et 3 au cours du TP, cependant, il faut traiter toutes les parties, en particulier la section 4.6 (reconstruction par marqueurs) que nous vous suggérons de faire après la section 3 (du point de vue temporel).

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 %efface ce qui est ecrit dans le terminal
clear all %libere les variable (elles deviennent inconues apres cette commande)
close all %ferme toutes les figures
```

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. Nous attirons votre attention sur les points suivant :

- une image est généralement en couleur, il y a donc plusieurs canaux (R: red, G: green et B: blue). Nous vous suggérons de vous en rendre compte en utilisant commande *size* sur la variable *imRGB*.

- une image est très souvent codée sur 8 bits lorsqu'il s'agit de photo "domestiques", pensez donc aux problèmes d'overflow lorsque vous manipulez les images. Le "castage" de type existe aussi sous octave, ex : *x=5.5; y = uint8(x);*.
- Essayez la commande *colormap* avec différent arguments et après différent type d'affichage.

```
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 correcte?

%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 commande imhist
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. **La transposée** d'un ensemble n'est pas sa transposée au sens matricielle (en générale).
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**.

Attention la fichier *bwhitmiss.m* à un problème. Il faut changer dans son script *erode* par *imerode*. Pour ce faire, puisque vous n'avez pas les droit administrateur, vous devez copier le fichier *bwhitmiss.m* dans votre repertoire de travail et changer cette version. Pour localiser la le fichier *bwhitmiss.m*, vous pouvez ouvrir un terminal (non octave) et taper : *locate bwhitmiss.m*, la commande *locate* localise toutes les occurence du fichier *bwhitmiss.m*, il faut donc choisir celui correspondant a la source octave.

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 + dL.^2)<5) | (sqrt((dC+25).^2 + dL.^2)<5) | (sqrt(dC.^2 + (dL-20).^2)<5) | ...
(sqrt(dC.^2 + (dL+30).^2)<5);

%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!

4.6 Reconstruction par marqueurs

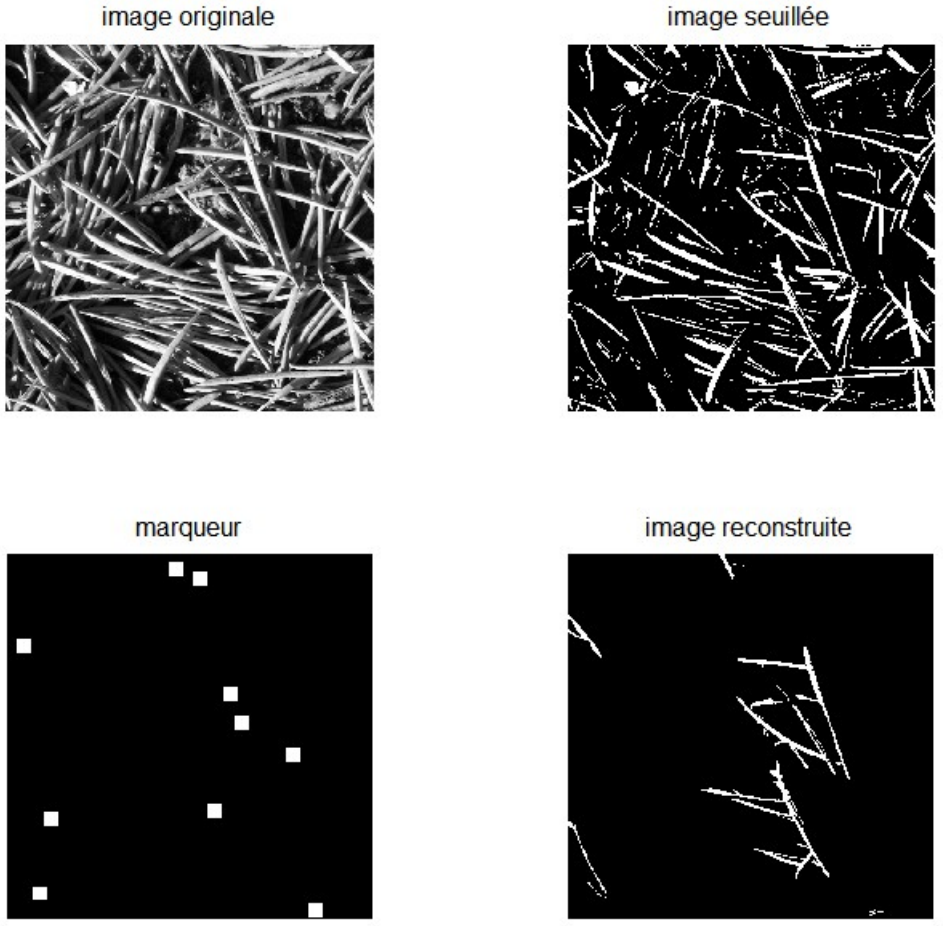

Figure 1: *Image d'un tapis d'épines de pin, seuillée pour en faire une image binaire. On construit une image "marqueur" constituée de 10 "points" indiquant des zones à reconstruire. On reconstruit une nouvelle image constituée des zones blanches de l'image seuillée qui avaient au moins un pixel commun 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');%\‘{a} 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 seuill\'{e}e');
marq=zeros(a,b);% \'{e}quivalent \‘{a} : 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)