Annexe 1

Otsu

%% Méthode d'Otsu

close all

clear

%lecture de l'image

O=imread('H01.bmp');

%conversion de l'image en niveaux de gris

Og=rgb2gray(O);

%calcul du seuil par Otsu

level=graythresh(O);

%binarization à partir du seuil

BW1=im2bw(O,level);

%affiche l'image binarisée

figure(1)

imshow(BW1)

title('Otsu seul')

Annexe 2

Méthode Canny

%%Méthode Canny

close all

clear

O=imread('H01.bmp');

Og=rgb2gray(O);

%test de fenêtres gaussiennes

%g=fspecial('gaussian',[5,5],0.83); % [7,7] , 5

%gauss=imfilter(Og,g,'same');

%figure;imshow(gauss)

%détection de contours par canny

%v: vecteur contenant les seuils bas et haut

%sigma=0.7 pour la gaussienne

v=[0.09 0.40001];

can1 = edge(Og,'canny',v,0.7);

figure;imshow(can1);

se = strel('disk', 2);

%numérotation des objets de l'image de contours

L2=bwlabel(can1);

figure;imshow(L2);

title('image des contours')

%définir les positions des objets

stat=regionprops(logical(L2),'BoundingBox');

%nombre d'objets

nL=max(max(L2));

%boucle sur tous les objets

%xy contient la position (xy(1) et xy(2))de l'objet n sur l'image et ses

%dimensions (xy(3) et xy(4))

for n=1:nL

xy=round(stat(n).BoundingBox);

%boucle sur chaque pixel de l'objet

for i=xy(1):(xy(1)+xy(3))

for k=xy(2):(xy(2)+xy(4))

%si le pixel est inférieur au seuil il fait partie de l'objet

if Og(k,i)<164

L2(k,i)=n;

end

end

end

end

%dimensions de l'image d'origine

[x y]=size(Og);

%si le pixel ne fait pas partie du fond(n=0) il est noir

L3=0\*L2+1;

L3(L2>0)=0;

figure; imshow(L3)

title('image après seuillage')

%numérotation des objets(pixels blancs dans les caractères) afin de les

%changés en pixels noirs

Lab=bwlabel(logical(L3),4);

figure;imshow(Lab)

title('labels')

%définir la position et l'air des objets

stat2=regionprops(Lab,'BoundingBox','Area');

nL2=max(max(Lab));

%boucle sur les objets hormis le fond(n2=1)

for n2=2:nL2

area=stat2(n2).Area;

%si air<15 l'objet fait partie du texte

if area<15

Lab(Lab==n2)=0;

end

%area=0;

end

%conversion en image binaire

Lab2=0\*Lab;

Lab2(Lab>0)=1;

figure;imshow(Lab2)

title('après remplissage des pixels blancs isolés')

%application d'un filtre méedian 4\*4 pour enlever le bruit

med=medfilt2(Lab2,[4,4]);

figure;imshow(med)

title('après filtrre médian')

Annexe 3

Méthode Canny et Sauvola

%%Méthode Canny + Sauvola

close all

clear

I=imread('H01.bmp');

I2=rgb2gray(I);

figure;imshow(I2)

v=[0.09 0.40001];

can1 = edge(I2,'canny',v,0.7);

figure;imshow(can1);

L2=bwlabel(can1);

figure(2);imshow(L2);

stat=regionprops(L2,'BoundingBox','Image','Area');

aL=max(max(L2));

%binarisation de l'image des contours

L3=0\*L2+1;

L3(L2>0)=0;

Im=I2;

%boucle sur chaque objet

for a=1:aL

%position de l'objet dans l'image

ss=round(stat(a).BoundingBox);

%parcourt de la l'"objet" à seuiller

%on fait glisser une fenêtre 13\*13 sur l'image de l'objet

for i=1:13:ss(4) %boucle sur les lignes

for j=1:13:ss(3) %boucle sur les colonnes

h=i;hi=i+13;

s=j;sj=j+13;

%choix de la taille de la fenêtre et création d'une matrice vide à remplir

%ensuite

%si la fenêtre 13\*13 dépace le bord bas de l'image

%sa largeur=largeur de l'image-position de la dernière fenêtre

%sa longueur=13

if ( hi>ss(4) && sj<=ss(3))

MI=zeros(ss(4)-h,sj-s);

hi=ss(4);

end

%%si la fenêtre 13\*13 dépace le bord droit de l'image

if ( hi<ss(4) && sj>ss(3))

MI=zeros(hi-h,ss(3)-s);

sj=ss(3);

end

%si la fenêtre est dans le coins en bas à droite

if ( hi>ss(4) && sj>ss(3))

MI=zeros(ss(4)-h,ss(3)-s);

hi=ss(4);

sj=ss(3);

end

%si la fenêtre est à l'intérieur de l'image

if ( hi<=ss(4) && sj<=ss(3))

MI=zeros(hi-h,sj-s);

end

%copie de la portion choisie de l'image dans cette fenêtre

o=1;

p=1;

for m=h:hi

for n=s:sj

%(-1) pour bien prendre en compte tous les pixels de l'image

%de l'objet

MI(o,p)=Im(ss(2)+m-1,ss(1)+n-1);

p=p+1;

end

p=1;

o=o+1;

end

%valeurs de MI sur une colonne

M=MI(:);

%moyenne des pixels de la fenêtre

moyen = mean(M);

%son ecart-type

deviation=std(M);

%%calcul du seuil pour la portion 15\*15

%paramètres pour le calcul du seuil

k=0.04;

R=25;

seuil= moyen\*(1 + k\*( ( deviation./R )-1));

%application du seuil sur l'image

%Im: image d'origine

%L3: image finale

u=1;

v=1;

for u=h:hi

for v=s:sj

%si le pixel est inférieur ou égale

%au seuil il fait partie du texte(pixel noir) dans l'image finale

if Im(ss(2)+u-1,ss(1)+v-1)<=seuil

L3(ss(2)+u-1,ss(1)+v-1)=0;

end

end

end

end

end

end

figure,imshow(L3);

title('image après seuillage sauvola')

mf=L3;

%numérotation des objets en vu de les changer en pixels noirs après

%détermination de leurs aires

Lab=bwlabel(logical(L3),4);

% figure;imshow(Lab)

% title('labels')

stat2=regionprops(Lab,'BoundingBox','Image','Area');

nL2=max(max(Lab));

for n2=2:nL2

area=stat2(n2).Area;

if area<7

Lab(Lab==n2)=0;

end

area=0;

end

figure;imshow(Lab)

title('après remplissage des pixels blancs isolés')

%élèment structurant

se = strel('disk', 3);

%dilatation de l'image

imd = imdilate(Lab, se);

figure, imshow(imd)

%reconstruction de l'image(mask) par sa dilatée(marker)

recons = imreconstruct(imcomplement(imd), imcomplement(Lab));

%inversion des couleurs pour avoir le texte en noir

recons = imcomplement(recons);

figure, imshow(recons), title('image après reconstruction')

%application d'un filtre médian pour enlever le bruit autour des caractères

med=medfilt2(recons,[3,3]);

figure;imshow(med)

title('reconstruction + filtre médian')

%différences entre l'image finale et l'image optimale donnée

figure;

orig=imread('H01\_GT.png');

diff=med-orig;

hold on

[x,y] = find(diff==-1);

plot(x,y,'r.')

[x,y] = find(diff==1);

plot(x,y,'b.')

hold off