Traitement d'images, TP1
Manipulations d'images en Matlab

ENSEEIHT 2EN

http://oberlin.perso.enseeiht.fr/teaching.html

1 Fonctionnement des TPs

1.1 Contenu, plan des séances

8 séances sont prévues, de 1h30 chacune :

◇ 6 TPs pour pratiquer le TI en matlab, et implémenter certaines des techniques vues en cours :

· TP1 : le TI en Matlab
· TP2 : transformée de Fourier et applications
· TP3 : restauration
· TP4 : approximation
· TP5 : segmentation
· TP6 : contours

◇ Les TPs ne sont pas notés, mais une partie de l'examen porte sur des commandes Matlab ou des algorithmes vus en TP.

1.2 Organisation

Les énoncés de TP décrivent le travail à effectuer. Pour gagner du temps il est conseillé de se mettre en binôme, mais attention à bien travailler ensemble. Des exercices sont proposés tout au long du TP, ils consistent le plus souvent à implémenter un algorithme et le tester sur une ou plusieurs images. Je vous conseille de conserver soigneusement le code développé pour chaque exercice (dans un fichier séparé), et de le commenter proprement, et de prendre quelques notes : ça vous sera utile pour l'examen, et aussi pour la suite.

2 Matlab pour le traitement d'images

2.1 Lecture et écriture

Commençons par les commandes de base qui permettent la manipulation d'images en Matlab. Cette partie présente les principales, mais n'hésitez pas à compléter en consultant l'aide ! Téléchargez un dossier d'images test disponible à `http://oberlin.perso.enseeiht.fr/cours/images_TP1.zip`, puis lire une image couleur et l'afficher en exécutant :

```
f = imread('fleur.png'); n = size(f,1);
figure;
f1 = cat(3, f(:,:,1), zeros(n), zeros(n));
f2 = cat(3, zeros(n), f(:,:,2), zeros(n));
f3 = cat(3, zeros(n), zeros(n), f(:,:,3));
subplot(2,2,1); imshow(f); title('image fleur');
subplot(2,2,2); imshow(f1); title('composante rouge');
```

```
subplot(2,2,3); imshow(f2); title('composante verte');
subplot(2,2,4); imshow(f3); title('composante bleue');
```

2.2 Exploration de l'image

Exercice 1

◇ Que valent les valeurs des trois composantes pour le pixel $(245, 50)$? On peut obtenir cette valeur directement par inspection du tableau **f**, ou bien sur la figure avec l'outil **datacursor**. Tester les deux méthodes. Zoomer sur une partir de l'image : que remarque-t-on ?

◇ Tracez des coupes de l'intensité de l'image avec **improfile**. Que peut-on dire (qualitativement) quant à la régularité de l'image ?

◇ Générez une image binaire où l'on met à 1 les valeurs pour lesquelles la composante **R** est prépondérante. Faites de même pour les autres composantes, et afficher le résultat.

Créer à présent une version en niveau de gris, et la sauvegarder dans un fichier (utiliser **rgb2gray** et **imwrite**).

2.3 Quelques transformations géométriques

La Toolbox *Image processing* de Matlab inclut des transformations géométriques comme la rotation ou le changement d'échelle (interpolation).

Exercice 2À partir de l'image *fleur_modif.png*, reconstruire approximativement l'image de départ en vous servant des fonctions **imresize, imcrop** et **imrotate**. Pourquoi oberve-t-on une perte de qualité ?

3 Manipulations d'histogramme

3.1 Contraste

On va modifier le contraste d'une image, défini comme l'écart type. Attention, lorsqu'on manipule les valeurs de niveau de gris, on doit convertir les valeurs **uint8** en **double**. On fera également attention à la conversion et au "scaling" lors de l'affichage. Exécutez les commandes suivantes, et expliquez la différence d'affichage pour les 2 dernières images :

```
f = double(imread('lena_gray.tif'));
f2 = f + 0.5*(f-mean(f(:))); % augmentation du contraste facteur 1.5
figure();imshow(uint8(f));
figure();imshow(uint8(f2));
figure();imshow(f2,[]);
```

Exercice 3

◇ Écrire une fonction **modif_contraste(f,new_mean,new_std)** qui retourne l'image **f** en modifiant sa moyenne et son écart-type (contraste).

◇ Chargez et visualisez les 2 images *IRM1.jpg* et *IRM2.jpg* correspondant à une coupe IRM d'un patient atteint de sclérose en plaque, les 2 IRMs ayant été effectuées à 3 mois d'intervalle. Pour faciliter l'analyse de ces images et de l'évolution sous-jacente il peut être intéressant de visualiser la valeur absolue de la différence des 2 images. Essayez.

◇ Quel problème rencontre-t-on ? Proposez une solution pour améliorer le résultat.

3.2 Égalisation d'histogramme

On calcule et visualise l'histogramme d'une image en nuances de gris avec la commande **hist**. L'histogramme est un outil simple pour ajuster la dynamique d'une image, c'est-à-dire la distribution des niveaux de gris. Comme on l'a vu en cours, on peut facilement modifier l'histogramme

d'une image pour qu'il ressemble à une distribution donnée, en utilisant l'histogramme cumulé (qu'on calcule avec la commande **cumsum**).

Exercice 4

Implémentez les fonctions suivantes, et les tester avec l'image **rock_uneq.ppm**.

◇ **g = egalise(f)** retourne l'image **g**, version égalisée de **f**.

◇ **g = transfert(f,p)** retourne l'image **g**, version modifiée de **f** ayant le même histogramme que **p**.

3.3 Cas des images couleurs

On s'intéresse ici au traitement par histogramme d'images couleurs.

Exercice 5

Peut-on généraliser l'égalisation d'histogramme au cas d'une image couleur ? Proposez une manière de contourner le problème. Testez cette technique sur l'image **mandrill.bmp**.