Université des frères Mentouri Constantine Département d'électronique

Master: Systèmes des Télécommunications

TPs Traitement d’images

Université des frères Mentouri Constantine Département d'électronique

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP1 : Toolbox de Matlab pour le traitement d'images

Image Processing Toolbox est une collection de fonctions qui étendent la capacité de MATLAB. Cette Toolbox prend fourni une large gamme d'opérations de traitement d'images (transformation d'images spatiales, opérations de blocs, filtrage linéaire ...) Bon nombre de ces fonctions sont (des fichiers .m de MATLAB) d'instructions MATLAB qui implémentent des algorithmes de traitement d'image spécialisés. Nous pouvons afficher le code MATLAB pour ces fonctions à l'aide de l'instruction: *type nom_fonction*. Nous pouvons étendre les capacités de cette Toolbox de traitement d'images en écrivant nos propres fichiers M ou en utilisant la Toolbox en combinaison avec d'autres Toolbox de Matlab (Signal Processing, Neural Networks …)
Matlab utilise plusieurs types d'images :
1- Image en niveaux de gris: c'est une matrice d'entiers dont les valeurs représentent des nuances de gris. Les pixels d'une image en niveaux de gris sont generalement de type uint8 (ou uint16), ils ont des valeurs entières dans la plage [0, 255] ([0, 65535]).
2- Image binaire est un tableau logique de 0s et 1s.
3- Image couleur RVB est un vecteur MxNx3 de pixels, chaque pixel est un triplet correspondant aux composantes rouge, vert et bleu d'une image RGB.
4- Image indexée a deux composantes: une matrice de données contenant des nombres entiers, X, et une matrice de carte de couleur (color map). Color map est un tableau mx3 de doubles contenant des dans la plage (0, l). La longueur du map est égale au nombre de couleurs qu'il définit. Chaque rangée spécifie le rouge, le vert et le bleu d'une seule couleur. La couleur de chaque pixel est déterminée en utilisant la valeur correspondante de la matrice entière X comme un index dans le color map.

1 Rappel Matlab

Parce que les images sont stockées sous formes de matrices, on va rappeler ici quelques opérations de manipulation des matrices sous Matlab. Tester le code suivant:

```
A=[1 2 3;4 5 6;7 8 9]      B=[10 20 30; 40 50 60;70 80 90]

A(:,1)   A(:,2)    A(:,3)    A(1,:)   A(2,:)   A(3,:)   A(:)

C=A+B    C=A*B    C=A.*B    A&B     A|B

max(A)    min(A)     sum(A)

min(min(A))   max(max(A))  sum(sum(A))

max(A(:))  min(A(:))  sum(A(:))   [C,I]=max(A)

isequal(A,B)  isprime(A)  numel(A)

AV= sum(A(:))/numel(A)
```

On va synthétiser plusieurs types d'images et on va les afficher.

Convertir une matrice simple en une image en niveaux de gris:

```
image(A)
```

Créer une image couleur aléatoire:

```
rgb=rand(10,10,3);
imagesc(rgb)
```

Créer une autre image couleur:

```
plane = zeros(11, 11, 3);
line = 0:0.1:1;
plane(:, :, 1) = ones(11, 1)*line;
plane(:, :, 2) = (ones(11, 1)*line)';
imagesc(plane)
```

On va utiliser la fonction `meshgrid()` pour créer des images:

```
x = linspace(-pi, pi, 201);
[xx,yy] = meshgrid(x);
A = 10;
I = sin(A*(xx.^2 + yy.^2));
imshow(I, [-1 1])
```

Tester aussi le code suivant:

```
[xx,yy] = meshgrid(-125:125);
[theta,R] = cart2pol(xx,yy);
I = sin(50*theta);
imshow(I, [-1 1])
```

Tester les maps

```
load(earth.mat);
colormap(map);
imagesc(X);
```

Tester le code suivant:

```
colormap(gray)
colorbar
```

Tester les color maps: hot, cool, bone, gray, spring, summer, autumn, winter, hsv, copper,… etc.

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP1.2 : Toolbox de Matlab de traitement d'images et de la vidéo

Une opération typique de traitement d'images avec Matlab Image processing Toolbox se compose de plusieurs étapes. Dans ce TP, on va voire les opérations de lecture, affichage et stockage des images. Maitriser ces opérations est important parce qu'on les utilise avec toutes les opérations de traitement d'images qu'on va étudier dans les autres TPs de ce module.

Etape 1: Lire et afficher une image:
Pour lire une image, nous utilisons la commande imread( ). Cette fonction lit une image à partir d'un fichier image et la stocke dans un tableau (matrice)
Example :

```
I = imread('pout.tif');
```

Imread va deviner le format du à partir de son nom (extension).cette fonction reconnais plusieurs formats. Voir la documentation de imread.
La plupart des formats de fichier d'image utilisent 8 bits pour stocker les valeurs des pixels. Quand ils sont lus dans la mémoire, Matlab les stocke comme des valeurs de type uint8. Pour les formats de fichiers prenant en charge les données 16 bits, Matlab les stocke avec le type uint16.

On peut aussi lire des images couleurs RGB. Exemple:

```
RGB = imread('football.jpg');
```

On peut aussi lire une image indexée avec sa colormap associe dans deux variables séparées:

```
[X, map] = imread('trees.tif');
```

Matlab prend en charge plusieurs formats de fichiers graphiques, tels que HDF et TIFF pouvant contenir plusieurs images. Lorsqu'un fichier contient plusieurs images qui sont liées d'une certaine manière, comme une séquence temporelle, nous pouvons stocker les images sous forme de tableau 4-D. Toutes les images doivent être de la même taille. Par exemple, lorsqu'on utilise des fichiers TIFF, nous pouvons utiliser une valeur d'un index avec imread pour identifier l'image dans le fichier. Nous pouvons également utiliser imfinfo pour déterminer combien d'images sont stockées dans le fichier.
Exemple:
Dans cet exemple, on va lire une série de 27 images à partir d'un fichier TIFF et les stocker dans un tableau 4D.

```
mri = uint8(zeros(128,128,1,27)); % réserver un tableau 4D
for frame=1:27
      [mri(:,:,:,frame),map] = imread('mri.tif',frame);
End
```

Etape 2: vérifier comme l'image apparait dans le Workspace
Matlab peut stocker des images sous diverse formes (uint8, uint16, doubles). Pour voir comment la fonction imread stocke les données de l'image dans le workspece, on peut le vérifier le avec le navigateur du workspace.
Vérifier la représentation dans le workspace de Matlab des images des exemples précédents.

Etape 3: Affichage des images
Pour afficher une image, on utilise la fonction : imshow
(On peut utiliser l'outil imtool qui un environnement graphique intégré qui permet l'affichage des images et d'effectuer certaines tâches courantes de traitement d'image)
exemple:

```
imshow(I)
```

On peut utiliser directement : `imshow ('image.bmp')`, mais dans ce cas l'image n'est pas enregistrée dans le workspace.
La fonction `image( )` affiche un objet image. Cette fonction crée un objet graphique de type image en interprétant chaque élément d'une matrice comme un index dans la colormap de la figure ou directement en tant que valeurs RGB, selon les données spécifiées.

La fonction `impixel()` renvoie les valeurs de couleur rouge, verte et bleue des pixels d'image spécifiés.

Etape 4: faire le traitement (ce que on va faire dans les prochains TPs).

Etape 5: écrire (sauvegarder) l'image dans un fichier (sur disque)
Pour écrire (sauvegarder) les images traitées ou les résultats du traitement dans un fichier, nous utilisons la fonction imwrite.

Exemple : `imwrite (I2, 'pout2.png');`
Dans ce cas, vous pouvez vérifier le résultat sur le disque. On peut aussi utiliser:

```
imfinfo('pout2.png')
```

La fonction imfinfo renvoie des informations sur l'image dans le fichier, telles que son format, sa taille, sa largeur et sa hauteur, etc.

`Size(I2)` par contre affiche la taille de l'image.
Tester aussi `Whos I2`.

Pour changer le format d'une image seulement, utilisez imread( ) puis utilisez imwrite( ):

```
bitmap = imread('mybitmap.bmp','bmp');
imwrite(bitmap,'mybitmap.png','png');
```

Université des frères Mentouri Constantine Département d'électronique

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP2.1 : Traitement numérique des images par MATLAB

Transformations couleur

Lire une image couleur (peppers.png)

Convertir cette image en espace HSV. Utiliser pour cela la fonction rgb2hsv( ).

```
HSV = rgb2hsv(RGB);
```

Séparer l'image résultante en plan (images) H, S et V.
Afficher les 4 images ensemble et voir le résultat. Utiliser pour cela le code suivant:

```
subplot(2,2,1), imshow(H)
subplot(2,2,2), imshow(S)
subplot(2,2,3), imshow(V)
subplot(2,2,4), imshow(RGB)
```

Convertir cette image en espace HSV. Utiliser pour cela la fonction rgb2cbcr( ).

```
YUV = rgb2ycbcr(RGB);
```

Séparer l'image résultante en plan (images) Y, U et V.
Afficher les 4 images ensemble et voir le résultat. Utiliser pour cela le code suivant:

```
subplot(2,2,1), imshow(Y)
subplot(2,2,2), imshow(U)
subplot(2,2,3), imshow(V)
subplot(2,2,4), imshow(RGB)
```

Réaliser la transformation inverse pour transformer l'image YUV vers une image RGB2. Afficher à côté de RGB et comparer.

Convertir l'image RGB en une image en niveaux de gris (L) et afficher le résultat. Utiliser la fonction rgb2gray( ). Afficher à côté de RGB et comparer.

Université des frères Mentouri Constantine Département d'électronique

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP2.2 : Traitement numérique des images par MATLAB

Lire deux images A et B (cameraman et rice.png)

Additionner les deux images pour former une image C
Afficher cette image C ?
Utiliser la fonction imadd( ) pour former une image D, afficher le résultat, quelle est la différence?

Former une image E en ajoutant une constante à l'image A, afficher le résultat

Tester le code suivant :

```
K = imadd(A,B,'uint16');
imshow(K,[])
```

Comparer!
Tester immultiply( ) par le code suivant:

```
I = imread('moon.tif');
J = immultiply(I,0.5);
subplot(1,2,1), imshow(I)
subplot(1,2,2), imshow(J)
```

Resize:

```
I = imread('rice.png');
J1 = imresize(I, 0.5);
figure, imshow(I), figure, imshow(J1)
J2 = imresize(I, 2);
figure, imshow(I), figure, imshow(J2)
J4 = imresize(I, 4);
figure, imshow(I), figure, imshow(J4)
```

Quantification

```
I1 = imread('cameraman.tif');
I2 = grayslice(I1,128); figure, imshow(I2,gray(128));
I3 = grayslice(I1,64); figure, imshow(I3,gray(64));
I4 = grayslice(I1,32); figure, imshow(I4,gray(32));
I5 = grayslice(I1,16); figure, imshow(I5,gray(16));
I6 = grayslice(I1,8); figure, imshow(I6,gray(8));
I7 = grayslice(I1,4); figure, imshow(I7,gray(4));
I8 = grayslice(I1,2); figure, imshow(I8,gray(2));
```

Transformations Géométriques sur l'image

Lire une image A.

Appliquer des opérations de rotation déplacement, affine

```
B = imrotate(A,angle)
```

Tester le code suivant:

```
cb = checkerboard;

imshow(cb)

xform = [ 1   0   0
          0   1   0
         40  40   1 ]

tform_translate = maketform('affine',xform);

[cb_trans xdata ydata]= imtransform(cb, tform_translate);

figure, imshow(cb_trans)
```

Tester d'autres transformations.

Dessiner une image (dans Paint) d'un cercle blanc (au centre) sur un fond noire; enregistrer la dans un fichier (png) comme une image couleur.

Lire l'image dans Matlab
Séparer les images R, G et B, déplacer les cercles dans des directions différentes avec un déplacement inferieur au diamètre du cercle.

Regrouper les images RGB et afficher l'image en couleur résultante.

TP de Traitement d'images

TP2.3 : Traitement numérique des images par MATLAB

Traitement sur l'histogramme

Lire l'image "pout.tif" et/ou "coins.png". afficher l'histogramme de cette image avec la fonction Matlab imhist( ) (Ex: `figure, imhist(I)`) tester aussi cumsum( ).

Quelle est la valeur moyenne et l'écart type de cette image?
Quels sont les valeurs min et max de cette image?
Es ce qu'elle a un bon contraste (avec calcul)?

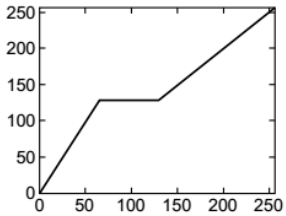
Etaler l'histogramme linéairement entre 0 et 255. Utiliser pour cela votre propre code.

Utiliser la fonction histeq( ) pour faire une égalisation de l'histogramme.
Exemple: (`I2 = histeq(I); figure, imshow(I2)`)
Afficher l'histogramme de cette nouvelle image.

Tester d'autres fonctions qui ajustent l'histogramme : imadjust( ), adapthisteq( ).
Tester aussi l'utilisation de "Image Tool".

Tester la fonction imadjust ( ) pour tester la transformation Gamma (=1, >1, <1).
Soit la transformation suivante:

$$s = \begin{cases} 2 \cdot f & \text{for } 0 < r \leq 64 \\ 128 & \text{for } 64 < r \leq 128 \\ f & \text{for } r > 128 \end{cases}$$



On va appliquer cette loi sur une image en utilisant un LUT (look up table)

```
LUT = uint8(zeros([1 256]));
LUT(1:65) = 2*(0:64);
LUT(66:129) = 128;
LUT(130:256) = (130:256)-1;
```

Utiliser ensuite la fonction intlut( ) (exemple: `intlut(I,LUT);`)
Afficher le résultat.

Filtrage

Les filtres linéaires sont implémentés dans Matlab par la fonction imfilter( ). Voir la documentation pour comprendre les différents paramètres de cette fonction.
La fonction fspecial( ) (ex: `h = fspecial(type, parameters)`) permet de créer les filtres connus: prewitt, gaussian, average…

Charger une image de teste I (exemple: cameraman.tif)
Créer un filtre avec `fltr = fspecial('average')`
Appliquer ce filtre à l'image par `Iav = imfilter(I,fltr);`
Afficher et comparer les images I et Iav. Quel est l'effet de ce filtre.

Créer un filtre moyenneur non uniforme par:

```
fltrm = [1 2 1; 2 4 2; 1 2 1]
fltrm = fltrm  * (1/16)
```

Appliquer ce filtre et comparer son résultat avec le filtre précédent.
Créer votre propre filtre (proposer des coefficients) et vérifier son résultat.

Créer un filtre gaussien avec:

```
fgaus = fspecial('gaussian',9,1.5);
figure, bar3(fgaus,'b');
```

Appliquer ce filtre à l'image I et afficher son résultat. Changer la variance de ce filtre et voir le résultat.

Créer un filtre Laplacien avec:

```
f = fspecial('laplacian',0);
```

Appliquer ce filtre à notre image de teste. Avant d'appliquer le filtre, il faut convertir l'image au format double.
Afficher le résultat de filtrage.

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP3 : Traitement fréquentiel des images sous Matlab

Lire l'image `Img` appelée 'testpat1.png'.
Effectuer sa transformation de Fourier avec:

```
FImg=fft2(single(Img)); FImg=fftshift(FImg);
```

Afficher l'image originale (avec: `subplot(2,2,1);`) et le résultat de TF avec:

```
subplot(2,2,2); imagesc(log(1+(PSF)));
axis image; axis off;
```

Utiliser (`imagesc(angle(FA));`) pour afficher la phase.
Créer un filtre passe bas qui a la même taille que l'image dont la TF est calculée avec:

```
LTF=zeros(size(A))
S=1;
LTF(128-S:128+S,128-S:128+S)=1;
```

Appliquer ce filtre à l'image dans le domaine fréquentiel (pour obtenir fImgTF) et calculer la TF inverse avec :

```
Imgfilt=ifft2(fImgTF);
Imgfilt=fftshift(Imgfilt);
```

Afficher les résultats avec :

```
subplot(2,2,3); imagesc(log(1+abs(LTF)));
axis image; axis off;
subplot(2,2,4); imagesc(abs(Imgfilt));
axis image; axis off;
```

Changer la valeur de S et voir ce qui se passe.

Créer un filtre Gaussien qui a la même taille que l'image et calculer sa TF avec:

```
PSF=fspecial('gaussian',size(Img),6);
OTF=fft2(PSF); OTF=fftshift(OTF);
```

Appliquer ce filtre à l'image dans le domaine fréquentiel et afficher le résultat.

Utiliser le même code du filtre passe-bas idéal, pour réaliser un filtrage passe-haut idéal.

Tester de Reconstruire une image avec la TF inverse en utilisant l'amplitude seulement ou la phase seulement. Tester le code suivant:

```
Img=imread('cameraman.tif');
FImg=fft2(single(Img)); Fimg=fftshift(FImg);

mag1=abs(FImg);
s=log(1+fftshift(mag1));
phase1=angle(FImg);

r1=ifftshift(ifftn(mag1));
r2=abs(ifftn(exp(1i*phase1)));

figure,imshow(FA);
figure,imshow(s,[]);
figure,imshow(uint8(r1));
figure,imshow(r2,[]);
```

Quells sont vos remarques?
Tester d'autres images

Calculer la DCT et éliminer quelques composants. Tester le code suivant:

```
S=10
RGB = imread('autumn.tif');
I = rgb2gray(RGB);
J = dct2(I);
imshow(log(abs(J)),[]), colormap(jet(64)), colorbar

J(abs(J) < S) = 0;
K = idct2(J);
imshow(I)
figure, imshow(K,[0 255])
```

Changer la valeur se S et voir le résultat, Remarques?

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP4.1 : Détection de contours

Dans l'exemple suivant, observerons le changement des valeurs des pixels voisins d'une seule ligne et le passage par un contour.

```
Img = imread('cameraman.tif');
figure, imshow(I), title('Input image')

Row = 164;    % ligne numéro 164 de l'image
x = double(Img(Row,:));
Col = size(Img,2);

figure,subplot(2,1,1),plot(1:Col,x,'k','LineWidth',2)
xlabel('Pixel number'), ylabel('Amplitude')
```

Calculer la dérivée de x et la dérivée seconde de x et afficher les.

Calculer les dérivée de l'image et afficher le résultat.
Calculer l'amplitude et la direction du gradient et afficher les.
3X3 et 5X5 et 7x7
[Fx,Fy]=gradient(I);
Fx_sq=Fx.^2; Fy_sq=Fy.^2; Fx_Fy=Fx.*Fy;

Lire une image (circuit.tif, cameraman.tif, peppers.png) et appliquer la fonction Matalb edge( ).
Pour les filtres : roberts, prewitt et sobel.

Afficher l'image originale et les 3 résultats de détection de contours.

Maintenant, on va tester LoG et Canny. Pour cela, lire une image et appliquer la fonction Matalb edge( ) avec les valeurs : log et canny.

On va utiliser 2 filtres gaussiens pour filtrer l'image avec sigma=6 et 12 respectivement.

```
h1=fspecial('gaussian',[15 15],6);
h2=fspecial('gaussian',[30 30],12);
```

Afficher le résultat du filtrage de l'image par ces deux filtres.

Appliquer la fonction Matalb edge( ), pour log et canny, sur le resultat.

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP4.2 : Segmentation

Seuillage

Le code suivant permet de faire un seuillage de l'image avec deux techniques. Pour commencer le test, lire l'image (coins.png) et afficher l'image et son histogramme.

Pour faire le seuillage, on va tester plusieurs seuils à la main. Trouver le meilleur seuil. Utiliser pour cela la fonction Matlab im2bw( ) (exemple: `im2bw(I,0.35);`)

Maintenant, on va calculer le seuil automatiquement (méthode de Otsu). Pour cela, on utilise la fonction Matlab graythresh( ).

```
level = graythresh(I);
BW= im2bw(I, level);
```

Le code suivant utilise la fonction `bwtraceboundary( )` pour tracer la bordure d'un objet.

```
dim = size(BW);
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)));
boundary = bwtraceboundary(BW,[row, col],'N');
imshow(I); hold on;
plot(boundary(:,2),boundary(:,1),'g','LineWidth',3);
```

Split-and-merge

Lire une image I (cameraman.tif) et appliquer le code suivant pour une décomposition quadtree. La fonction Matlab qui réalise cette opération est qtdecomp( ).

```
S = qtdecomp(I,.17);
blocks = repmat(uint8(0),size(S)); %Create empty blocks
for dim = [256 128 64 32 16 8 4 2 1];
    numblocks= length(find(S==dim));
    if (numblocks > 0)
        values = repmat(uint8(1),[dim dim numblocks]);
        values(2:dim,2:dim,:)= 0;
        blocks = qtsetblk(blocks,S,dim,values);
    end
end
blocks(end,1:end) =1;
blocks(1:end,end) = 1;
subplot(1,2,1), imshow(I);
k=find(blocks==1); %Find border pixels of regions
A=I; A(k)=255; %Superimpose on original image
subplot(1,2,2), imshow(A);
```

Tester aussi les images : coins.png et liftingbody.png

Université des frères Mentouri Constantine Département d'électronique

Master: Systèmes des Télécommunications

TP de Traitement d'images

TP5 : Binarisation d’images et opérations morphologiques