



## Module VIS-AMR - Master MITIC TP2 : Manipulation d'images

O. LE MEUR & A. ROUMY  
olemeur@irisa.fr

2011-2012

### 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](mailto:olemeur@irisa.fr) ou [aline.roumy@inria.fr](mailto:aline.roumy@inria.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 constante 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 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 paramètres 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(imIn, b, v)
```

### 3 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)
```

### 4 Création d'une image de bruit avec une décroissance fréquentielle en $1/f^\alpha$

La figure 2 illustre le type d'images qu'on souhaite obtenir. L'idée est ici de se familiariser avec les transformées de Fourier. Analyser le code suivant et créer le filtre *filter* afin de filtrer le spectre fréquentiel du bruit. On appliquera sur le spectre de l'image de bruit un filtre passe-base (on utilisera la distance euclidienne d'un point par rapport au centre du spectre pour représenter *f*).

```
%-----  
% NOISEONF - Creates 1/f spectrum noise images.  
%  
% Function to create noise images having 1/f amplitude spectrum properties.  
% When displayed as a surface these images also generate great landscape  
% terrain.  
%
```

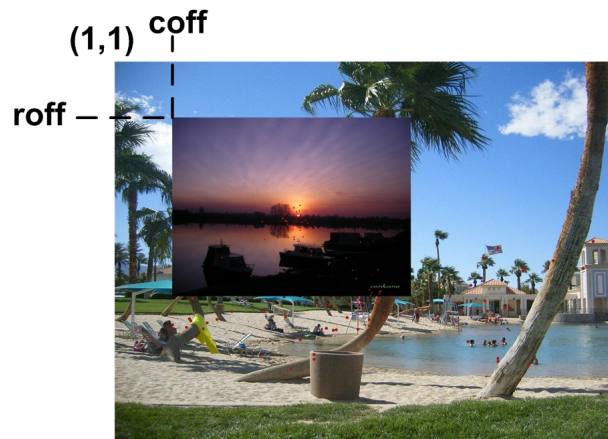


FIGURE 1 – Illustration d'un résultat de la fonction *implace*

```

% Usage: im = noiseonf(size, factor)
%
%     [rows cols] - indicate the size of image to produce
%     factor - controls spectrum = 1/(f^factor)
%
%     factor = 0   - raw Gaussian noise image
%               = 1   - gives the 1/f 'standard' drop-off for 'natural' images
%               = 1.5 - seems to give the most interesting 'cloud patterns'
%               = 2 or greater - produces 'blobby' images
%-----
function im = noiseonf(row, col, factor)

im = randn(rows,cols); % noise picture
imfft = fft2(im);      % Take fft of image.
imfft = fftshift(imfft); % Shift 0 frequency to the middle.
mag = abs(imfft);
phase = angle(imfft);
real = mag .* cos(phase);
imaginary = mag .* sin(phase);

% Create two matrices, x and y. All elements of x have a value equal to its
% x coordinate relative to the centre, elements of y have values equal to
% their y coordinate relative to the centre. From these two matrices produce
% a radius matrix that gives distances from the middle

<-- Put your code here -->
% the output is a filter (1/f^alpha)

% Reconstruct fft of noise image, but now with the specified amplitude spectrum
newfft = (real + 1i .*imaginary);
im = abs(ifft2(fftshift(filter .*newfft))); % Invert to obtain final noise image
caption = sprintf('noise with 1/(f^{%.1f}) amplitude spectrum',factor);
imagesc(im), axis('equal'), axis('off'), title(caption);

```

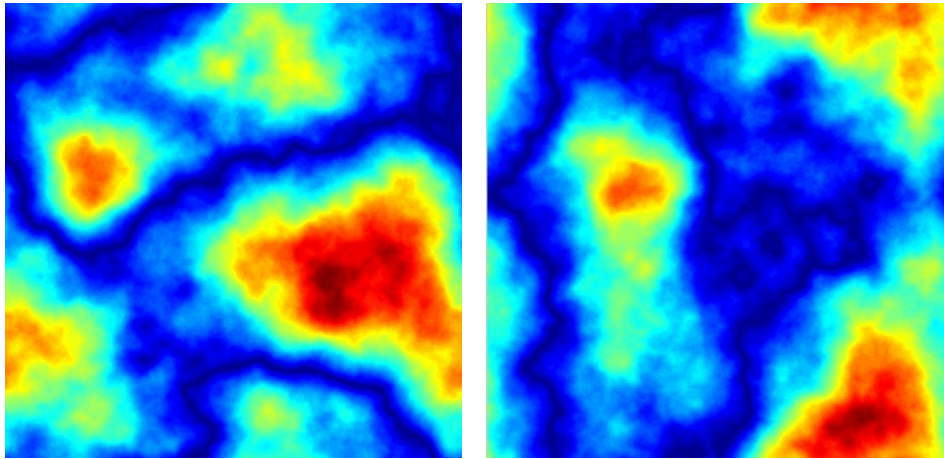


FIGURE 2 – Illustration d'un résultats de la fonction *noiseonf*