MASTER EEEA --- 1ERE ANNEE

TRAITEMENT DU SIGNAL

ÉNONCE DE TRAVAUX PRATIQUES

JULIEN MAROT ET OUSSAMA DJEDIDI
MARC ALLAIN

TABLE DES MATIÈRES

1. OPERATIONS PONCTUELLES

1.0 Introduction

On s'intéresse dans ce TP au réhaussement d'image, c'est-à-dire à l'amélioration de l'aspect visuel (essentiellement le contraste) d'une image, par des opérations ponctuelles (pixel à pixel).

Les savoirs et savoir faires à acquérir lors de ce TP sont :

- Savoir transcrire une méthode de traitement d'image en un script Matlab.
- Appliquer la méthode programmée dans diverses conditions (images, etc.).
- Interpréter les résultats de traitement obtenus, comparer plusieurs opérations ponctuelles selon un critère subjectif (l'aspect visuel de l'image transformée).

1.1 Structure du programme de traitement d'image

Choisissez un espace de travail sur le disque réseau (et nom sur le bureau car il n'est pas sauvegardé). Créez un dossier où seront stockés les différents fichiers fournis et créés lors de ce TP (ce dossier pourra par exemple s'appeler `Nom1_Nom2_TP_Image_1` où « `Nom1` » et « `Nom2` » correspondent évidemment aux noms de votre binôme).

Dans ce dossier, vous créerez ensuite les sous-dossiers `Images`, `Scripts` et `Resultats` ; ces dossiers contiendront, respectivement, les images à traiter, les scripts Matlab écrits et les résultats de traitement. Dans le dossier `Images`, copiez dans l'ensemble des images utilisées pour ce TP (et que vous demanderez à l'encadrant). Dans le dossier `Nom1_Nom2_TP_Image_1`, créez les fichiers Matlab `main.m` et `setup.m`. Dans le dossier `Scripts`, créez le fichier `display_results.`

* Le fichier `setup.m` permettra de créer en début de programme les variables nécessaires au bon déroulement des programme (notamment, le chemin pour trouver les scripts de traitement et les images à traiter).

* Le fichier `main.m` est le script principal qui lance les scripts de configuration et de traitement. Il effectue également la visualisation et la sauvegarde des résultats.

* Le fichier `display_results.m` permet de faire l'affichage des images et des histogrammes que vous allez obtenir tout au long du TP.

Le fichier `main.m` a la structure suivante :

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Nom           : main.m
%%
%% Description   : Script de lancement de techniques de rehaussement
%% d'image. Ces scripts permettent : la construction d'histogramme, la
%% correction gamma, l'égalisation d'histogramme.
%%
%% Auteur        : Julien Marot
%% Date          : 8/10/2012
%% Version       : 1.0
%% Historique    : création (8/10/2012)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%  On efface l'espace de travail...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all

%% PARTIE 1 : Script de configuration...
%%        Definition des variables d'environnement
%%        Defintion du nom de l image a traiter
%%        Definition du nombre de niveaux de gris
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setup


%% PARTIE 2 : Chargement de l'image a traiter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
out                     = load(nom_image);
image_a_traiter         = double(out.A) ;


%% PARTIE 3 : Rehaussement d'image par méthodes point à point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sortie1                 = histogramme(image_a_traiter,NNdG)
sortie2                 = correction_gamma(image_a_traiter,NNdG)
sortie3                 = nom_traitement(image_a_traiter,NNdG)

%% PARTIE 4 : affichage et sauvegarde…
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

display_results
```

Nous décrivons maintenant les différentes parties de ce script. Le fichier `setup.m` permet de spécifier les chemins, le nom de l'image à traiter et le nombre de bits sur lequel l'image est codée. Un exemple de fichier est donné ci-dessous.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Nom           : setup.m
%% Description   : Script de configuration permettant de définir les
%%                 chemins utilisés par les script de traitement d'image.
%% Auteur        : Julien Marot
%% Date          : 8/10/2012
%% Version       : 1.0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%  Chemin vers les images à traiter...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
addpath ??????????????????????????? (A COMPLETER)

%%  Nom de l'images à traiter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nom_image        = 'Asteroid.mat'

%% Nombre de NdG a considérer sur l'image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NNdG             = 2^8;
```

On effectue ensuite le chargement de l'image à traiter avec la fonction Matlab `load`. Puis, on effectue un certain nombre d'opérations sur l'image par des fonctions dédiées. Enfin, on effectue l'affichage des résultats des traitements.

> **Exercice 1** : Rédigez et compléter les scripts `main.m` et `setup.m` en vous assurant de bien en comprendre les différentes étapes.

1.2 Construction des fonctions de rehaussement d'images

L'objectif principal de ce TP est de créer les fonctions qui permettent d'effectuer les différents traitements appelés par le script `main.m`. Ces opérations sont les suivantes

(a) la construction d'un histogramme (`histogramme.m`),

(b) la correction gamma (`corr_gamma.m`),

(c) l'égalisation d'histogramme (`egalisation_hist.m`),

(d) l'ajustement de dynamique (`ajust_dyn.m`),

(e) le seuillage d'histogramme (`seuillage_hist.m`),

(f) le moyennage (`moyennage_image.m`).

On s'intéresse tout d'abord à la construction de l'histogramme. La structure de ce code, et certains éléments qui le constituent, serviront également dans les autres fonctions.

Construction d'histogramme

Avant de construire la fonction histogramme, nous allons créer une fonction qui permet de quantifier une image en nombres entiers, ceci sur un nombre de niveaux de gris spécifié en entrée. Le prototype de cette fonction sera donc le suivant :

```
function image_out = quantification_image(image_in,NNdG)

%% image_out = quantification_image(image_in,NNdG)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Nom           : quantification_image.m
%%
%% Description   : fonction permettant de quantifier une image sur NNdG niveaux.
%%
%% Entrées       : image_in (matrice NxN) = image d'entrée non quantifiée,
%%                 NNdG     (scalaire)    = nombre de NdG pour la quantification
%%
%%  Sortie       : image_out(matrice NxN) = quantifiée sur NNdG niveaux
%%
%% Auteur        : Julien Marot
%% Date          : 8/10/2012
%% Version       : 1.0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%% (A COMPLETER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

On souhaite que l'image `image_in` soit transformée en nombres entiers de façon à ce que `image_out` prennent ses valeurs entre `0` et `NNdG`. Cette fonction utilisera la fonction `Dilatation_Valeurs_Donnee.m`, qui échelonne les valeurs d'un ensemble de données entre 0 et une valeur maximum, vue en CAO. Chaque traitement qui sera appliqué le sera en précision double, qui nécessite 64 bits, soit deux mots de 32 bits.

Exercice 2 : complétez et validez le programme ci-dessus.

L'histogramme a été vu en cours. Nous rappelons qu'il s'agit d'un graphe donnant la proportion de chacun des niveaux de gris présent dans l'image. Le prototype de la fonction est donné ci-dessous.

```
function h = histogramme(image_in,NNdG)

%% h = histogramme(image_in,NNdG)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Nom           : histogramme.m
%%
%% Description   : fonction permettant de construire l'histogramme d'une image.
%%
%% Entrées       : image_in (matrice NxN) = image d'entrée non quantifiée,
%%                 NNdG     (scalaire)    = nombre de NdG pour la quantification
%%
%%  Sortie       : h        (vecteur Nx1) = liste des proportions des NdG
%%
%% Auteur        : Julien Marot
%% Date          : 8/10/2012
%% Version       : 1.0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%  Quantification de l'image d'entrée sur NNdG niveaux...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
image_in_q = quantification_image(image_in,NNdG)

%%  Construction de l'histogramme (A COMPLETER)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercice 3 : complétez et validez le programme ci-dessus. Attention, rappelez-vous que l'histogramme a une somme normalisée à 1.

Testez votre fonction sur différentes images et interprétez vos résultats.

On souhaite créer une fonction qui affiche une image et son histogramme sur la même figure. Cela facilitera le travail d'affichage par la suite.

```
function affiche_image_histogramme(num_figure,image_in,titre,NNdG)

%% affiche_image_histogramme(num_figure,image_in,titre,NNdG)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Nom          : affiche_image_histogramme.m
%%
%% Description :    Affichage d'une image et de son histogramme, enregistrement
%%                  sous format bmp.
%%
%% Entrées      : num_figure (scalaire) = numéro de la figure pour l'affichage
%%                image      (matrice NxN) = matrice image d'entrée à afficher,
%%                Titre    (chaîne de caractère) = nom du fichier de sauvegarde
%%                NNdG     (scalaire) = nombre de NdG pour la quantification
%%
%% Auteur    : Julien Marot
%% Date      : 8/10/2012
%% Version   : 1.0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%(A COMPLETER)

Name_Print_Bmp =['.\Resultats\',titre,'.bmp'];
Valeurs_Ng      = [0:1:??];
H               = Histogramme(image_in,NNdG);

% Affichage de l'image
figure(??)
subplot(211)
image(image_in)
colormap(gray(??)), colorbar, axis image, xlabel('Pixel'), ylabel('Pixel')
title(??)

% Affichage de l'histogramme
subplot(212)
stem(??,??),
axis tight, xlabel('Niveau de gris'), ylabel('Proportion')

imwrite(image_in/max(image_in(:)),Name_Print_Bmp)
```

Exercice 4 : complétez et validez le programme ci-dessus. Testez votre fonction sur une image au choix en appelant la fonction `affiche_image_histogramme` dans le programme `display_results`. Vous afficherez vos images résultats au fur et à mesure en utilisant la fonction que vous venez de créer.

Correction gamma

Soit I_{out} l'image que vous obtiendrez par rehaussement gamma de l'image d'entrée I_{in}. La relation entre ces deux images est la suivante :

$$I_{out} = C\ (I_{in})^{\gamma}$$

où C et γ sont deux constantes positives.

> **Exercice 5** : Implantez cette méthode sous la forme d'une fonction `corr_gamma.m`. Vous prendrez soin de bien définir les entrées et les sorties de cette fonction. L'image en sortie devra être codée sur le nombre de niveaux de gris `NNdg`.
>
> Définissez une valeur adéquate de γ. Visualisez l'histogramme des images à traiter et rehaussez en conséquence. Interprétez vos résultats.

Egalisation d'histogramme

L'objectif est ici d'apprécier l'effet de l'égalisation d'histogramme sur l'aspect d'une image. Nous rappelons que l'égalisation d'histogramme nécessite que vous utilisiez l'histogramme cumulé. Pour cette opération, vous pourrez utiliser la fonction de calcul d'histogramme programmée plus haut ainsi que la fonction `cumsum.m` de Matlab.

> **Exercice 6** : Implantez cette méthode sous la forme d'une fonction `egalisation_hist.m`. L'image en sortie devra être codée sur le nombre de niveaux de gris `NNdg`.
>
> Visualisez l'histogramme des images avant et après traitement. Interprétez vos observations.

Ajustement de dynamique

L'ajustement de dynamique consiste à redéfinir les valeurs minimale et maximale de l'image d'entrée avant de la quantifier sur `NNdG` niveaux. Ce faisant, la dynamique effective est affectée à une plage de valeurs plus restreinte mais généralement plus utile.

> **Exercice 7** : Retrouvez dans le cours la transformation ponctuelle qui correspond à cet ajustement de dynamique. Implantez cette méthode sous la forme d'une fonction `ajust_dyn.m` ; vous utiliserez la fonction `quantification_image.m` déjà construite. L'image en sortie devra être codée sur le nombre de niveaux de gris `NNdg`.
>
> Visualisez les images et leurs histogrammes avant et après traitement. Pour quels types d'images cette transformation est-elle utile ?

Seuillage d'histogramme

Le seuillage d'histogramme définit un seuil qui permet de construire une image binaire.

> **Exercice 8** : Implantez cette méthode sous la forme d'une fonction nommée `seuillage_hist.m`. L'image en sortie devra être codée sur le nombre de niveaux de gris `NNdg`.
>
> Visualisez les images et leurs histogrammes avant et après traitement. Dans quels cas de figure cette transformation est-elle utile ?

Moyennage d'images

L'objectif est ici d'étudier l'effet de réduction du bruit par moyennage d'images.

> **Exercice 9** :
>
> Pour un niveau de bruit `niveau_bruit` donné et une image de votre choix, générez `nb_real` réalisations bruitées à partir de la commande Matlab
>
> ```
> Image_Bruitee = Image_a_traiter+Niveau_Bruit*(NNdG-1)*randn(size(Image_a_traiter);
> ```
>
> Implantez ensuite le calcul de l'image moyenne sous la forme d'une fonction nommée `moyennage_image.m`.
>
> Visualisez l'image moyennée et son histogramme. Faites de même avec l'une des images bruitées. Que constatez-vous ? Dans quels cas de figure cette transformation sera-t-elle mise en échec ?

1.3 Commentaires qualitatifs

Quelle est selon vous l'opération qui conduit à un meilleur rendu visuel ? Ce constat est-il valable pour toutes les images ?

2. MATLAB ET LA TRANSFORMEE DE FOURIER

2.0 Introduction

L'objectif de ce TP est double. Il s'agit tout d'abord de vous faire manipuler les fonctions MATLAB permettant le calcul de la transformée de Fourier (TF) et de son inverse. D'autre part, nous chercherons à observer la signature dans le spectre de Fourier de certaines structures visibles dans le domaine spatial.

2.1 Méthodes de visualisation de la TFD

Note Matlab 1 – On notera que sous Matlab le *module* et la *phase* d'une composante complexe s'obtient respectivement par la fonction **<abs>** et **<angle>**.

Note Matlab 2 – La commande **<fft2>** calcule la TFD d'une image et la fonction **<ifft2>** calcule la TFD inverse.

Note Matlab 3 – Vous utiliserez les fonctions de visualisation <**imagesc>** et **<mesh>** pour afficher les images et la fonction **<subplot>** qui permet de tracer plusieurs images sur la même figure. Vous noterez l'existence de la fonction **<fftshift>** qui intervertit les cadrans d'une matrice et permet ainsi de « recentrer » le spectre avant l'affichage.

Exercice 1 :

Chargez l'image **visage.mat**; l'image est dans la variable **img**. Sous Matlab, calculez le spectre de **visage.mat** et affichez le *avec* et *sans* centrage. Comme la composante harmonique (0,0) domine les autres valeurs, vous la supprimerez pour mieux observer le spectre. Vous expliquerez dans votre rapport pourquoi l'harmonique (0,0) est réelle et contient bien la somme des pixels ; vous utiliserez enfin la fonction **<sum>** pour le vérifier.

Exercice 2 :

L'image contenue dans **visage.mat** est constituée de plusieurs gaussiennes. Sachant que la transformée de Fourier (TF) d'une gaussienne est également une gaussienne (*cf.*, Sec. 3.3), commentez le spectre de l'image **visage.mat**.

Exercice 3 :

Expliquez aussi pourquoi le fait d'enlever la valeur moyenne d'une image revient à mettre à zéro l'harmonique (0,0). Vous vérifierez que c'est effectivement le cas à l'aide de la fonction **<mean>**.

Exercice 4 :

Il est souvent plus facile d'observer un spectre **y** en visualisant sa transformation par :

>> visuF = log(1+abs(y))

Essayez cette transformation pour afficher le spectre. Vous noterez qu'elle permet de mieux voir les composantes spectrales de faible amplitude.

2.2 Périodicité implicite et rotation

Exercice 5 : observation...

Lisez une image de réseau coronarien **coroav1.mat** dont le nom de la variable est **seqb24**. Affichez son spectre et observez les directions principales du spectre. Comment vous renseignent-elles sur l'orientation de l'arbre coronarien ? Vous remarquerez que le spectre comporte une croix. C'est un artefact dû au fait que l'image est supposée périodique.

Exercice 6 : fenêtre de ponderation...

Pour pallier à ce problème, il est courant d'effectuer un fenêtrage de l'image, c.à.d., de multiplier l'image par une fonction 2D ajustée de façon a s'annuler sur les bords. Par exemple, pour mettre en oeuvre un fenêtrage de *Hamming,* tapez :

```
>>fenetre1D = hamming(256);                % Const. fenetre de Hamming 1D
>>[fenetre_x,fenetre_y] = meshgrid(fenetre1D); % Const. fenetre de Hamming 2D
>>fenetre2D =fenetre_x.*fenetre_y;
>>seqb24_fen = seqb24.*fenetre2D;          % Ponderation...
```

Visualisez la construction de la fenêtre aux différentes étapes, puis visualisez le nouveau spectre et commentez le résultat dans votre rapport.

Exercice 7 :

Faites une rotation de cette image à l'aide de la fonction **<imrotate>**. Visualisez le spectre et commentez le résultat dans votre rapport.

2.3 Écart-types spatiaux *vs*. écart-types spectraux

Nous illustrons ces propriétés en étudiant les caractéristiques spectrales des gaussiennes bi-dimensionnelles. La fonction gaussienne normalisée (c.-à-d. de surface unitaire et centrée à l'origine) est ainsi définie :

$$g(x) = \frac{1}{\sigma_x \sqrt{2\pi}} \exp(-\frac{1}{2\,{\sigma_x}^2})$$

où σ_x est l' « écart type », c.-à.-d. un paramètre de la fonction qui contrôle l'étalement de la fonction autour de l'origine. Pour étudier la TF d'une gaussienne, il est utile de se référer à la fonction

$$f(x) = \exp(-\pi\, x^2)$$

dont la transformée de Fourier est de forme identique, soit :

$$f(u) = \exp(-\pi\, u^2)$$

Exercice 8 :

En utilisant la propriété de « mise à l'échelle » de la TF, démontrez que la TF de *g*(*x*) est également une gaussienne dont l'écart type σ_u vaut $\sigma_u = (2\,\pi\,\sigma_x)^{-1}$.

Exercice 9 :

Démontrez que la *largeur à mi-hauteur* de la gaussienne notée est reliée à l'écart type σ_x par la relation .

Exercice 10 :

En deux dimensions, la gaussienne normalisée (c.-à-d. de volume unitaire) a la forme :

$$g_2(x,y) = g(x)g(y) = \frac{1}{2\pi\ \sigma_x\ \sigma_y} \exp - \left(\frac{1}{2\ {\sigma_x}^2} + \frac{1}{2\ {\sigma_y}^2}\right)$$

La fonction g_2 étant séparable, démontrez que sa transformée l'est aussi, c.-à-d. qu'elle est égale au produit *G*(*u*)*G*(*v*), les transformées respectives de *g*(*x*), *g*(*y*). Déduisez-en l'expression de la TF de g_2.

Exercice 11 :

Note Matlab 4-- Pour afficher des fonctions dont on connaît l'expression analytique, on utilisera d'abord la fonction **<meshgrid>** qui permet de générer un plan croissant en x et un plan croissant en y. Par exemple, si on souhaite afficher $L(x,y) = x^2+y^2$, il suffit de taper **[x,y] = meshgrid (-32:31,-32:31);** puis directement **output = u.^2 + v.^2;** pour calculer la fonction sur la grille générée.

Pour étudier une gaussienne avec Matlab, nous construisons une gaussienne bidimensionnelle centrée à l'origine et d'écart type **s_x** et **s_y** de, respectivement, 3 pixels en *x* et 6 pixels en *y.*

```
>>[x,y]=meshgrid(-32:31,-32:31);                    % Génération maillage
>>gauss=exp( - ( x.^2/(2*sigmax^2) + y.^2/(2*sigmay^2) ) );   % Calcul de la gaussienne
>>gauss=fftshift(gauss);                          % On recentre sur origine Matlab
>>Tfgauss=fft2(gauss);                            % calcul de la TF
```

Tracez **TFgauss** selon les axes *u* et *v* et observer l'écart-type de cette fonction selon ces axes. Estimez visuellement la largeur à mi-hauteur selon les axes *u* et *v*. et vérifiez la relation ci-dessus. Refaites cette vérification pour un σ_x plus grand (7 pixels) et un σ_y plus grand (10 pixels). Qu'observez-vous sur les écarts type dans le domaine fréquentiel σ_u et σ_v ? Commentez.

Exercice 12 :

Note Matlab 5-- On notera que le calcul de la phase devient imprécis lorsque l'amplitude d'une harmonique est faible. Il convient alors de masquer les composantes de phase qui seraient calculées avec des harmoniques d'amplitude trop faible (par exemple .001 dans cet exemple).

Note Matlab 6-- La fonction déduite de **<angle>** est souvent en « dent de scie » car elle calcule la phase entre –pi et +pi. On peut en général rétablir une fonction de phase plus lisse avec la fonction **<unwrap>**.

Pour étudier une gaussienne décentrée, construisez une gaussienne bidimensionnelle d'écart-types identiques mais **décalée** de 3 pixels en *x* et de 5 pixels en *y*. Visualisez les spectres des deux fonctions. Trouvez la position (*ligne,colonne*) des maxima et commentez. Visualisez la phase avec la fonction **<angle>**. Comparez les maxima de la phase pour les deux gaussiennes et commentez cette comparaison. Quelle est la forme de la phase pour la gaussienne décalée ? Comparez également les pentes en *u* et celle en *v*.

3. DETECTION DE CONTOURS PAR LAPLACIEN DISCRET

3.0 Introduction

L'objectif de ce TP est d'introduire et d'analyser les outils de détection de contour dans les images basés sur la différentiation numérique. Les aspects de robustesse au bruit seront notamment soulignés.

3.1 Analyse des propriétés du laplacien discret

Note Matlab 1-- L'opérateur de convolution numérique 2D est implémentée avec la fonction **<conv2>**. Vous noterez que la taille de l'image en sortie ne correspond pas à celle de l'image d'entrée car il faut tenir compte des « effets de bord », *cf*. cours.

Note Matlab 2-- La génération d'un bruit uniforme est implémentée par la fonction **<rand>** ainsi, l'instruction **rand(64)** permet de générer une image 64x64 dont chaque point correspond à la réalisation d'une variable aléatoire uniforme tirée sur [0,1].

Le théorème de convolution est ainsi formulé :

$$TF(f * g) = F \,.\, G$$

où * désigne l'*opération de convolution* et avec *F* et *G* les TF de *f* et *g,* respectivement. On illustre ce théorème en étudiant les propriétés d'un opérateur discret, *c.-à-d*. d'un opérateur (numérique) linéaire noté **lap_d** et défini par

$$lap_d = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Comme cette formulation ne permet pas un traitement « isotrope » des images, on préfère souvent d'autres formulations de laplacien discret qui le permettent,

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \text{ ou } \begin{pmatrix} -1 & -2 & -1 \\ -2 & 4 & -2 \\ -1 & -2 & -1 \end{pmatrix}$$

L'opérateur laplacien a la propriété d'amplifier les transitions dans l'image, c.-à-d. les changements de niveau de gris. Il peut donc servir à extraire les arêtes des objets.

<u>Exercice 1.1 :</u> forme analytique de la fonction de transfert

On appelle « *gain en continu* » la somme des termes de l'opérateur. On remarquera que pour les opérateurs ci-dessus, ce gain est nul. Que cela signifie t-il si l'opérateur agit sur une image de niveau de gris uniforme ? Calculez la transformée en *Z* du laplacien discret et faites la substitution :

$$z_1 = \exp(j\, 2\, \pi\, u\, \Delta_x)$$
$$z_2 = \exp(j\, 2\, \pi\, u\, \Delta_y)$$

Vous obtiendrez ainsi une fonction de **(***u,v***)** que vous noterez $L_d(u,v)$. Cette fonction est la *fonction de transfert* (FT) *du laplacien discret*. Comparez cette FT à celle de l'opérateur continu dont l'expression est donnée en cours (vous tracerez notamment ces deux expressions sous Matlab).

Exercice 1.2 : évaluation numérique de la fonction de transfert

On peut observer la FT d'un opérateur dicret quelconque en effectuant la TFD d'une image de la réponse à l'impulsion de l'opérateur (ici, du laplacien discret) :

```
>> repcentre = zeros(64);                    % Initialise l'image
>> repcentre(32:34,32:34) = lap_d;           % Insère le laplacien
>> repimp=fftshift(repcentre));              % Centrer sur l'origine Matlab
```

Calculez et visualisez le spectre d'amplitude de repimp. Comparez ce spectre à celui calculé à la question précédente.

3.2 Mise en œuvre et robustesse du laplacien discret

Exercice 2 : détection de bords (1)

Il s'agit d'observer l'effet de l'opérateur sur une image d'un carré blanc (niveau 60) sur fond noir (niveau 0). Construisez une telle image avec la fonction **<carre_fo>** en choisissant un carré 32x32 sur fond 64x64 ; on notera par la suite cette image **carres**. Convoluez **carres** avec le laplacien discret en prenant soin de ne conserver qu'une partie de la convolution pour conserver la taille initiale. Affichez ensuite les deux images côte à côte et observez l'effet. Commentez.

Exercice 3 : détection de bords (2)

Convoluez l'image **carres** par un autre carré de taille 9 x 9; on notera par la suite cette image **carfilt**. Quel est le gain continu du filtre que vous venez d'utiliser ? Commentez le résultat de cette opération. Convoluez ensuite **carfilt** avec le laplacien et commentez le résultat.

Exercice 4 : immunité au bruit

Ajoutez du bruit à **carfilt** et refaites la convolution avec le laplacien. Observez le résultat et noter qu'il est difficile d'identifier la frontière, alors qu'à l'œil nu on peut très bien la voir sur l'image bruitée. Expliquez le problème.

4. FILTRE DE MARR-HILDRETH

4.0 Introduction

Le TP précédent a insisté sur la nécessité de construire un détecteur de bord numérique qui soit robuste au bruit présent dans l'image. De manière standard, un tel détecteur de bord peut être conçu en lissant au préalable l'image bruitée. C'est le cas du filtre de Marr-Hildreth vu en cours et que nous étudions dans ce TP.

4.1 Étude du filtre de Marr-Hildreth

Le filtre de Marr-Hildreth (MH) est un filtre dont la réponse à l'impulsion est décrite par la relation

$$h_{MH}(x,y) = \nabla^2 \exp -\left(\frac{x^2}{2\,\sigma_x{}^2} + \frac{y^2}{2\,\sigma_y{}^2}\right) \text{ avec } \nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$$

où σ_x et σ_y sont respectivement l'écart type en *x* et en *y* du filtre. Ce filtre peut être représenté comme la mise en cascade de deux filtres linéaires : l'image *f*(*x*,*y*) est d'abord filtrée par un filtre gaussien passe-bas pour produire une image $f_1(x,y)$, que l'on soumet au laplacien dont la propriété est de faire ressortir les frontières des objets. Le laplacien étant un filtre passe-haut, *la mise en cascade produit un filtre passe-bande bidimensionnel*. Dans cet exercice, nous allons d'abord étudier les propriétés de ce filtre, en particulier l'écart type de la réponse impulsionnelle gaussienne, puis observer que son écart type sert à adapter le laplacien qui suit à la détection des frontières des objets de taille (et d'orientation) particulière.

Notre premier objectif est d'observer que le filtre de MH est un filtre *passe-bande* de « fréquence centrale » distribuée selon une ellipse dans le plan de Fourier. Le laplacien d'une gaussienne d'écart type σ_x et σ_y est un filtre dont la fonction transfert est :

$$H_{MH}(u,v) = (u^2+v^2)\exp -\left(\frac{u^2}{2\,\sigma_u{}^2} + \frac{v^2}{2\,\sigma_v{}^2}\right)$$

Exercie 1 :

En utilisant le lien existant entre écart-types spatiaux et fréquentiels, donnez une autre expression de la fonction de transfert ci-dessus. Déterminez analytiquement la position en *u* du maximum du passe-bande lorsque $\sigma_x = \sigma_y$; notez que cette position est justement fonction de ces écarts types.

Exercice 2 :

La fonction <**marr>** présentée en Annexe 2 réalise le filtrage de MH, en spécifiant σ_x et σ_y en pixels ; la fonction est en librairie et peut être appelée directement. Nous allons l'utiliser pour observer la fonction transfert du filtre. Pour cela, il s'agit simplement de filtrer une impulsion à l'origine, puis de faire la transformée de Fourier de cette réponse. Observer le comportement passe-bande du filtre avec $\sigma_x = \sigma_y = 4$ et $\sigma_x = \sigma_y = 8$. Tracez une tranche centrale du filtre et déterminez la position du maximum. Vérifiez que cette position est bien celle prédite analytiquement.

4.2 Détection de contours

Nous commençons l'étude de ce filtre par la biais d'une image synthétique. Plus précisément, il s'agit de quatre carrés de tailles diverses, filtrés passe-bas par un filtre uniforme, sont disponibles pour ces observations dans le fichier **enscarfl.mat**.

Exercice 3 : compromis précision/robustesse

Observez d'abord le comportement du filtre de MH lorsqu'il n'y a pas de bruit, et ceci pour trois écarts types, dont l'un, très faible, approxime le laplacien seul [vous expliquerez pourquoi]. Observez ensuite que le plus petit carré est très peu visible pour l'écart-type le plus important. Donnez une explication de ce phénomène. Ajoutez du bruit avec un écart-type de 500 et observez l'effet de cette perturbation sur la détection des frontières. Notez aussi les effets de périodicité de l'image.

Exercice 4 : détection de bords sur une image réelle

Utilisez l'image **coroav1.mat** pour étudier l'effet de σ_x et σ_y sur la détection des frontières. Affichez l'image et déterminez la largeur approximative de l'artère, puis utilisez cette valeur pour préciser les écart-types qui seraient adaptés à la détection des contours. Filtrez :

```
>> img = marr(.....);
```

On obtient une image représentant les contours seuls, en détectant les passages par zéro de **img** ; pour y parvenir, on effectuera d'abord une image binaire :

```
>> imgbin = img<0;
```

Essayez quelques valeurs d'écarts types pour observer leurs effets. On obtient les frontières en déterminant les pixels pour lesquels la norme du gradient de **imgbin** est non nulle

```
>> frontiere = abs(gradient(double(imgbin))) ~= 0;
```

On pourra alors superposer les frontières à l'image originale :

```
>> affiche(seqb24.*(~frontiere) + frontiere * 255);
```

Le symbole **~** signifie « **not** » et inverse la logique de frontière pour laisser l'image intacte partout sauf à la frontière.

4.3 Application au calcul de la fraction ventriculaire

La fraction d'éjection ventriculaire est la quantité relative de sang éjecté lors d'une contraction **(***volume en fin de diastole* - *volume en fin de systole*)/(*volume en fin de diastole***)**. Elle exprime l'efficacité de la pompe cardiaque à faire circuler le sang à chacune des contractions. Une valeur normale est de 0.75. On détermine la fraction d'éjection à l'aide de mesures géométriques de la cavité ventriculaire au début et à la fin de la contraction et de mesures de la section du ventricule en fin de systole et en fin diastole. À partir de ces mesures et en faisant appel à un modèle géométrique approprié (ellipsoïde), il est possible d'estimer les volumes correspondants, donc la fraction d'éjection (l'annexe 3 donne des détails sur le sujet). Les filtres de Marr-Hildreth peuvent être utiles pour automatiser cette procédure.

Exercice 5 :

Lisez les images **syst90.mat**, **diast90.mat** et **fd90.mat** qui correspondent, respectivement, à la systole et à la diastole cardiaque et au fond. Dans un premier temps, obtenir une image plus nette des cavités cardiaques en soustrayant l'image de systole (et diastole) de l'image du fond. Sur les images résultantes, déterminez l'intérieur du ventricule et son contour avec la même procédure que celle utilisée pour les coronaires (avec σ_x plus élevé toutefois). Obtenez une image binaire montrant l'intérieur des cavités ventriculaires (cf. annexe 3). Estimez la surface en calculant le nombre de pixels qui appartiennent à l'intérieur du ventricule ; vous utiliserez pour cela la fonction **<sum>**. Évaluez la longueur du grand axe ventriculaire. Estimez la fraction d'éjection.