UNIVERSITÉ DE RENNES 1

TP2 VIS: Manipulation d'images

O. LE MEUR (olemeur@irisa.fr)

2 décembre 2015

But du TP :
– Manipulation d'images sous Matlab.

Travail à rendre :
Le travail est à rendre pour la semaine suivante (voir avec votre encadrant pour définir une date). Pour chaque question, on donnera les commandes et traitements effectués, les images et valeurs obtenues et les commentaires correspondants.

L'ensemble doit être rendu *sous forme électronique*(au format PDF, Word ou Open Office). Ce compte rendu est à envoyer à votre encadrant : `olemeur@irisa.fr`. Si vous travaillez à deux, le nom du fichier doit inclure les noms des binômes.

1 Ajouter une couronne autour de l'image d'entrée

Le travail consiste à ajouter une couronne autour de l'image d'entrée. La taille de la couronne est constance et donnée en paramètre. Par conséquent, si la taille de l'image originale est $[cols, rows]$, l'image de sortie aura une taille de $[cols + 2 \times b, rows + 2 \times b]$. On considère que la valeur de la couronne est nulle (voir fonction *zeros*). Le prototype de la fonction est le suivant *imOut = impad(imIn, b)* :
– *imIn*, l'image à modifier
– b, la largeur de la couronne
L'image de sortie est *imOut*.

En-tête de la fonction :

```
%--------------------------------------------------------------------
% IMPAD - adds zeros to the boundary of an image
%
% Usage:  imOut = impad(imIn, b)
%
% Arguments:     imIn - Image to be padded (greyscale or colour)
%                 b - Width of padding boundary to be added
%
% Returns: imOut - Padded image of size rows+2*b x cols+2*b
%
%--------------------------------------------------------------------
function imOut = impad(imIn, b)
```

2 Supprimer une couronne autour de l'image d'entrée

Le travail consiste à supprimer une couronne autour de l'image d'entrée. La taille de la couronne est constance et donnée en paramètre. Par conséquent, si la taille de l'image originale est $[cols, rows]$, l'image de sortie aura une taille de $[cols - 2 \times b, rows - 2 \times b]$. Le prototype de la fonction est le suivant *imOut = imErase(imIn, b)* :
– *imIn*, l'image à modifier
– b, la largeur de la couronne
L'image de sortie est *imOut*.

En-tête de la fonction :

```
%--------------------------------------------------------------------
% IMERASE - remove the boundary of an image
%
% Usage:  imOut = imErase(imIn, b)
%
% Arguments:     imIn - Image to be padded (greyscale or colour)
%                 b - Width of the boundary to remove
```

```
%
% Returns: imOut - Padded image of size rows-2*b x cols-2*b
%
%---------------------------------------------------------------------
function imOut = imErase(imIn, b)
```

3 Mettre à une valeur prédéfinie les bords d'une image

Le travail consiste à écrire une fonction *imSetBorder* permettant de mettre à une valeur prédéfinie les bords d'une image. Le prototype de la fonction est le suivant *imIn = imsetborder(imIn, b, v)* :

- $imIn$, l'image à modifier
- b, la taille du bord à modifier
- v, la valeur à utiliser pour les bords

L'image de sortie est également $imIn$. Faire attention aux valeurs de paramètes en s'assurant que la valeur b est entière et supérieure à 1. L'image d'entrée peut être couleur.

En-tête de la fonction :

```
%---------------------------------------------------------------------
% IMSETBORDER - sets pixels on image border to a value
%
% Usage:  imIn = imsetborder(imIn, b, v)
%
% Arguments:
%           imIn - image
%            b - border size
%            v - value to set image borders
%---------------------------------------------------------------------
function imIn = imsetborder(imIm, b, v)
```

4 Insérer une image dans une autre image

L'idée est d'insérer une image dans une autre image. La figure 1 illustre le résultat qu'on souhaite obtenir. La spécification de la fonction *implace* est la suivante :

- $im1$ est l'image de destination
- $im2$ est l'image à incruster dans l'image $im1$
- $roff$ et $coff$ représentent le décalage souhaité par l'utilisateur (voir figure 1)
- *newim* est la nouvelle image

On fait l'hypothèse que les deux images d'entrées sont des images couleurs.

En tête de la fonction :

```
%---------------------------------------------------------------------
% IMPLACE - place image at specified location within larger image
%
% Usage: newim = implace(im1, im2, roff, coff)
%
% Arguments:
%
%       im1  - Image that im2 is to be placed in.
%       im2  - Image to be placed.
%       roff - Row and column offset of placement of im2 relative
%       coff   to im1, (0,0) aligns top left corners.
%---------------------------------------------------------------------
function newim = implace(im1, im2, roff, coff)
```

FIGURE 1 – Illustration d'un résultat de la fonction *implace*

5 Automatisation de test

L'idée est d'appliquer la fonction IMPAD à toutes les images d'un répertoire donné. Il faut donc :
- lister toutes les images de type jpg (ou bmp, ou ppm...) d'un répertoire donné ;
- faire une boucle sur toutes les images et appliquer la fonction IMPAD ;

Les fonctions utiles sont :

```
imglist = dir([directoryOrigPict '*.bmp']);
fnum    = length(imglist);
```

La première instruction permet de lister toutes les images de type bmp du répertoire directoryOrigPict. Le nom des images est dans la structure imgList. Pour accéder au i^{ieme} nom de fichier, il faut faire imgList(i).name.

La deuxième fonction permet de connaitre le nombre d'images de type bmp dans le répertoire.

Ecrire le script permetttant d'automatiser le test IMPAD.