# Travaux pratiques de Traitement d'Images sous MATLAB
# Etude de la transformée de Fourier discrète bidimensionnelle FFT2
# Simulation de l'expérience d'Abbe et Porter
# Etude de la tranformée en cosinus discrète bidimensionnelle DCT2 et de la compression JPEG

Année 2015

## 1 MATLAB

MATLAB (« matrix laboratory ») est un langage de programmation émulé par un environnement de développement du même nom; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran. Les utilisateurs de MATLAB (environ un million en 2004) sont de milieux très différents comme l'ingénierie, les sciences et l'économie dans un contexte aussi bien industriel que recherche. Matlab peut s'utiliser seul ou bien avec des toolbox (« boîtes à outils »).
L'organisation IEEE a établi un classement de la popularité des langages de programmation en 2014; MATLAB se classe dans le TOP 10 [1]. Pour ce classement, IEEE [2] prend en compte le nombre de références sur Twitter [3], le nombre de nouveaux projets sur GitHub [4], le nombre de questions concernant le langage sur StackOverflow [5], les références sur Hacker News et Reddit [6] et le nombre d'offre d'emplois sur différents sites demandant une compétence dans ces langages.
MATLAB est un langage orienté vers la programmation "vectorielle" et "matricielle" , donc particulièrement bien adapté au traitement d'images. Il est souvent possible d'écrire du code MATLAB de façon extrêmement concise sans boucle for. C'est ce que vous vous efforcerez de faire durant ce TP.

1. 1. Java , 2. C , 3. C++ , 4. Python , 5. C # , 6. PHP , 7. Javascript , 8. Ruby , 9. R , 10. MATLAB

2. L'Institute of Electrical and Electronics Engineers est la plus grande association professionnelle du monde pour le développement de la technologie. Elle compte plus de 400.000 membres répartis dans plus de 160 pays.

3. Twitter est un outil de microblogage géré par l'entreprise Twitter Inc. Il permet à un utilisateur d'envoyer gratuitement de brefs messages, appelés tweets (« gazouillis »), sur internet, par messagerie instantanée ou par SMS. Ces messages sont limités à 140 caractères. Le service est rapidement devenu populaire, jusqu'à réunir plus de 500 millions d'utilisateurs dans le monde fin février 2012, dont environ 200 millions utilisant Twitter au moins une fois par mois

4. GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. En plus d'offrir l'hébergement de projets avec Git, le site offre de nombreuses fonctionnalités habituellement retrouvées sur les réseaux sociaux comme les flux, la possibilité de suivre des personnes ou des projets ainsi que des graphes de réseaux pour les dépôts. Github offre aussi la possibilité de créer un wiki et une page web pour chaque dépôt. Le site offre aussi un logiciel de suivi de problèmes. Alors que le système traditionnel open source amène chaque contributeur à télécharger les sources du projet et à proposer ensuite ses modifications à l'équipe du projet, Github repose sur le principe du fork par défaut : toute personne « forkant » le projet devient publiquement de facto le leader de son projet portant le même nom que l'original. Avec 16,7 millions de dépôts en 2014, GitHub est le plus grand hébergeur de code du monde.

5. Stack Overflow est un site web proposant des questions et réponses sur un large choix de thèmes concernant la programmation informatique.

6. Reddit est un site web communautaire de partage de signets permettant aux utilisateurs de soumettre leurs liens et de voter pour les liens proposés par les autres utilisateurs. Ainsi, les liens les plus appréciés du moment se trouvent affichés en page d'accueil. Hacker News est centré autour de l'actualité des technologies informatiques, du hacking et des startups, et promeut tout contenu susceptible de « gratifier la curiosité intellectuelle » des lecteurs.

## 2 Introduction aux images TRIMAGO et JPEG

Vous venez de télécharger le dossier "Images" contenant deux types d'images numériques

1. les images dont les noms sont suffixés par .tri sont des images au format TRIMAGO (un format "propriétaire"). Ces images sont non comprimées.
2. les images dont les noms sont suffixés par .jpg sont des images au format JPEG. Elles sont donc comprimées.

Pour s'en rendre compte, comparez les volumes des deux images couleur singe.tri et singe.jpg qui représentent la même scène, à savoir le visage d'un mandrill. Leur nombre de pixels est le même : 512 x 512 = 262.144. Mais :

1. singe.tri occupe 786,9 ko soit 3 octets = 24 bits par pixel [7]
2. singe.jpg occupe 78,6 ko soit 2,4 bit par pixel

**L'image singe.tri est donc 10 fois plus volumineuse que l'image singe.jpg** . Vous vérifierez néanmoins qu'elles paraissent visuellement identiques sur l'écran de votre PC. Cet exemple montre donc la qualité de la compression JPEG.

## 3 Lecture et affichage des images au format TRIMAGO

Ces images se composent d'un descripteur suivi de l'image proprement dite rangée ligne par ligne et canal par canal dans l'ordre R , V , B s'il s'agit d'une image couleur. Chaque mesure d'éclairement d'un pixel est quantifié sur un octet (entre 0 et 255)
La taille du descripteur est celle d'une ligne de l'image. Sa composition est :
*descripteur = (typeimage, etiquette,nlig,ncol,ncan,zonelibre)*
où

1. *typeimage* code le type de l'image sur 4 caractères
   (a) IMON s'il s'agit d'une image monochrome (noir et blanc, à niveaux de gris)
   (b) ITRI s'il s'agit d'une image trichrome (couleur) codée en R (rouge) , G (vert) et B (bleu)
2. *etiquette* contient un commentaire sur l'image de 64 caractères
3. *nlig* contient le nombre de lignes de l'image, codé en "integer big endian" [8] sur 32 bits.
4. *ncol* contient le nombre de colonnes de l'image, codé en "integer big endian" sur 32 bits.
5. *ncan* contient le nombre de canaux de l'image, codé en "integer big endian" sur 32 bits.
6. *ncan* zone libre est une zone libre de ncol - 80 caractères.

Vous allez commencer par lire une image au format TRIMAGO (dans le dossier "Images") puis à l'afficher avec son commentaire sur votre écran de machine. L'image lue sera rangée dans le tableau im. Les '??' indiquent les champs qui doivent être complétés.
**Solution : script lecshowtrimago.m**

```
%
NOMFIC = uigetfile('Images/*.tri');
%
NOMFIC = ['Images/' NOMFIC];
imtri = fopen(NOMFIC,'r','??');
%
typeimage = fread(imtri,4,'??');
typeimage = typeimage';
etiquette = fread (imtri,64,'??');
etiquette = etiquette';
```

7. la dynamique du capteur étant de 256 (un octet) sur les trois primaires R, V ,B et l'image n'étant pas comprimée, il s'ensuit que son volume est (approximativement) égal à son nombre total de pixels fois 24 bits, soit 24 bits par pixel

8. Le codage "big endian" (en français, mot de poids fort en tête) veut dire que l'octet de poids le plus fort est enregistré à l'adresse mémoire la plus petite, l'octet de poids inférieur à l'adresse mémoire suivante et ainsi de suite. Dans le codage "little endian"(en français, mot de poids faible en tête), c'est l'inverse, l'octet de poids le plus faible est enregistré à l'adresse mémoire la plus petite, etc. Il faut savoir que les processeurs x86 qui se trouvent dans les PC ont une architecture "little endian"

```
nlig = fread(imtri,1,'??');
ncol = fread(imtri,1,'??');
ncan = fread(imtri,1,'??');
fread (imtri,ncol-80);
%
if strcmp(typeimage,'ITRI')
   im = fread(imtri)[9];
   im = reshape(im,'??','??','??');
   im = permute(im,'??');
   him[10] = figure ('Name',etiquette,'Units','??');
   imshow(im/255);
else if strcmp(typeimage,'IMON')
      im = fread (imtri,['??','??']);
      im = im';
      him = figure ('Name',etiquette,'Colormap',gray(256),'Units','??');
      imshow(im,[0,255]);
   else
      disp('ERREUR');
   end
end
fclose(imtri);
```

# 4 Lecture et affichage des images au format JPEG

**Solution : script lecshowjpg.m**

```
% Lecture d'une image JPEG
%
NOMFIC = uigetfile('Images/*.jpg');
%
NOMFIC = ['Images/' NOMFIC];
im = imread[11] (NOMFIC);
pas = str2double(input('Pas d"echantillonnage : ','s')); % pas d'échantillonnage de im
im = double(im(1 :pas :end,1 :pas :end,1 :end)); %sous-échantillonnage de l'image et transformation en double
%
nlig = size(im,1);
ncol = size(im,2);
%
% Affichage de l'image
him = figure ('Name','Image JPEG','Units','pixels');
if ismatrix(im)
   % L'image JPEG est monochrome
   typeimage = 'IMON';
   ncan = 1;
   colormap(gray(256));
   imshow(im,[0,255]);
else if ndims(im) == 3
      % L'image JPEG est en couleur
      typeimage = 'ITRI';
      ncan = 3;
```

9. par defaut, fread lit un fichier octet par octet, l'interprete comme un entier non signe (uint8) et retourne un vecteur de type double
10. him est le "handle" , l'identifiant de la figure ; en tapant figure(him), la figure identifiee par him devient visible
11. la fonction *imread* réalise la décompression totale de l'image JPEG alors que la fonction *jpeg_read* (cf.12.2) ne réalise qu'un décodage partiel.

```
        imshow(im/255);
    else
        disp('ERREUR');
    end
end
```

# 5 Compression des images au format TRIMAGO selon la norme JPEG

On désire maintenant comprimer **toutes** les images .tri au format .jpg. Une fois cette conversion effectuée, un calculera pour chaque image .tri le taux de compression obtenu.

**Solution : script tri2jpg.m**

```
% tri2jpg
%
lecshowtrimago;
NOMFIC = [NOMFIC(1 :find(NOMFIC =='.')),'jpg'];
imwrite(im/255,NOMFIC,'jpg','Comment',etiquette);
disp(imfinfo(NOMFIC));
```

# 6 Affichage des profils ligne

Le profil ligne d'une image monochrome est le graphe de la fonction qui à tout pixel d'une ligne fait correspondre la valeur quantifiée d'éclairement de ce pixel. Le profil ligne d'une image trichrome RGB est composé des trois graphes des fonctions qui à tout pixel d'une ligne font correspondre les valeurs quantifiées d'éclairement de ce pixel dans les trois canaux primaires R, G et B. [12] L'utilisateur désigne une ligne de l'image avec la souris et le programme affiche son profil.

**Solution : script profim.m** [13]

```
%% Trace le profil horizontal de la ligne de l'image (monochrome ou couleur) designee par le curseur.
disp('Designer une ligne dont on veut le profil avec le curseur');
[ ,x] = ginput (1);
tempy = x*ones(ncol,1)';
tempv = 1 :ncol;
%On efface toute trace de marque anterieure.
eval('delete(hl)',");
%
if all(typeimage == 'IMON')
    hl = line(tempv,tempy,'Color','g');
    %On efface toute trace de marque anterieure.
    eval('delete(hp)',");
    hp = line(tempv,nlig*(1 - im(round(x), :)/256),'Color','r');
else
    if all(typeimage == 'ITRI')
        hl = line(tempv,tempy,'Color','w');
```

12. Ceci n'est vrai que pour les capteurs Foveon X3 des reflex numériques SIGMA ; ceux-ci enregistrent bien , pour chaque pixel, les trois couleurs primaires R, G et B. Les autres capteurs utilisent un filtre qui ne permet à chaque pixel de ne voir qu'une seule couleur : soit R, soit G, soit B. L'image capturée par les reflex SIGMA est donc une image vue intégralement en couleurs par le capteur, alors que les autres appareils photo numériques restituent une image calculée, reconstituée à partir d'informations partielles correspondant au tiers de l'information totale de l'image réelle. Il en résulte une qualité d'image inconnue jusqu'ici en photo numérique, et un rendu des détails plus riche que celui d'un capteur conventionnel qui comporterait au moins le double de pixels "classiques" obtenus par interpolation. En effet, un capteur conventionnel de 10 millions de pixels n'enregistre que le tiers de l'image finale, avec 2,5 million d'informations en bleu, 5 millions en vert et 2,5 million en rouge et calcule donc les 2/3 de l'image qui lui manquent lors de la capture. L'image issue des reflex numériques SIGMA est plus précise, avec en particulier plus de détails dans les couleurs, plus naturelle, et sans artefacts tels que le moirage.

13. le scipt profim permet de révéler l'image numérique sous-jacente à l'image affichée à l'écran

```
        %On efface toute trace de marque anterieure.
        eval('delete(hpr)',") ;
        eval('delete(hpg)',") ;
        eval('delete(hpb)',") ;
        hpr = line(tempv,nlig*(1 - im(round(x), :,1)/256),'Color','r') ;
        hpg = line(tempv,nlig*(1 - im(round(x), :,2)/256),'Color','g') ;
        hpb = line(tempv,nlig*(1 - im(round(x), :,3)/256),'Color','b') ;
    else
        disp('ERREUR') ;
    end
end
```

# 7 Etude de la transformée de Fourier

## 7.1 Normalisation de l'image

Normaliser l'image im (notée **imn**) de telle sorte que la moyenne des intensités soit nulle et que son énergie soit égale à l'unité.
**Solution : normalise.m**

## 7.2 Calcul de la transformée de Fourier normalisée de im

Si
imf = fft2(im) ;
alors

$$imf(u+1,v+1) = \sum_{k=0}^{nlig-1} \sum_{l=0}^{ncol-1} im(k+1,l+1) W_l^{-uj} W_c^{-vk} \quad u = 0 \dots nlig-1 \quad v = 0 \dots ncol-1$$

$W_l = e^{\frac{2i\pi}{nlig}}$ $W_c = e^{\frac{2i\pi}{ncol}}$ [14]

Calculer la transformée de Fourier normalisée de l'image *normalisée* imn (notée imfn) [15] ; imfn est donc une **représentation orthogonale** de imn
**Solution : fft2n.m**

Que doit valoir imfn(1,1) ?

Vérifier la conservation de l'énergie par fft2n :

**Solution : conservation.m**

## 7.3 Vérification de la relation de conjugaison

La relation à vérifier est donc :

$$IM(1-f_x, 1-f_y) = IM^*(f_x, f_y) \quad \forall f_x, f_y$$

où IM est la matrice de Fourier indicée par les fréquences spatiales en x (lignes) et en y (colonnes).
Pour vérifier la relation de conjugaison, vous établirez les relations existant entre $f_x$ et $u$ d'une part, $f_y$ et $v$ d'autre part, puis vous exprimerez la relation de conjugaison sur imf (ou sur imfn).

14. on remarquera que la fonction de MATLAB fft2 n'est pas normalisée : il n'y a pas conservation de l'énergie
15. la transformée de Fourier normalisée de im décompose l'image im sur les images de base des "exponentielles complexes" lesquelles forment une base **orthonormée** de $C^{nlig*ncol}$

**Solution : vconjg.m**

```
% Verification de la relation de conjugaison
%
if all(abs(imfn(1,2 :1+ncol/2)-conj(imfn(1,ncol :-1 :1+ncol/2))) == zeros(1,ncol/2))
   disp('Relation de conjugaison verifiee sur (1,2 :1+ncol/2)') [16]
else
   disp('ATTENTION Relation de conjugaison NON verifiee sur (1,2 :1+ncol/2)')
end
%
if '??'
   disp('Relation de conjugaison verifiee sur (2 :1+nlig/2,1)')
else
   disp('ATTENTION Relation de conjugaison NON verifiee sur (2 :1+nlig/2,1)')
end
%
if '??'
   disp('Relation de conjugaison verifiee sur (2 :1+nlig/2,2 :1+ncol/2)')
else
   disp('ATTENTION Relation de conjugaison NON verifiee sur (2 :1+nlig/2,2 :1+ncol/2)')
end
```

### 7.4 Affichage du module de la transformée de Fourier de l'image normalisée imn

imfa = abs(imfn) ;

**Solution : afficher_imsc0.m**

### 7.5 Déplacer l'origine de la transformée de Fourier au centre de l'image

Revenons à l'image initiale im et à sa transformée de Fourier non normalisée imf.
Pour centrer cette transformée de Fourier (le résultat est **imfc**) , on utilise la fonction fftshift. Afficher le module de imfc.
Calculer ensuite la transformée de Fourier inverse de imfc et vérifier que l'on obtient une nouvelle image $\tilde{im}$.

$$\tilde{im}(x,y) = (-1)^{x+y} im(x,y)$$

Démontrer cette dernière relation.

**Solution : centrefourier.m**

## 8 Compression de l'image par transformée de Fourier

On testera deux méthodes :

1. Annulation des coefficients de hautes fréquences
2. Annulation des coefficients de plus faibles poids

### 8.1 Annulation des coefficients de hautes fréquences

Pour faire cela, on reprend l'image centrée **imfc** et on la multiplie par un masque *mask* centré de taille (*hauteur,largeur*). Les coefficients du masque valent 1 à l'intérieur du masque et 0 à l'extérieur.

16. Pour afficher du texte en couleur dans la fenêtre de commande (Command Window) on utilisera la fonction **cprintf** fournie (et non pas disp)

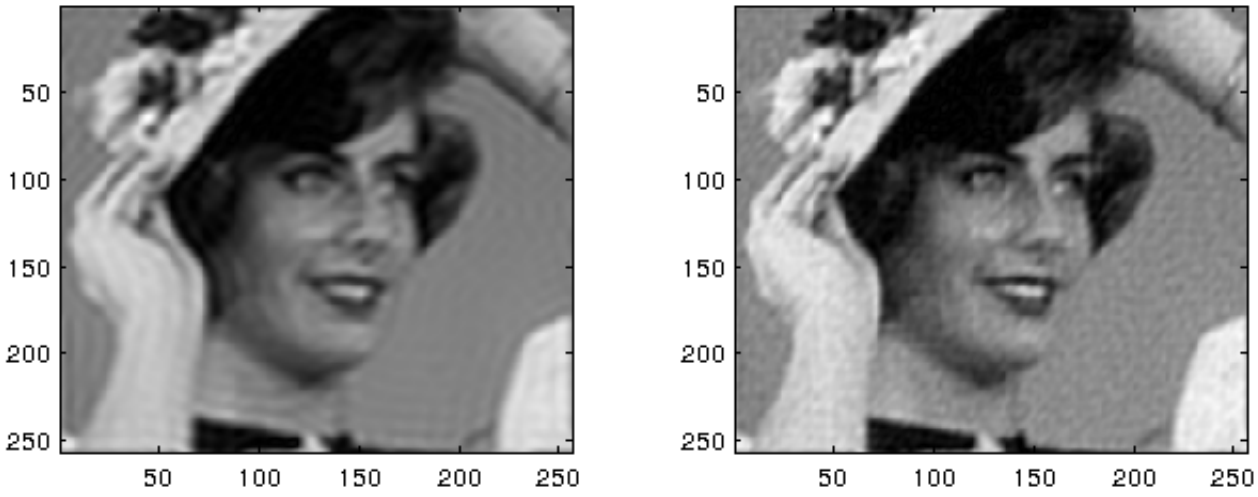


FIGURE 1 – Comparaison des deux méthodes de compression

1. Construire le masque[17] *mask*
2. Comprimer l'image et l'afficher (début de solution : imfc0 = mask.*imfc ;)
3. Faire varier la taille du masque

**Solution : comp0.m**

## 8.2 Annulation des coefficients de plus faibles poids

Etant donnée une erreur quadratique eps, on cherche le plus petit nombre de coefficients de Fourier non nuls qui fournissent une erreur inférieure à eps. L'idée est de mettre à zéro les coefficients de Fourier qui sont les plus faibles. Il est donc nécessaire de les trier.

**Solution : comp_opt.m**

## 8.3 Comparaison des deux méthodes de compression

Il s'agit de savoir quelle est la meilleure méthode de compression. Pour cela on doit comparer les deux images comprimées imc0 et imc1 lorsqu'on met à 0 le même nombre de coefficients de fourier.

**Solution : comparaisoncompressions.m**

```
% Comparaisons compressions 0 et opt
%
%% Compression par annulation des coefficients de Fourier de plus faibles poids (compression opt)
%
imfc1 = imf;
energie = norm(imf,'fro')^2; % somme des modules au carre des coefficients de Fourier non normalises
pourcentage = str2double(input('pourcentage de conservation de l'énergie de l'image :,'s'));
seuilerreur = (1 - pourcentage/100)*energie;
%
[imftri,index] = sort(abs(imf( :))); % on trie les modules des coefficients par ordre croissant
%
% attention sort(imf( :)) est erronné car on trie alors selon la partie
% réelle (cf. doc)
```

17. On remarquera que la matrice *mask* est de rang 1 ; donc $mask = c' * l$ où c est un vecteur ligne et l un vecteur colonne ; que valent c et l ?

```
%
erreur2 = 0;
i = 1;
erreur2 = erreur2 + abs(imftri(i))^2;
while erreur2 <= seuilerreur
    k=rem(index(i)-1,nlig)+1;
    l=floor((index(i)-1)/ncol)+1;
    imfc1(k,l)=0;
    i=i+1;
    erreur2=erreur2 + abs(imftri(i))^2;
end
nombrea0=i-1; % nombre de coefficients mis a 0
cprintf('blue',['Nombre de coefficients mis à 0 : ',num2str(nombrea0)]);
disp(' ');
pourcent0 = 100*nombrea0/(nlig*ncol);
cprintf('blue',['Pourcentage de coefficients mis à 0 : ',num2str(pourcent0)]);
disp(' ');
%
imc1 = ifft2(imfc1);
imc1 = abs(imc1); % imc1 est l'image comprimée par annulation des coefficients de plus faibles poids
%
%% Compression par annulation des coefficients de hautes fréquences spatiales (compression 0)
% Construction du masque
hauteur = round(sqrt(nlig*ncol - nombrea0));
largeur = hauteur;
c = '??'
l = '??'
mask = c'*l;
% Visualisation du masque
xori = (9*ncol)/4;
yori = 0;
hmask = figure ('BackingStore'[18], 'on' ,'Color','k','Colormap',gray(256),...
'Name','Masque','Units','pixels',...
'Position', [xori , yori , ncol , nlig ]);
imagesc(mask);
%
% Filtrage par le masque mask : les coefficients en dehors du masque sont
% mis a 0.
%
imfc0=imfc.*mask; % imfc est la transformee de Fourier centree
%
imc0 = ifft2(imfc0);
imc0 = abs(imc0); % imc0 est l'image comprimee par annulation des coefficients de hautes frequences spatiales

%
%% Affichage des deux images comprimees au meme taux (meme nombre de coéfficients de Fourier mis a 0)
xori = 0;
yori=0;
h2 = figure ('BackingStore','on' ,'Color','k','Colormap',gray(256),...
'Name',['imc0 et imc1 nombre de 0 : ',num2str(nombrea0)],'Units','pixels',...
'Position', [xori , yori , round(2.8*ncol) , nlig ]);
```

18. Le mot "BackingStore" désigne un buffer utilisé par MATLAB pour stocker une copie du contenu de la fenêtre d'une figure. Par défaut, Backingstore est on ; lorsque la mémoire de votre système est limitée, il faut établir BackingStore à off pour récupérer la mémoire utilisée par ce buffer. De même, lorsqu'on cherche à rafraîchir l'écran rapidement, on doit établir BackingStore à off pour éviter de réécrire à la fois dans le buffer et dans la fenêtre de la figure

```
subplot(1,2,1) ;
imagesc(imc0) ;
subplot(1,2,2) ;
imagesc(imc1) ;
```

# 9 Calculer et afficher quelques images de base réelles pour différents couples fréquentiels

On supposera pour simplifier que le nombre de lignes de l'image est égal au nombre de colonnes : nlig = ncol = n. Les images de base de la représentation de Fourier sont réelles lorsque l'image est réelle. Elles sont formées des images des fonctions sinus et cosinus [19] paramétrées par les fréquences spatiales :

$f_x = 0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{1}{2}$

$f_y = 0, \frac{1}{n}, \frac{2}{n}, \ldots, 1 - \frac{1}{n}$

ce qui fait bien 2 (sinus + cosinus) $* \frac{n}{2}(f_x) * n(f_y) = n^2$ images de base pour une image (n,n).
Pour calculer ces images de base, le plus simple est de commencer par calculer leurs transformées de Fourier.

**Solution : basereellefourier.m**

```
% Calcul et affichage d'une image de base reelle de la transformee de
% Fourier d'une image reelle
% on supose que nlig == ncol!!
%
rep='t';
n = nlig;
while rep  = 's' & rep  = 'c'
   rep = input ('Desirez-vous afficher un sinus ou un cosinus (s/c) ?', 's');
end
u = str2double(input ('n * frequence en x ? ','s'));
v = str2double(input ('n * frequence en y ? ','s'));
%
% mise a zero de imf
basef = zeros(n,n);
%
if rep == 'c'
   if u  = 0 && v  = 0
      basef(u+1,v+1)= '??';
      basef(n+1-u,n+1-v)= '??';
   elseif u == 0
      basef(1,v+1)= '??';
      basef(1, n+1-v)= '??';
   else
      basef(u+1,1)= '??';
      basef(n+1-u,1)= '??';
   end
else
   %rep == 's'
```

19. on peut donc dire que la transformée de Fourier décompose une image réelle en ses composantes en sinus et en cosinus ou mieux encore en une somme de composantes en cosinus déphasées les unes par rapport aux autres. On a en effet :

$$\frac{n}{2} im(x,y) = \frac{1}{2} IM(0,0) + \sum_{\substack{f_x, f_y = 0 \\ (f_x,f_y) \neq (0,0)}}^{\frac{1}{2}, 1-\frac{1}{n}} |IM(f_x,f_y)| \cos(2\pi(f_x x + f_y y) + \phi(f_x,fy)) \qquad \text{où} \quad IM(f_x,f_y) = |IM(f_x,f_y)| e^{i\phi(f_x,fy)}$$

Par leurs phases $\phi(f_x, fy)$ ces images de base dépendent de l'image im elle-même (ce qui ne sera pas le cas pour la représentation en cosinus)

```
    if u  = 0 && v  = 0
        basef(u+1,v+1)= '??';
        basef(n+1-u,n+1-v)= '??';
    elseif u == 0
        basef(1,v+1)= '??';
        basef(1, n+1-v)= '??';
    else
        basef(u+1,1)= '??';
        basef(n+1-u,1)= '??';
    end
end
%
base=ifft2(basef)*n;
base = 10000 * real(base);
base = base -min(base( :));
%
map = exp(1.*gray(63))-1;
map = map/max(max(map));
hbase = figure('BackingStore','on','Colormap', map, 'Units', 'pixels');
imagesc(base);
axis('square');
% Tracer un profil ligne de base
profimfunct(base);
```

## 10 Affichage de résultats

**Solution : affichage1.m**

```
%% Affichage des principaux résultats de l'étude de la transformée de Fourier
%
xori = 0;
yori= 0;
h1 = figure ('BackingStore','on' ,'Color','k', 'Colormap',gray(256),...
'Name','Resultats sur l'etude de la Transformee de Fourier','Units','pixels',...
'Position', [xori , yori , 2*ncol , 2*nlig ]);
%
a(1) = subplot(3,3,1);
image(ind2rgb(im,jet(256)))20;
%
a(2) = subplot(3,3,2);
imfa = abs(imf);
ecart_type=std(std(imfa));
imfa = round((imfa/ecart_type)*256);
image(imfa);
%
a(3) = subplot(3,3,3);
imfa = abs(imfc);
ecart_type=std(std(imfa));
imfa = round((imfa/ecart_type)*256);
image(imfa);
%
a(4) = subplot(3,3,4);
imagesc(mask);
%
```

```
a(5) = subplot(3,3,5);
image(ind2rgb(round(imc0),jet(256)))[20];
%
a(6)= subplot(3,3,6);
image(ind2rgb(round(imc1),jet(256))) 20;
%
sincos3; % calcule sin30 , cos30 et cosphi30[21]
%
a(7) = subplot(3,3,7);
imagesc(sin30);
%
a(8) = subplot(3,3,8);
imagesc(cos30);
%
a(9) = subplot(3,3,9);
imagesc(cosphi30);
%
ht1 = title(a(1),'image initiale');
set(ht1,'Color','white');
ht2 = title(a(2),'TF');
set(ht2,'Color','white');
ht3 = title(a(3),'TF centrée');
set(ht3,'Color','white');
ht4 = title(a(4),'masque');
set(ht4,'Color','white');
ht5 = title(a(5),'compression par masque');
set(ht5,'Color','white');
ht6 = title(a(6),'compression optimale');
set(ht6,'Color','white');
ht7 = title(a(7),'sin(3,0)');
set(ht7,'Color','white');
ht8 = title(a(8),'cos(3,0)');
set(ht8,'Color','white');
ht9 = title(a(9),'cosinus phase (3,0)');
set(ht9,'Color','white');
%
```

20. Afin de mieux comparer visuellement les résultats des deux compressions, on affiche les images monochromes en pseudo-couleurs plutôt qu'en niveaux de gris ; pour cela, on utilise la table de couleur jet(256) et on convertit l'image indexée im en une image non-indexée RGB au moyen de la fonction ind2rgb. L'affichage se fait alors en RGB et tout se passe comme si on affichait alors une image indexée par la table jet(256) alors qu'elle était préalablement indexée par la table de la figure où elle s'inscrivait à savoir gray(256)

21. sin30 ,cos30 et cosphi30 sont respectivement les images de base en sinus, cosinus et cosinus phasés (cf note 19) calculées aux fréquences $f_x = \frac{3}{n}$ et $f_y = 0$

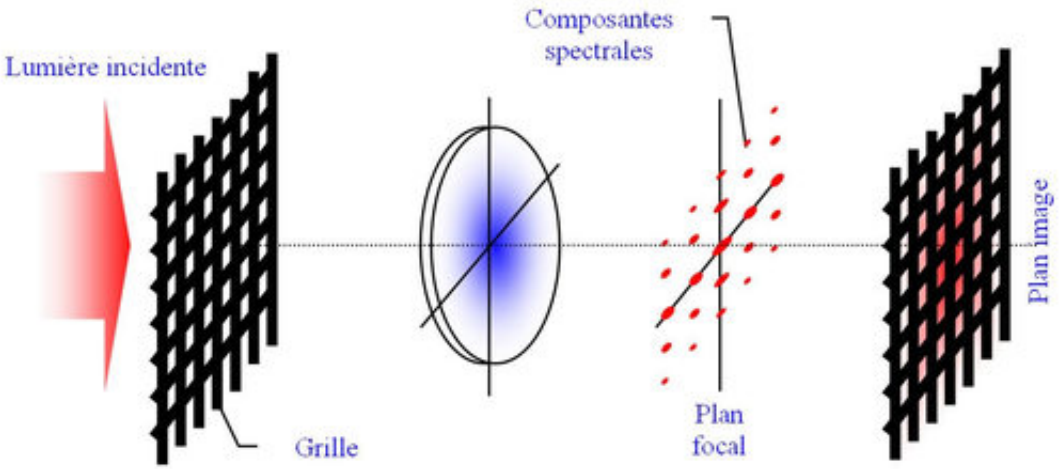


FIGURE 2 – L'expérience d'Abbe et Porter (issu du site *www.optique-ingenieur.org*)

## 11 Simulation numérique des expériences d'Abbe et Porter

La transformée de Fourier (continue) joue un grand rôle en optique[22].Ceci a été prouvé de façon éclatante par les expériences réalisées par Abbe en 1893 et plus tard par Porter en 1906. Nous nous proposons à présent de réaliser une simulation numérique de ces expériences. Celles-ci consistent à éclairer à l'aide d'une source de lumière cohérente une grille métallique. Dans le plan focal d'une lentille apparaît alors la transformée de Fourier de cette grille et finalement dans le plan image, les différentes composantes de Fourier transmises par la lentille se recombinent pour former une réplique de cette grille. En plaçant divers obstacles (par exemple un diaphragme à iris ou une une fente) dans le plan focal, on modifie la transformée de Fourier (et donc l'image) de la grille de différentes façons.

La similation numérique de cette expérience va consister à programmer les étapes suivantes :

1. Lecture du fichier grillec.tri
2. Calcul de la transformée de Fourier centrée
3. Construction d'une fente horizontale de hauteur 10 pixels et de largeur 250 pixels (par exemple)
4. Filtrage par la fente horizontale : les coefficients de Fourier en dehors de la fente sont mis à 0
5. Calcul de l'image filtrée
6. Rotation de 90° de la fente pour la rendre verticale
7. Filtrage par la fente verticale
8. Calcul de l'image filtrée
9. Affichage des résultats de l'expérience dans une même figure avec de haut en bas et de gauche à droite :
   (a) Affichage de l'image de la grille
   (b) Affichage des intensités (amplitudes au carré) des coefficients de Fourier
   (c) Affichage de l'image filtrée par la fente horizontale
   (d) Affichage du spectre de la grille filtrée par la fente horizontale
   (e) Affichage de l'image filtrée par la fente verticale
   (f) Affichage du spectre de la grille filtrée par la fente verticale

**Solution : AbbePorter.m**

22. La prise de conscience de l'utilité de la transformée de Fourier dans l'analyse des systèmes optiques est due en grande partie à P.M. Duffieux dont les travaux de recherche aboutirent en 1946 à la publication de l'ouvrage : "L'intégrale de Fourier et ses applications à l'Optique", Faculté des Sciences, Besançon.

# 12 Etude de la compression JPEG

## 12.1 Diagrammes de compression et de décompression

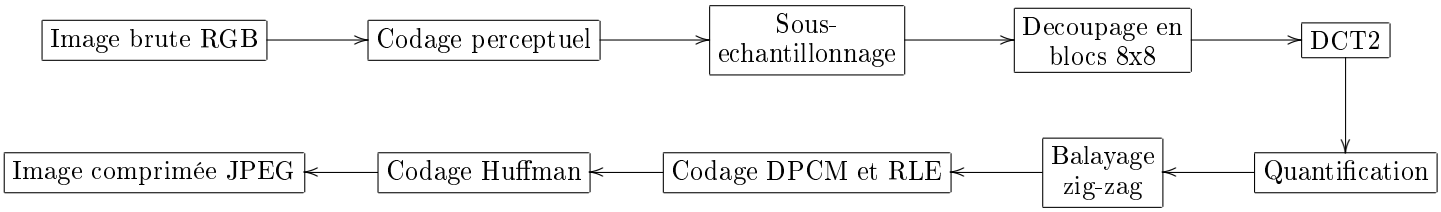


FIGURE 3 – Compression JPEG

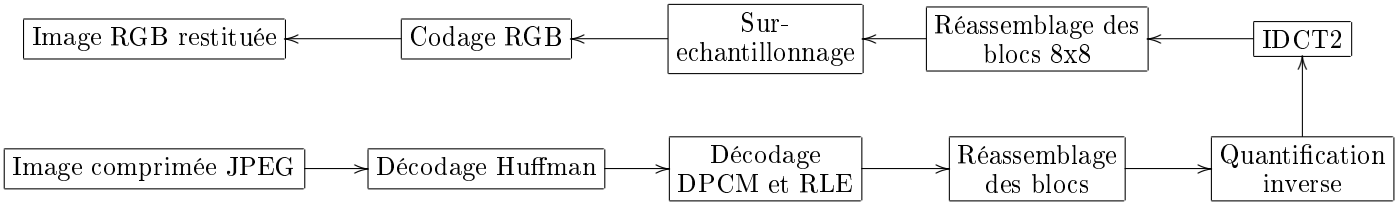


FIGURE 4 – Décompression JPEG

## 12.2 La fonction jpeg_read

Nous commençons par décoder partiellement le contenu d'une *image monochrome comprimée JPEG* (telle que Lena.jpg[23]) Pour ce faire nous allons utiliser le fichier MEX **jpeg_read.mexa64**[24]. Le fichier jpeg_read.mexa64 contient la fonction **jpeg_read** écrite en C, laquelle réalise les décodages Huffman[25] et Run-Length du train de bits et le réarrangement des coefficients DCT après quantification en une matrice de blocs 8x8.

```
jobj = jpeg_read('Images/Lena.jpg');
```

Le résultat se trouve ici dans la structure jobj qui contient 14 champs dont les deux plus importants sont :

1. coef_arrays : cellule contenant la matrice composée des coefficients DCT des blocs 8x8
2. quant_table : matrice 8x8 des coefficients utilisés pour la quantification

## 12.3 Décompression grossière de l'image JPEG

Notre premier objectif est de décomprimer l'image JPEG le plus simplement possible sans utiliser la transformée en cosinus inverse, ni la table de quantification. Pour ce faire nous allons seulement utiliser la composante continue de chaque bloc qui n'est autre que le coefficient DCT pour un couple fréquentiel nul (situé en haut et à gauche de chaque bloc)
**Solution : decode0**

## 12.4 Décompression de l'image JPEG sans utiliser la table de quantification

La décompression se fera en calculant les transformées en cosinus inverse de chaque bloc 8*8
**Solution : decode1**

23. Lena est un morceau de photo d'une playmate prise dans le numéro de novembre (miss novembre) 1972 du magazine Playboy. Elle sert d'image de test pour les algorithmes de traitement d'image et est devenue de facto un standard industriel et scientifique.
24. Les fichiers .mex sont des fichiers écrits en C, C++ ou en Fortran, qui une fois compilés, peuvent être utilisés dans MATLAB de la même manière que les fichiers .m A l'inverse, ils permettent aussi d'appeler directement des fonctions MATLAB dans du code C, C++ ou Fortran.
25. le code d'Huffman est un code entropique qui permet d'obtenir une longueur moyenne de codage de chaque symbole qui soit proche de l'entropie laquelle représente la longueur minimale de codage possible

### 12.5 Décompression de l'image JPEG en utilisant la table de quantification

Cette fois, on multiplie chaque coefficient DCT par son coefficient de quantification correspondant.
**Solution : decode2**

### 12.6 Comparaison des résultats

Comparer visuellement les trois images décomprimées et les classer par ordre de qualité visuelle décroissante. Pour cela, on affichera les trois images côte à côte dans une même figure.
**Solution : affichage2.m**

## 13 Récapitulatif des principaux identificateurs des variables du Workspace

- im : image initiale
- imn : image normalisée (moyenne des intensités nulle et énergie égale à 1)
- imf : transformée de Fourier, non normalisée, de im
- imfn : transformée de Fourier, normalisée de imn
- imfc : transformée de Fourier centrée et non normalisée de im
- imfa : module de imfn ou de imfc
- imtilde : transformée de Fourier inverse (non normalisée) de imfc
- imc0 : image Fourier-comprimée par annulation des coefficients de hautes fréquences spatiales
- imc1 : image Fourier-comprimée par annulation des coefficients de plus faibles poids
- base : image de base réelle (sinus ou cosinus) de la représentation de Fourier d'une image réelle
- jobj : structure d'une image JPEG
- im0 : image JPEG-décodée par decode0
- im1 : image JPEG-décodée par decode1
- im2 : image JPEG-décodée par decode2

## 14 Travail demandé

On fournira une archive .tar contenant

1. Un rapport au format pdf avec
   (a) tous les scripts MATLAB mentionnés dans ce texte. Pour l'édtion de ces scripts, on utilisera l'outil "Publish" de Matlab
   (b) répondant aussi aux questions posées dans le texte
2. toutes les sources de ces scripts

Cette archive sera transmise à l'enseignant responsable de votre groupe de TP avant le **vendredi 15 mai 2015 18h**.