BIMA : **B**ases de l'**IMA**gerie

Énoncés de TD/TME
Séances 1-5

Master 1 Informatique - Spécialité **IMA**gerie
Université **P**ierre et **M**arie **C**urie
Année Universitaire 2016-2017

Table des matières

Semaine 1

Traitements globaux d'images

1.1 TD 1 : manipulations de base

Exercice 1 — Quelques rappels

1. Rappeler les deux manières de représenter une image en informatique. En supposant que le premier pixel est indicé par 0, comment, selon ces deux types de représentation, peut-on accéder au pixel de coordonnées (i, j) d'une image de taille $N \times M$? De même, comment accéder aux voisins N, S, O et E de ce pixel (i, j) ?
2. Quelle est la taille en bits d'une image de taille 256×256 codée sur un octet (on négligera la taille de l'en-tête) ? Quelle est la dynamique de ce type d'image ?

Exercice 2 — Opérations matricielles sur les images

On considère dans cet exercice les deux images suivantes (pixels compris entre 0 et 255) :

$I=$

100	100	100	100	100	100
100	200	200	100	100	100
100	200	200	100	100	100
100	100	100	100	100	100
100	100	100	100	100	100

$J=$

100	100	100	100	100	100
100	100	200	200	100	100
100	100	200	200	100	100
100	100	100	100	100	100
100	100	100	100	100	100

1. Donner l'image résultat de la somme de I et J. Que constate-t-on ?
2. Faire la soustraction $|I - J|$ de ces deux images (différence absolue). Que remarque-t-on ?
3. Comment peut-on faire pour obtenir, à partir de I, une image de la même taille que I, ne contenant qu'une de ses sous-parties entourée de noir ?

Exercice 3 — Histogramme : calcul et manipulation

Dans cet exercice, on considère l'image suivante :

$I=$

10	2	1	1	10	2	6
7	5	5	4	4	2	3
7	2	0	3	2	1	8
6	3	2	0	3	0	3
5	3	2	7	0	3	5

1. Quel est le niveau de gris minimal $k_{\min}$? le niveau de gris maximal $k_{\max}$? En déduire la dynamique L spécifique à cette image.
2. Rappeler ce qu'est un histogramme et donner un algorithme pour le calcul d'un histogramme d'une image de taille $N \times M$ stockée dans un vecteur f. Donner le code `Matlab` correspondant.
3. Dessiner l'histogramme de I.
4. Proposer un algorithme à appliquer sur l'histogramme pour inverser les niveaux de gris de l'image. Peut-on retrouver l'image inversée à partir de ce nouvel histogramme ? Justifier.
5. Rappeler comment on calcule et dessine l'histogramme cumulé.
6. Proposer un algorithme à appliquer sur l'image pour la binariser à 2 valeurs k_1 et k_2, en fonction d'un seuil S ($I_b = k_1$ si $I \leq S$ et $I_b = k_2$ sinon). Tracer l'histogramme de l'image binarisée.
7. On souhaite changer l'intervalle de variation des pixels de cette image de $[k_{\min}, k_{\max}]$ vers $[I_1, I_2]$. En supposant qu'on effectue une normalisation `linéaire`, écrire la transformation subie pour chaque pixel. Donner le code `Matlab` correspondant.
8. Déterminer l'histogramme étiré vers l'intervalle $[2, 30]$, l'histogramme rétréci vers l'intervalle $[2, 8]$
9. L'égalisation d'histogramme consiste à appliquer une transformation à l'image de sorte que son histogramme soit plat. Dans le cas continu, on a donc :

$$\int_{k_{min}}^{k_{max}} p(x)dx = \int_{k_{min}}^{k_{max}} p'(x)dx \tag{1.1}$$

avec $p'(x) = \frac{1}{k_{max}-k_{min}}$.
Dans le cas de l'image I précédente, montrer que l'application de l'équation 1.1 au cas discret consiste à transformer un niveau de gris k en $k' = \text{Int}\left(\frac{k_{max}}{N\times M} H_c(k)\right)$, avec :
— N le nombre de lignes de l'image ;
— M le nombre de colonnes de l'image ;
— $H_c(k)$ l'histogramme cumulé du niveau de gris k ;
— `Int` la fonction qui arrondit à l'entier le plus proche.

1.2 TME 1 : améliorations globales d'image

1.2.1 Préliminaires : premiers pas avec `Matlab`

Avant toute chose, on pensera à utiliser la fonction d'aide de `Matlab` : `help fonction`. Ce tutoriel est un très bref résumé des fonctionnalités de ce logiciel. On en aura besoin de bien d'autres.

`Matlab` traite un seul 1 type d'objet : les matrices. Les scalaires sont des matrices (1×1), les vecteurs lignes des matrices $(1 \times n)$, les vecteurs colonnes des matrices $(n \times 1)$. Le type des données est, par défaut, en `double`.

1.2.1.1 Les constantes prédéfinies

Il existe deux constantes prédéfinies particulières : `Nan` ou `Not a Number`, pour exprimer une indétermination et `Inf` pour un nombre infini. Il existe aussi les deux constantes suivantes :

```
>> pi
ans =
    3.1416

>> eps
ans =
   2.2204e-16
```

1.2.1.2 Déclaration d'une matrice

Il existe plusieurs manières de déclarer une matrice :

1. Déclaration par listing de coefficients :

   ```
   >> I=[1 2 3; 4 5 6;7 8 9]
   I =
        1     2     3
        4     5     6
        7     8     9
   ```

   On déclare une matrice entre crochets (pour un vecteur, on peut les omettre), les éléments d'une même ligne sont séparés par des espaces, et les lignes sont séparées entre elles par des points virgules.

2. Allocation d'une matrice dont les coefficients seront remplis plus tard (i.e. matrice nulle) :

   ```
   >> I =zeros(3,4)
   I =
        0     0     0     0
        0     0     0     0
        0     0     0     0
   ```

   Cette fonction permet d'allouer une matrice de la taille et la dimension désirée (ici 3 lignes et 4 colonnes). De même, il existe la fonction `ones` qui a les mêmes propriétés et qui construit une matrice de 1. Le nombre minimum de paramètres est de 2. Ainsi, pour construire des vecteurs (matrice de dimension 1) il faut mettre un paramètre à 1 (on le choisit suivant que l'on souhaite un vecteur colonne ou un vecteur ligne.).

```
>> v =zeros(1,4)
v =
     0     0     0     0
```

3. Suite de nombres

```
>> v1 =1:10
v1 =
     1     2     3     4     5     6     7     8     9    10

>> v2=10:-2:1
v2 =
    10     8     6     4     2
```

La syntaxe `deb:fin` permet de créer une suite d'entiers de `deb` à `fin` avec un pas par défaut de 1. Si on veut modifier le pas, on utilise la syntaxe `deb:pas:fin` qui propose une suite de `deb` à `fin` avec un pas de `pas` (le pas peut être quelconque).

1.2.1.3 Taille d'une matrice : `size`

La taille d'une matrice est donnée par la fonction `size(I)` qui renvoie deux scalaires, dans l'ordre le nombre de lignes puis le nombre de colonnes (pour le cas d'une matrice $2D$). Pour récupérer uniquement le nombre de lignes (resp. le nombre de colonnes), on peut spécifier `size(I,1)` (resp. `size(I,2)`).

```
>> I=ones(2,5)
I =
     1     1     1     1     1
     1     1     1     1     1

>> size(I)
ans =
     2     5

>> size(I,1)
ans =
     2

>> size(I,2)
ans =
     5
```

1.2.1.4 Éléments de la matrice

Pour extraire un élément, il faut donner un indice pour la ligne puis un indice pour la colonne de la matrice (qui doivent être valides). Attention, dans `Matlab` les indices d'une matrice commencent par 1.

```
>> I=[1 2 3 4 5;6 7 8 9 10]
I =
```

```
     1     2     3     4     5
     6     7     8     9    10

>> I(2,4)
ans =
     9

>> I(3,2)
??? Attempted to access I(3,2); index out of bounds because size(I)=[2,5].
```

On peut extraire toute une ligne ou toute une colonne d'une matrice, pour cela, on utilise l'opérateur ':'.

```
>> I(1,:)
ans =
     1     2     3     4     5

>> I(:,4)
ans =
     4
     9
```

Dans ces exemples on a extrait la ligne 1, puis la colonne 4 de la matrice d'origine. On peut selon le même principe extraire une sous-matrice.

```
>> I(1:2,3:4)
ans =
     3     4
     8     9
```

Dans cet exemple on a extrait les lignes 1 à 2 et les colonnes 3 à 4 de la matrice d'origine.

1.2.1.5 Opérations entre les matrices

On peut effectuer un certain nombre d'opérations de base sur les matrices (`Matlab` vérifie que les dimensions des matrices permettent d'effectuer ces opérations entre elles, et affiche un message d'erreur en cas d'incompatibilité, parmi lesquelles :

- Le produit, la division matriciels, respectivement par les opéateurs `*`, `/` (exemple : `I*J`).
- Le produit, la division, la somme, la soustraction matriciels terme à terme, respectivement par les opérateurs `.*`, `./`, `.+` et `.-` (exemple : `I.*J`)
- L'opposé d'une matrice par l'opérateur préfixe `-` (exemple : `-I`)
- La transposée d'une matrice par l'opérateur postfixe `'` (exemple : `I'`)

```
>> I=[1  2;4  6]
I =
     1     2
     4     6

>> J=[4  5;  2  5]
J =
     4     5
     2     5
```

```
>> I*J
ans =
     8    15
    28    50

>> I.*J
ans =
     4    10
     8    30
```

Ne pas confondre une opération matricielle avec la même opération terme à terme (par exemple le cas de la multiplication illustrée ci-dessus).

1.2.1.6 Opération sur une matrice

Ces opérations s'effectuent terme à terme. Il y en a beaucoup de possibles, car les fonctions `Matlab` sont très nombreuses (opération avec un scalaire mais aussi fonctions de base mathématiques, par exemple).

```
>> I=[pi 0 ; -pi pi/2]
I =
    3.1416         0
   -3.1416    1.5708

>> cos(I)
ans =
   -1.0000    1.0000
   -1.0000    0.0000

>> I*2
ans =
    6.2832         0
   -6.2832    3.1416
```

1.2.2 Premières fonctions pour la manipulation d'images

Il est vivement conseillé de consulter l'aide de `Matlab` pour connaître le fonctionnement complet des fonctions citées plus bas.

1.2.2.1 Chargement d'une image : `imread`

La fonction `imread` prend en paramètre le nom d'un fichier, localisé par son chemin à partir du répertoire courant de travail. Les formats acceptés par la fonction sont : BMP, JPEG, GIF, PNG, TIFF, ... Dans le cas d'une image en niveaux de gris, la matrice est une matrice d'entiers (entre 0 et 255) de dimension 2. Dans le cas d'une image en couleurs, le format de base de `MatLab` est une matrice de dimension 3.

```
>> I=imread('pulse.gif');
>> size(I)
ans =
   129   132
```

Ici l'image est de taille (129×132) en niveaux de gris.

```
>> I=imread('fleur.jpg');
>> size(I)
ans =
    320    400      3
```

Ici l'image est une image couleur de taille (320×400) à trois composantes codant la couleur.

N.B : l'appel de la méthode `I = imread(nom)` va renvoyer une matrice I au format `uint8` (entier non signé). les calcul effecté à partir des valeurs de I peuvent alors poser des problèmes liés aux opérations sur les entiers. Pour effectuer des calculs en nombres réels, effectuer la conversion suivante : `I=double(I)`.

1.2.2.2 Enregistrement d'une image : `imwrite`

La fonction `imwrite` prend en premier paramètre la matrice à enregistrer comme image. Le format d'enregistrement est choisi par l'extension qui est fournie dans le nom qui sert de deuxième argument.

```
imwrite(I,'image.gif','gif');
```

Cette instruction sauve le contenu de la matrice I dans une image au format GIF appelée "image.gif".

1.2.2.3 Visualisation d'une image : `image`, `imagesc` ou `imageview`

Ces différentes fonctions permettent de visualiser une image en niveaux de gris ou couleur stockée dans une matrice

```
imagesc(I);
```

Exercice 1 — Premières fonctions personnalisées

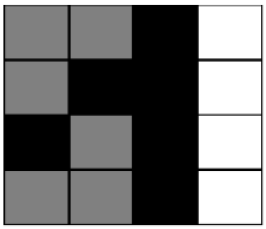
Dans cet exercice, on va travailler systématiquement sur l'image `moon.tif` qui est fournie avec le logiciel `Matlab`.

1. Écrire et tester une fonction `[I,n, m] = ouvrirImage(nom)` qui permet d'ouvrir une image, de la renvoyer ainsi que sa taille `(nbLignes, nbColonnes)`.
2. Écrire et tester une fonction `nb =compterPixels(I,k)` qui renvoie le nombre de pixels de niveau de gris k dans l'image I.
3. Écrire et tester un script `J = remplacerPixels(I,k1,k2)` qui renvoie l'image correspondant à I quand on a changé les valeurs `k1` en `k2`.
4. Écrire et tester une fonction `J = normaliseImage(I,k1,k2)` qui renvoie l'image correspondant à I après changement de l'intervalle de variation des niveaux de gris à $[k1, k2]$.
5. Écrire et tester une fonction `J = inversionImage(I)` qui renvoie l'image inversée (*i.e.* $k' = 255 - k$).
6. Écrire et tester une fonction `h = calculerHisto(I)` qui renvoie un vecteur contenant l'histogramme de I. Regarder l'aide sur la fonction `plot` et tracer cet histogramme.
7. Écrire et tester une fonction `I = seuillerImage(I,s)` qui renvoie l'image I seuillée (les pixels de valeur supérieure à s sont mis à 255, les autres à 0).

8. Quelle série d'instructions (à partir des fonctions déjà écrites) proposer pour pouvoir, à partir d'un nom d'image, l'ouvrir, l'afficher, afficher son l'histogramme, l'inverser, calculer son histogramme après inversion et visualiser le résultat des traitements.
9. Quelle série d'instructions proposer pour pouvoir, à partir d'un nom d'image, l'ouvrir, l'afficher, afficher son l'histogramme, la normaliser dans $[10, 50]$, calculer son histogramme après normalisation et visualiser le résultat des traitements. Interpréter le résultat de la visualisation.
10. Quelle série d'instructions proposer pour pouvoir, à partir d'un nom d'image, l'ouvrir, l'afficher, afficher son l'histogramme, la seuiller par un seuil $s = 128$, calculer son histogramme après seuillage et visualiser le résultat des traitements.

Exercice 2 — Génération d'images quelconques

1. Créer la matrice I qui représente l'image suivante :



où un pixel noir a la valeur 0, un pixel gris la valeur 128 et un pixel blanc la valeur 255. La visualiser avec `imagesc` et regarder son histogramme avec la fonction `calculerHisto(I)` précédente.

2. Pour générer une matrice de nombres aléatoires, il est possible d'utiliser la fonction `rand` (distribution uniforme), la fonction `randn` (distribution gaussienne), et la fonction `randi` (distribution uniforme sur les entiers). Utiliser ces fonctions pour générer une matrice de taille (512×512). Pour `randn`, tester avec une moyenne de 128 et un écart-type variable. Visualiser l'image obtenue et son histogramme. Que remarquez-vous sur cet histogramme ?

Exercice 3 — Manipulation d'images

Dans cet exercice, on va travailler sur l'image `pout.tif` qui est fournie avec le logiciel `Matlab`

1. Ouvrir cette image et la visualiser.
2. Regarder l'histogramme de cette image. Déterminer le niveau de gris minimum et le niveau de gris maximum de cette image. Que constatez-vous ?
3. A partir des fonctions de l'exercice 1, mettre au point une fonction `egalisationHistogramme(I,h)` qui, à partir de l'image I et de son histogramme h, renvoie l'image après égalisation.
   N.B : il faudra pour cela calculer l'histogramme cumulé.

Semaine 2

Transformée de Fourier

2.1 TD 2 : outils mathématiques

Transformée de Fourier (TF) continue : rappels de cours

On rappelle ici la définition des transformées de Fourier 1d et 2d :

1. Transformée de Fourier continue monodimensionelle : on condidère un signal $x(t)$ continu. La transformée de Fourier de $x(t)$, notée $X(f)$, se calcule de la manière suivante :

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-2i\pi ft} dt \tag{2.1}$$

2. Transformée de Fourier continue bidimensionnelle : on condidère un signal $x(t,u)$ continu. La Transformée de Fourier de $x(t,u)$, notée $X(f,g)$, se calcule de la manière suivante :

$$X(f,g) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x(t,u) e^{-2i\pi(ft+gu)} dtdu \tag{2.2}$$

Exercice 1 — Propriétés fonctionnelles de la TF continue

Quelques propriétés classiques des TF 1d/2d, utiles pour le TD/TME, sont rassemblées dans le tableau 2.1. Nous proposons ici d'en démontrer certaines.

1. Montrer que la TF 2d revient à appliquer successivement deux transformées de Fourier 1d, *i.e.* $X(f,g) = TF\left[Z(f,u)\right]$. Expliciter $Z(f,u)$.
2. Montrer que, dans le cas où la fonction 2d est séparable en un produit de 2 fonctions 1d, *i.e.* $x(t,u) = z(t) \cdot k(u)$, on a : $X(f,g) = Z(f) \cdot K(g)$, avec $Z(f) = TF\left[z(t)\right]$ et $K(g) = TF\left[k(u)\right]$.
3. Montrer les propriétés liant convolution / produit en 1d :
   $TF\left[x(t) \star y(t)\right] = X(f) \cdot Y(f)$ et $TF\left[x(t) \cdot y(t)\right] = X(f) \star Y(f)$. On rappelle que l'opérateur $\star$ est le produit de convolution entre signaux. Considérons $x(t)$ et $y(t)$, on a :

$$z = x \star y = \int_{-\infty}^{+\infty} x(\tau) y(t-\tau) d\tau \tag{2.3}$$

4. Montrer la propriété de rotation dans le cas 2D :
   $TF\left[x\left(t\cos\theta+u\sin\theta,-t\sin\theta+u\cos\theta\right)\right]=X\left(f\cos\theta+g\sin\theta,-f\sin\theta+g\cos\theta\right)$

	TF continue 1D		TF continue 2D	
	x(t)	X(f)	x (t,u)	X(f,g)
linéarité	$x(t)+\lambda y(t)$	$X(f)+\lambda Y(f)$	$x(t,u)+\lambda y(t,u)$	$X(f,g)+\lambda Y(f,g)$
translation	$x(t-t_0)$	$X(f)\ e^{-2i\pi f t_0}$	$x(t-t_0,u-u_0)$	$X(f,g)\ e^{-2i\pi(ft_0+gu_0)}$
contraction	$x(\alpha t)$	$\frac{1}{\lvert\alpha\rvert}X(\frac{f}{\alpha})$	$x(\alpha t,\beta u)$	$\frac{1}{\lvert\alpha\rvert\lvert\beta\rvert}X(\frac{f}{\alpha},\frac{g}{\beta})$
convolution	$x(t)\star y(t)$	$X(f)\cdot Y(f)$	$x(t,u)\star y(t,u)$	$X(f,g)\cdot Y(f,g)$
produit	$x(t)\cdot y(t)$	$X(f)\star Y(f)$	$x(t,u)\cdot y(t,u)$	$X(f,g)\star Y(f,g)$

TABLE 2.1: Propriétés des TF continues 1D et 2D

Exercice 2 — Calcul de la TF continue de quelques signaux usuels

1. À partir de l'équation 2.1, calculer la TF de la fonction $x(t)=Rect\left(\frac{t}{T}\right)$, où $Rect(t)$ est la fonction "Porte" :

$$Rect(t)=\begin{cases}1 & \text{si } |t|\leq\frac{1}{2}\\ 0 & \text{sinon}\end{cases} \tag{2.4}$$

   En déduire la TF du signal correspondant en 2D : $x(t,u)=Rect\left(\frac{t}{T}\right)\cdot Rect\left(\frac{u}{T}\right)$
2. Soit la distribution de Dirac $\delta(t)$ (voir annexe 2.1.1) :

$$\delta(t)=\begin{cases}O & \text{si } t\neq 0\\ \infty & \text{sinon}\end{cases}$$

   On rappelle que $TF\left[\delta(t-t_0)\right]=e^{-2i\pi f t_0}$ et $TF\left[e^{2i\pi f_0 t}\right]=\delta(f-f_0)$
   En déduire :
   (a) la transformée de Fourier des signaux $x(t)=\cos\left(2\pi f_0 t\right)$ et $x(t)=sin\left(2\pi f_0 t\right)$;
   (b) la transformée de Fourier d'un pic de dirac bidimensionnel : $\delta(t-t_0,u-u0)$;
   (c) la transformée de Fourier de la fonction 2d suivante :
   $s_\theta(x,y)=A\cos\left[2\pi f_o\left(x\cos(\theta)+y\sin(\theta)\right)\right]$

2.1.1 Annexe : distribution de Dirac

Intiuitivement, la distribution de Dirac peut être interprétée de la manière suivante : $\delta(t)=\lim_{T\to 0}\left[\frac{1}{T}Rect\left(\frac{t}{T}\right)\right]$.

Propriétés de la distribution de Dirac

— $\int_{-\infty}^{+\infty}\delta(t)dt=1.$
— $x(t)\cdot\delta(t-t_0)=x(t_0)\delta(t-t_0)$
— $x(t)\star\delta(t-t_0)=x(t-t_0)$
— scaling property : $|\alpha|\cdot\delta(\alpha t)=\delta(t)$
— $TF\left[\delta(t-t_0)\right]=e^{-2i\pi f t_0}$ et $TF\left[e^{2i\pi f_0 t}\right]=\delta(f-f_0)$

2.2 TME 2 : Transformées de Fourier sur les images réelles

Dans ce TME, on se propose de visualiser la transformée de Fourier (TF) sur les images de la figure 2.1, et d'analyser les caractéristiques du spectre (*i.e.* fréquences spatiales) extraites.
Pour cela, on s'appuiera sur les fonctions `Matlab` suivantes :

- La fonction `fft2` permet de calculer la transformée de Fourier (TF) d'une image.
- Pour visualiser les basses fréquences au centre de l'image, on utilisera la fonction `fftshift`.
- La TF étant une fonction complexe, on ne peut la visualiser directement, on choisira de visualiser son module (utiliser la fonction `abs`).
- Dans la plupart des images naturelles, les hautes fréquences ont une énergie négligeable par rapport aux basses fréquences. Afin d'avoir une visualisation satisfaisante, on choisira une échelle logarithmique et on visualisera : $\log(1 + \text{abs}(TF))$.

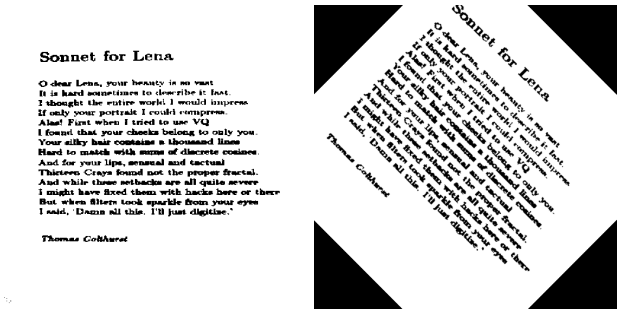


FIGURE 2.1: La Transformée de Fourier : un outil pour l'analyse fréquentielle des images

Exercice 1 — Propriétés de la TF sur des images réelles

1. Récupérer les images `son.gif` et `sonrot.gif`
2. Écrire les fonctions matlab suivantes :
   - `[If] = compute_FT(I)` qui calcule la transformée de Fourier de l'image I.
   - `[Ifv] = to_visualize_TF(If)` qui prend en paramètres une matrice If contenant la transformée de Fourier d'une image, et renvoie le module centré de la transformée (pour visualisation).
   - `[Ifv] = to_visualize_TF_log(If)` qui effectue la même opération que précédemment, en échelle logarithmique.
3. Écrire un script matlab `testProp.m` qui sera le programme principal pour vérifier certaines propriétés de la TF. On effectuera les traitements suivants :
   - Ouverture des deux images et calcul de leurs transformées de Fourier.
   - Visualisation des spectres en échelle logarithmique
   - Binarisation des spectres avec le seuil (avec un seuil de $3 \cdot 10^5$). Utiliser la fonction `J = image_binarization(I,seuil)` mise en place lors du TME 1.
   - Visualisation des spectres seuillés.

4. Interprétation des résultats : que constatez vous sur les modules seuillés ? Quelle(s) propriété(s) la forme des spectre de deux images illustre-t-elle ?
5. On demande maintenant d'écrire une fonction `I3 = blend(I1,I2, alpha)` qui prend 2 images en paramètres et un réel α et qui renvoie l'image résultant du traitement suivant $I_3(t,u) = \alpha I_1(t,u) + (1-\alpha)I_2(t,u)$ (on appliquera la fonction `double(I)` sur chacune des deux images d'entrée pour effectuer des calculs réels). À la suite du script `testProp` précédent :
   (a) Appelez la fonction `blend` avec les deux images précédentes ($\alpha = 0.5$, par exemple).
   (b) Calculez la TF de l'image résultant du traitement.
   (c) Seuillez son module en utilisant la fonction précédente.
   (d) Visualisez le module seuillé.
6. Comparer le module seuillé de l'image combinée aux modules seuillés des deux images initiales. Quelle propriété de la TF la forme du spectre illustre-t-elle ?
7. L'objectif final est de déterminer l'orientation du texte dans l'image, afin de produire une image traitée où le texte est horizontal. On demande de mettre en place une fonction `[Ir,ordom] = rectify_orientation(I)` qui effectue les opérations suivantes sur l'image I passée en paramètre :
   (a) Calcul du module M de la TF de I et seuillage de M pour obtenir une image binaire Mb, puis mise au centre des basses fréquences (`fftshift`).
   (b) Calcul de l'orientation dominante de Mb, en utilisant la fonction `[Ior,ordom] = orientationDominante(Mb)` fournie. N.B. : Cette fonction calcule l'orientation en chaque pixel Ior de l'image Mb, ainsi que l'orientation dominante ordom (*i.e.* la plus fréquente), exprimée en degrés.
   (c) D'inverser l'orientation estimée `ordom` afin de produire une image où le texte est horizontal (utiliser la fonction `rotationimage` fournie)
8. Mettre en place un script `rectifcationOrientation` qui teste la fonction précédente sur l' image `sonrot.gif` (ou sur des images tournée artificiellement en utilisant `rotationimage` sur `son.gif`).
   Bonus : tester le script `rectifcationOrientation` sur l'autre image fournie (port.jpg).

Semaine 3

Numérisation - transformée de Fourier avancée

3.1 TD 3 : impact de la numérisation

Numérisation des signaux et des images : rappels de cours

Les signaux et images qu'on manipule en informatique sont des objets issus d'un processus de **numérisation**. Le passage du continu au discret va influencer leur analyse fréquentielle de trois manières principales :

1. Analyse du signal sur un domaine (temporel, spatial) limité : quel est le lien entre la TF d'un signal continu à bande infinie et le signal continu à bande limité ? (voir exercice 2).
2. Échantillonnage du signal continu, *i.e.* prélèvement d'un ensemble dénombrable de valeurs : quelle est la forme de la TF du signal échantillonné ? Y a-t-il perte d'information entre le signal continu et le signal numérique ? Dans quel(s) cas ? (voir exercice 1).
3. Quantification du signal (cours) & Transformée de Fourier Discrète (voir exercice 3) .

Exercice 1 — Echantillonnage du signal continu

L'échantionnage idéal consiste à prélever du signal continu un ensemble de points régulièrement espacés pour former un signal discret $x_e(t)$.
Mathématiquement, le modèle idéal en 1d s'écrit (voir annexe 3.1.1.2) :

$$x_e(t) = \sum_{k=-\infty}^{+\infty} x(t)\delta(t - kT_e) = x(t) \cdot \text{⨿}_{T_e}(t) \tag{3.1}$$

où T_e est la période d'échantillonnage, et $\text{⨿}_{T_e}(t)$ est un "peigne de dirac", correspondant à la périodisation d'un pic de Dirac à la période T_e (voir annexe 3.1.1.1).

1. Montrer que le spectre du signal échantillonné $x_e(t)$ s'écrit :

$$X_e(f) = \frac{1}{T_e} \sum_{k=-\infty}^{+\infty} X(f - kf_e) \tag{3.2}$$

Indication : On utilisera les propriétés de l'annexe 3.1.1.3.
Interprétation : le spectre du signal échantionné $x_e(t)$ correspond au spectre du signal continu $x(t)$ périodisé à la période f_e.

2. **Reconstruction du signal continu à partir du signal numérique.**

   (a) Proposer un traitement dans le domaine fréquentiel permettant de retrouver le spectre $X(f)$ du signal continu à partir du spectre $X_e(f)$ du signal échantionné. Quelle condition entre $X(f)$ et f_e doit être satisfaite pour que cette opération soit possible ? Ces conditions sont connues sous le nom de **théorème de Shannon**.

   (b) En supposant les conditions de Shannon vérifiées, il est possible de reconstruire le signal continu $x(t)$ à partir de ses échantillons $x_e(kTe)$, en inversant la formule 3.2. Montrer que la formule de reconsutruction s'écrit :

$$x_r(t) = \sum_{k=-\infty}^{+\infty} x(kT_e)\frac{sin(\pi f_e(t-kT_e))}{\pi f_e(t-kT_e)} = \sum_{k=-\infty}^{+\infty} x(kT_e)sinc(\pi f_e(t-kT_e)) \qquad (3.3)$$

3. **Recouvrement spectral.** Quand le critère de Shannon n'est pas vérifié (car f_e est trop faible ou quand il est impossible à satisfaire), quelle est la conséquence sur le spectre du signal échantillonné ?

   Exemple : étude d'un signal sinusoïdal
   on considère le signal sinusoïdal suivant : $c(t) = A\cos(2\pi f_0 t)$.

   (a) On échantillonne ce signal à la fréquence f_e pour extraire un signal numérique $c_e(k)$, $k \in \{1; N\}$. Exite-t-il une gamme de pas d'échantillonnage permettant une reconstruction du signal ? Si oui, la préciser.

   (b) On échantillonne le signal aux fréquences $4f_0$ et $2f_0$. Représenter graphiquement le spectre des signaux échantillonnés à ces fréquences.

   (c) On décide maintenant d'échantillonner $c(t)$ à $f_e = \frac{3*f_0}{2}$. Représenter le spectre de $c_e(t)$. Quel est le phénomène observé ? Quel va être le signal reconstruit si on applique la formule 3.3 ?

Exercice 2 — Signaux à support temporel limité.

On considère un signal continu $x(t)$, et le signal $x_L(t)$ correspondant à $x(t)$ mais réduit à l'intervalle $\left[-\frac{L}{2}; -\frac{L}{2}\right]$, $L \in \mathbb{R}$, *i.e.* : $x_L(t) = x(t) \cdot Rect\left(\frac{t}{L}\right)$, où $Rect(t)$ est la fonction porte :

$$Rect(t) = \begin{cases} 1 & \text{si } |x| \leq \frac{1}{2} \\ 0 & \text{sinon} \end{cases} \qquad (3.4)$$

1. Exprimer $X_L(f) = TF\left[x_L(t)\right]$ en fonction de $X(f)$.
2. Calculer $X_L(f)$ pour $x(t) = Acos(2\pi f_o t)$. Représenter graphiquement $|X_L(f)|$.
3. Est-il théoriquement possible de retrouver $X(f)$ à partir de $X_L(f)$? Dans notre exemple, proposer un méthode pratique permettant de garantir que $X_L(f)$ sera "peu" dégradé par rapport à $X(f)$.
   Indication : l'amplitude des lobes secondaires de la fonction sinus cardinal est faible par rapport à celle du lobe principal. On peut donc considérer que l'amplitude de $sinc(t)$ de vient nulle au delà de k lobes secondaires.

4. **Bonus** : Reprendre la question en 2D : comparer $x(t,u)$ et $x_L(t,u) = x(t,u) \cdot Rect\left(\frac{t}{L}\right) \cdot Rect\left(\frac{u}{L}\right)$, dans le cas où : $x(t) = A\cos\left[2\pi f_o\left(x\cos(\theta) + y\sin(\theta)\right)\right]$.

Exercice 3 — TFD et Résolution fréquentielle

L'objectif de cet exercice est de voir le résultat des trois étapes de numérisation sur un exemple particulier. On considère le signal sinusoïdal suivant : $c(t) = A\cos(2\pi f_0 t) Rect\left(\frac{t}{L}\right)$, signal fenêtré de l'exercice 3 où la TF a été calculée. On considérera ici que $L = 2T_0$

1. On échantillonne $c(t)$ à la fréquence $f_e = 4f_0$. Représenter graphiquement le signal échantilonné $c_e(t) = \sum_k c(t)\delta(t - kT_e)$, et le signal numérique $c(k)$.
2. Représenter la TF du signal échantillonné.
3. On rappelle que la TFD consiste à échantillonner la TF du signal échantillonné dans l'intervalle $[-\frac{f_e}{2}; \frac{f_e}{2}[$. Représenter la TFD $X(h)$ de $c(k)$. Expliquer pourquoi les lobes secondaires du sinus cardinal n'apparaissent pas.
4. On échantillonne maintenant $c(t)$ à la fréquence $f_e = 3.7f_0$. Calculer $X(h)$ et faire un graphique. Conclure.
5. Zéro-Pading : on forme le signal numérique de $2N$ valeurs $p(k)$ suivant :

$$p(k) = \begin{cases} x(k) & \text{si } k \in \{0; N-1\} \\ 0 & \text{si } k \in \{N; 2N-1\} \end{cases} \tag{3.5}$$

Calculer la TFD $P(h) = TFD\left[p(k)\right]$, et la comparer à $X(h)$. Conclure sur l'effet du zéro padding.

3.1.1 Annexe : échantillonnage

On rappelle que la distribution de Dirac $\delta(t)$ est définie de la manière suivante :

$$\delta(t) = \begin{cases} O & \text{si } t \neq 0 \\ \infty & \text{sinon} \end{cases} \tag{3.6}$$

3.1.1.1 Peigne de Dirac

On appelle "peigne de dirac" la périodosation d'un pic de Dirac à la période T_e :
$\text{⨿}_{T_e}(t) = \sum_{k=-\infty}^{+\infty} \delta(t - kT_e)$.

3.1.1.2 Modèle d'échantillonnage

L'échantillonnage idéal d'un signal $x(t)$ 1d à la période d'échantillonnage T_e s'écrit :

$$x_e(t) = \sum_{k=-\infty}^{+\infty} x(t)\delta(t - kT_e) = x(t) \cdot \text{⨿}_{T_e}(t) \tag{3.7}$$

Avec ce modèle d'échantillonnage idéal, on a : $\lim_{Te \to 0} x_e(t) = x(t)$ et $\lim_{Te \to 0} \int_{-\infty}^{+\infty} x_e(t) = \int_{-\infty}^{+\infty} x(t)$

3.1.1.3 Transformées de Fourier des distributions de Dirac et de quelques fonctions usuelles

Signal	TF	Signal	TF
1	$\delta(f)$	$\delta(t)$	1
$e^{-2i\pi f_0 t}$	$\delta(f-f_0)$	$\delta(t-t_0)$	$e^{-2i\pi f t_0}$
$\text{⨿}_{T_e}(t) = \sum_{k=-\infty}^{+\infty} \delta(t-kT_e)$	$\frac{1}{T_e}\text{⨿}_{f_e}(f) = \frac{1}{T_e}\sum_{k=-\infty}^{+\infty} \delta(f-\frac{k}{T_e})$		
$Rect\left(\frac{t}{T}\right)$	$Tsinc(\pi f T)$	$f_0 sinc(\pi f_0 t)$	$Rect\left(\frac{f}{f_0}\right)$

3.2 TME 3 : échantillonnage des images & aliasing

Échantillonnage des signaux 2D

Échantillonnage des signaux

Les propriétés démontrées en TD se généralisent facilement au cas bidimensionel. On admettra donc les résultats suivants. En considérant une grille carrée, l'échantillonnage idéal d'un signal 2d est modélisé de la manière suivante :

$x_e(t,u) = x(t,u) \cdot e(t,u)$ où $e(t,u) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \delta(t - kT_e, u - lT_e)$ est la brosse de Dirac.

Le spectre du signal échantillonné s'écrit :

$$X_e(f,g) = \frac{1}{T_e^2} \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} X(f - kf_e, g - lf_e) \tag{3.8}$$

En 2d, l'échantillonnage conduit à une périodisation du spectre dans les 2 dimensions. A partir du signal échantillonné, il est possible de reconstruire le signal analogique si celui-ci est à bande limitée (version 2d du thérorème de Shannon). Dans ce cas, la formule de reconstruction s'écrit :

$$x_r(t,u) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} x(kT_e, lT_e) sinc(\pi f_e(t - kT_e)) sinc(\pi f_e(u - lT_e)) \tag{3.9}$$

Exercice 1 — Aliasing et fenêtrage pour des signaux 2d

On considère le signal suivant, dont un exemple est montré à la figure 3.1 ($\theta = 45^\circ$) :

$$s_\theta(t,u) = A \cos\left[2\pi f_o \left(t\cos(\theta) + u\sin(\theta)\right)\right] \tag{3.10}$$

Le but de ce TME est d'étudier les conditions limites d'échantillonnage de cette image afin de ne pas introduire d'Aliasing.

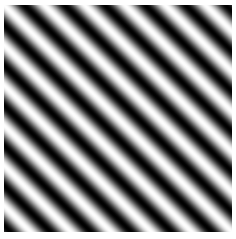


FIGURE 3.1: Une sinusoïde 2d de synthèse

1. La fonction `sinus2d(A, theta,L, TO,Te)` fournie permet d'échantillonner la fonction précédente :

— à une période d'échantillonnage T_e.
N.B : on suppose ici qu'on a une grille carrée d'échantillonnage, *i.e.* on échantillonne à la même période T_e les axes t et u de l'image.
— avec une période $T_0 = \frac{1}{f_0}$ de la sinusoïde (voir équation 3.10)
— sur une fenêtre spatiale L
— avec l'angle θ (voir équation 3.10)
— avec une amplitude A (voir équation 3.10)

Utiliser cette fonction pour générer $s_\theta(t,u)$ avec $A = 10$, $\theta = 45^\circ$, $L = 512$, $T_0 = 64$ et $T_e = 1$.

2. Quelle est la fréquence maximale du signal en t (notée f_t^{max}) et u (notée f_u^{max}) ? On notera $f_m = max(f_t^{max}, f_u^{max})$. Expliquer pourquoi f_m est la fréquence qui définira la fréquence d'échantillonnage limite.
   (a) On échantillonne $s_\theta(t,u)$ avec $f_e = 16 \cdot f_m$. Visualiser l'image échantillonnée.
   (b) Utiliser la fonction `fft2` pour calculer la Transformée de Fourier Discrète 2d de l'image précédente. Visualiser cette image, en utilisant les fonctions `imagesc` ou `mesh(3d)`.
   N.B : pour visualiser les basses fréquences au centre de l'image, utiliser la fonction `fftshift` avant la visualisation.
   (c) Commenter le forme de la DFT :
   — Vérifier la présence des deux pics en fréquence.
   — Faire varier la période d'échantillonnage T_e de la sinusoïde et observer la différence sur le spectre (DFT). D'où provient cette différence alors qu'on visualise la Transformée de Fourier de la même fonction continue ?
   — **Question bonus (pour la fin)** :
   — Pourquoi n'observe-t-on pas exactement deux pics sur la DFT ? D'où proviennent les lobes supplémentaires sur le spectre ?
   — Augmenter T_0 jusqu'à obtenir un spectre avec un unique pic. Expliquer l'origine du phénomène. Vérifier la valeur limite de L à partir de laquelle les deux pics interfèrent.
3. On demande maintenant d'échantillonner $s_\theta(t,u)$ avec $f_e = 4 \cdot f_m$.
   (a) Écrire une fonction `Matlab imager = reconstruction(imagee,Te,L)` permettant d'effectuer la reconstruction décrite à l'équation 3.9 à partir des échantillons.
   N.B : on reconstruira une image avec autant de pixels que l'image de départ (*i.e.* L^2).
   (b) Écrire une fonction `eps = erreur(imager,imaged)` pour calculer l'erreur moyenne relative de reconstruction : $\varepsilon_r = \frac{1}{2A \cdot L^2} \sum_{k=1}^{L} \sum_{l=1}^{L} |x_r(k,l) - x_d(k,l)|$
   (c) Écrire un script permettant d'échantillonner $s_\theta(t,u)$ à $f_e = 4 \cdot f_m$, de reconstruire l'image et de calculer l'erreur. Afficher l'image reconstruite. Conclure. D'où vient l'erreur de reconstruction ?
4. Reprendre la question 4 pour $f_e = \frac{3}{2} \cdot f_m$. Commenter l'effet de l'Aliasing.
5. On se place maintenant dans le cas $\theta = 10^\circ$. Que vaut-alors f_m ? En échantillonnant à $f_e = \frac{3}{2} \cdot f_m$, quelle est la perturbation supplémentaire apparaissant lors de la reconstruction ? Expliquer.

Exercice 2 — Effets de l'Aliasing sur des images réelles

Dans cette partie, on utilisera l'image `barbara.png` fournie. On souhaite analyser l'impact visuel du sous-échantillonnage sur cette image. Mettre en place un script `TestsubSampling` qui va effectuer un sous-échantillonnage de l'image courante d'un facteur 2 (utiliser pour cela la fonction `subSampling2()` fournie).

- Itérer le sous-échantillonnage et oberver l'image résultante.
- Calculer les transformées de Fourier de l'images intiale et sous-échantillonnée à chaque itération.
- Décrire et interpréter les effets de l'alisasing sur l'image sous-échantillonnée (pour cela, s'appuyer sur le calcul et la visualisation des transformées de Fourier). En quoi ces effets sont-ils visuellement gênants ?
- Bonus : renouveler les expérimentations sur l'image `mandrill.png` fournie.

Semaine 4

Filtrage d'images

4.1 TD 4 : convolution et filtrage

Filtrage spatial & fréquentiel d'images : rappels de cours

Filtrer une image consiste à effectuer un traitement de son contenu fréquentiel, pour par exemple :

1. Ne conserver que les basses fréquences : filtrage passe-bas (*e.g.* atténuer le bruit)
2. Ne conserver que les hautes fréquences : filtrage passe-haut (*e.g.* détection de contour, séance TD-TME 5)
3. Ne conserver qu'une certaine gammes de fréquences : filtrage passe-bande

En considérant une image $x(t,u)$, le filtrage fréquentiel correspond à multiplier le spectre (cad la Transformée de Fourier) $X(f,g)$ par une fonction $H(f,g)$ pour obtenir un spectre filtré $X_F(f,g)$:

$$X_F(f,g) = X(f,g) \cdot H(f,g) \tag{4.1}$$

Dans le domaine spatial, ce traitement fréquentiel corrrespond à un produit de convolution (noté $\star$) entre $x(t,u)$ et $h(t,u)$ (transformée de Fourier inverse de $H(f,g)$) :

$$x_F(t,u) = x(t,u) \star h(t,u) \tag{4.2}$$

$h(t,u)$, appelé réponse implusionnelle du filtre, correspond à l'image filtrée si l'entrée est un pic de dirac centré en $(0,0)$ (*i.e.* $x(t,u) \leftarrow \delta(t,u)$). Lorsqu'on considère des signaux (images) discrets, et que le support du filtre h est à Réponse Impulsionnelle Finie (RIF) de taille d (impaire), on obtient :

$$x_F(t,u) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} x(i,j)h(n-i,m-j) \tag{4.3}$$

Une méthode pratique pour effectuer le filtrage entre une image x et un filtre de réponse impulsionnele h consiste donc à :

1. Faire une rotation de π du noyau par rapport à son centre : $h(n,m) \Rightarrow h(-n,-m) = g(n,m)$
2. Centrer le filtre sur p en le superposant à l'image
3. Effectuer la somme pondérée entre les pixels de l'image et les coefficients du filtre $g(n,m)$
4. Le pixel p dans l'image but (filtrée) aura comme valeur cette somme pondérée

Exercice 1 — Filtres

1. Quel filtre a pour réponse impulsionnelle $h_1(u,v)$ une porte bidimensionnelle centrée de longueur 3 ?
2. Soit le filtre $h_2(u,v) =$ | 1 | 0 | -1 |
| --- | --- | --- |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

. Écrire la réponse impulsionnelle $h(u,v)$ en fonction d'une somme d'impulsions de Dirac décalées.
3. Montrer que le filtre h_2 est séparable et donner les deux filtres unidimensionnels tels que $h_2 = (f_1)^T.f_2$
4. **Effets de bord.** Un pixel situé sur le bord de l'image n'a pas de voisinage en dehors de l'image. On ne peut donc pas, *a priori*, calculer directement l'élément de bord obtenu après convolution. Proposez des méthodes pour traiter tout de même les bords.
5. Soit l'image $I =$ | 1 | 2 | 3 |
| --- | --- | --- |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

. Calculer $I \star h_2$, $(I \star f_1) \star f_2$. Que peut-on en conclure ?

   — **Généralisation** : on considère un filtre linéaire séparable, *i.e.* $h(n,m) = f_1(n) \cdot f_2(m)$. Monter que dans ce cas le filtrage 2d peut se décomposer en deux filtrage sucessifs 1d.

Exercice 2 — Différents filtres

On considère l'image de taille 8×8, numérisée selon 8 niveaux de gris (de 0 à 7), suivante :

0	0	0	0	0	0	0	0
0	5	5	5	5	5	5	0
0	5	7	7	7	7	5	0
0	5	7	7	7	7	5	0
0	5	7	7	7	7	5	0
0	5	7	7	7	7	5	0
0	5	5	5	5	5	5	0
0	0	0	0	0	0	0	0

1. Dessiner le profil des niveaux de gris de la ligne 3 de cette image (rappel : les lignes sont numérotées à partir de 0)
2. Réaliser le filtrage de cette image en utilisant le filtre $\frac{1}{9} \times$ | 1 | 1 | 1 |
| --- | --- | --- |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

, en précisant les valeurs des niveaux de gris de l'image filtrée (on utilisera une convolution linéaire). Dessiner le profil des niveaux de gris de la ligne 3 de cette image, que remarquez-vous ?

3. Mêmes question que précédemment en utilisant maintenant le filtre

0	-1	0
-1	4	-1
0	-1	0

4. On souhaite réaliser un filtrage en utilisant un filtre gaussien d'écart type $\sigma = 0.5$. Calculer les coefficients de ce filtre. Tracer le profil de niveau de gris selon la ligne centrale du filtre. Mêmes questions avec $\sigma = 1$.
5. On considère que la fonction gaussienne est nulle après $3 \cdot \sigma$. Quelle taille de masque gaussien dans le domaine spatial appliquer pour avoir un filtre de fréquence de coupure (domaine fréquentiel)= $\frac{(fe)}{2}$?

Exercice 3 — Convolution

Nous allons appliquer l'opérateur de convolution sur une petite portion (rectangle rouge) de l'image ci-dessous, contenant une partie du tangon du bateau (partie servant à amarrer les embarcations) ainsi qu'un filin, juste en-dessous.

La matrice I suivante contient les luminance de la zone en question, de taille 20×11.

$I=$

196	196	199	203	200	188	195	201	204	198	200
198	212	188	206	199	202	204	201	204	187	213
199	187	206	188	209	181	213	208	208	194	186
198	218	202	198	194	210	195	195	208	163	128
184	194	187	206	207	223	202	181	149	125	81
207	195	206	217	205	195	174	151	108	70	74
205	198	187	193	190	161	133	94	80	62	210
179	196	205	186	155	119	75	66	139	223	209
204	196	185	115	91	77	60	189	202	208	188
208	159	105	95	68	90	222	214	206	199	191
146	100	87	55	151	210	210	200	192	207	201
84	69	64	207	212	212	194	211	195	196	147
60	108	219	194	198	187	188	204	191	214	208
174	218	209	195	203	190	205	192	201	201	192
200	215	201	187	204	195	203	192	210	198	148
207	194	192	184	198	197	188	205	164	161	208
176	191	200	195	191	197	207	180	176	217	198
197	190	201	200	208	195	154	171	198	198	192
193	189	205	198	216	141	192	203	194	199	196
196	217	207	186	146	207	211	188	203	188	180

1. Quelles sont, dans cette matrice, les zones de pixels appartenant au tangon et au filin ?
2. **Filtrage passe-bas.** On considère un filtre passe-bas (binomial) 3×3, dont le maque de convolution est le suivant :

$$h_1 = \frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

   L'application de ce maque sur la zone 20×11 considérée délivre le résultat partiel suivant :

111	149	149	150	149	146	148	150	150	149	113
149	199	198	199	199	198	201	203	201	197	149
149	200	199	198	198	199	202	204	200	188	135
148	200		198	201	203	200	196	185	161	105
146	197	199	203	206	205	194	175	151	121	75
149	197	199	203	201		169	142	112	96	75
148	197	196	193	181	157	129	108	102		113
145	195	189	170	146	117	98		140	175	146
146	185	163	131	105	96	115	155	187	200	149
134	154	121	98		126	169	197	202	200	147
99	111	95	109	143	178	201	204	200	196	143
68		117	158	189	200	201	200	199	193	138
78	130	169	193	199	196	196		199	198	145
123	185	200	199	196	195	196	197	200	198	143
150	204	200	194	196	197	197	196	194	187	134
148	197	194		194	196	196	191		184	141
142	192	195	195	196	194	189	183	184	193	149
142	193	197	201	197	186	179	182	191	198	148
146	197	200	198	189		184	191	195	195	144
112	153	151	141	133	139	147	148	147	144	105

   Compléter les zones grisées afin d'obtenir la matrice résultat (arrondir à la partie entière de l'élément calculé).

3. **Filtre rehausseur de contraste.** On souhaite la qualité de cette portion d'image en accentuant le contraste sur les bords des objets. Pour cela, on utilise le filtre d'amélioration dont le masque de convolution est le suivant :

$$h_2 = \frac{1}{16} \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

   L'application de ce masque sur la zone considérée donne le résultat partiel suivant :

586	373	408	410	410	343	382	405	417	399	589
383	291	117	252	178	238	209	188	220	126	492
412	100	265	121	283	71	277	223	226	226	395
389	309	201	200	[redacted]	257	155	183	325	160	210
321	186	127	221	207	301	237	208	123	162	78
451	170	244	275	216	212	189	198	90	[redacted]	9
441	207	133	185	236	168	161	40	-3	-273	705
290	202	271	262	189	127	[redacted]	-167	[redacted]	497	424
437	236	304	18	40	25	-263	403	268	228	332
531	186	[redacted]	132	[redacted]	-127	536	253	223	183	367
338	39	[redacted]	-265	210	387	224	173	152	247	460
145	-11	-262	510	292	257	149	262	[redacted]	217	130
-66	[redacted]	520	151	194	147	[redacted]	238	141	274	487
392	384	212	182	228	160	252	158	211	200	403
404	262	202	151	[redacted]	181	235	150	295	270	142
465	165	181	148	214	207	128	301	[redacted]	[redacted]	533
285	195	221	200	157	195	316	141	121	352	373
426	172	210	198	238	275	[redacted]	120	251	184	368
383	140	230	183	387	-105	251	270	167	219	409
570	493	427	379	121	537	468	323	445	358	516

Complétez les zones grisées afin d'obtenir la matrice résultat.

4.2 TME 4 : filtrage fréquentiel et couleur

Pour ces exercices, on utilisera les images `mandrill.png`, `lena.jpg` et `barbara.jpg`.

Exercice 1 — Filtrage fréquentiel

1. Visualiser le module centré de la transformée de Fourier de l'image `mandrill.png`.
   Rappels :
   — On utilisera les fonctions `fft2`, `fftshift` (pour placer les basses frquences au centre de l'image), et `abs` (pour extraire le module de la fft).
   — Pour se ramener à une échelle de valeurs convenables, si on appelle F le spectre centré, on visualisera $1 + \log(F)$.
2. Écrire une fonction `Ff = filtrePasseBasIdeal(n,m,fc)` qui calcule un filtre passe-bas idéal, de taille $n \times m$ et de fréquence de coupure f_c
   — Cette fonction mettra à 0 tous les "pixels" dont la distance au centre (correspondant aux basses fréquences) sera supérieure à un seuil donné (défini par f_c).
3. Écrire un script `Filtrage` dans lequel on effectuera le filtrage passe-bas de l'image I en se servant de la fonction `filtrePasseBasIdeal(n,m,fc)` précédemment écrite. On devra donc :
   — Ouvrir l'image de départ et calculer sa Transformée de Fourier.
   — Filtrer la Transformée de Fourier après application de la fonction `fftshift`.
   Rappel : le filtrage dans le domaine fréquentiel consiste à multiplier terme à terme le filtre et la TF de l'image originale.
   — Inverser le shift de la FFT filtrée, puis calculer sa transformée de Fourier inverse.
   N.B. : on utilisera pour cela les fonctions `ifft2` et `ifftshift` qui ont l'effet inverse des fonctions `fft2` et `fftshift`.
4. Tester le script principal sur les images `mandrill.png` et `lena.jpg`, en appelant la méthode `filtrePasseBasIdeal(n,m,fc)` avec différentes valeur de la fréquence de coupure f_c. Interpréter le résultat obtenu :
   (a) Quels sont les effets visuels lorsqu'on diminue f_c (donner 2 effets) ?
   (b) Donner deux applications de ce type de filtrage fréquentiel.

Exercice 2 — Convolution linéaire

1. Pour effectuer une convolution linéaire, on suppose que l'image est entourée de zéros. Étant donnée la taille d du filtre (on suppose d impaire), combien de colonnes et de lignes de zéros faut-il ajouter de part et d'autre de l'image ?
2. Écrire une fonction `imagePad` qui, étant donnés une image I et un filtre h, renvoie l'image entourée du cadre de zéros adéquat (voir question ci-dessus).
3. Écrire une fonction `convolution` qui, étant donnés un filtre h et une image I renvoie l'image filtrée I_f résultat de la convolution linéaire de h par I.
4. Tester cette fonction avec les filtres moyenneurs de taille 3×3, 5×5 et 7×7. Que constate-t-on ?

5. Visualiser la fonction de transfert des filtre moyenneurs de taille 3×3, 5×5 et $7\times$ (`imagesc` ou `mesh`).
   N.B. : On rappelle qu'on obtient la fonction de transfert d'un filtre par la méthode du *zero-padding*, visant à étendre la taille du filtre à celle de l'image, en complétant par des zéros (ici on prendra, par exemple, une taille 256×256), et à calculer le module centré de la transformée de Fourier de ce filtre agrandi.
6. Interprétation de la fonction de transfert visualisée : quelle est la formule analytique de la fonction de transfert d'un filtre moyenneur ? Ce filtre est-il un filtre passe-bas idéal ? Justifier.
7. **Bonus** : Reprendre les questions précédentes avec un filtre Gaussien de taille 3×3, 5×5 et 7×7.

Exercice 3 — Filtrage anti-aliasing

Dans cette partie, on utilisera l'image `barbara.png` fournie. On souhaite mettre en place un script `FiltrageAntiAlisasing` qui va sous-échantilloner une image sans introduire d'effets d'Aliasing.

1. Dans le script, effectuer un sous-échantillonnage de l'image courante d'un facteur 2 (utiliser pour cela la fonction `subSampling2()` fournie), de manière récursive (voir le TME 3).
2. Avant de sous échantillonner l'image, appliquer à l'image initiale un filtrage passe-bas adéquat (utiliser pour cela la fonction `antiAliasingFilter(I)` fournie).
   — Rappeler le principe de ce filtrage dans le domaine fréquentiel.
3. Décrire et interpréter l'image résultante après filtrage et sous-échantillonnage. Y a-t-il eu perte d'information entre l'image résultante et l'image initiale ? Compararer qualitativement les dégradations par rapport au sous-échantillonnage brutal (sans filtrage).
4. Bonus : renouveler les expérimentations sur l'image `mandrill.png` fournie.

Exercice 4 — Eclatement d'une image couleur

Pour cet exercice, on utilisera les images `CLOWN` et `CLOWN_LUMI`.

1. Créer deux variables I_1 et I_2 dans lesquelles on chargera respectivement les images `CLOWN` et `CLOWN_LUMI`. Visualiser ces deux images, observer leur type et la taille des données associées, quelles différences observe-t-on ?
2. I_1 est un tableau à trois dimensions. Préciser, en le justifiant, à quoi correspond chacune de ces dimensions. Créer 3 nouvelles images I_R, I_V, et I_B dans lesquelles on stockera respectivement la composante rouge, la composante verte et la composante bleue de l'image `CLOWN`. Visualiser ces 3 plans et expliquer cette le résultat de cette visualisation.
3. Créer une nouvelle image I_3 en mélangeant les plans de couleur (essayer plusieurs mélanges).
4. Proposer une solution pour voir le plan rouge en rouge, le plan bleu en bleu et le plan vert en vert. On appellera respectivement R, V et B les images obtenues.

Semaine 5

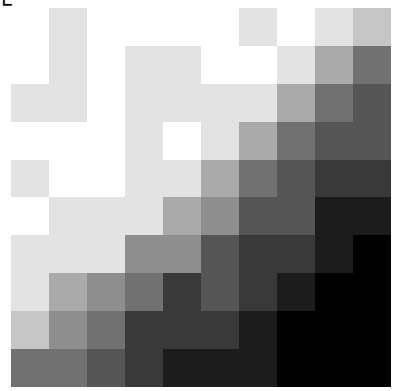
Détection de contours

5.1 TD 5 : filtrage et détection de contours

Exercice 1 — Filtre détecteur de contours de Sobel

On considère le tableau I suivant :

$$I = \begin{bmatrix} 9 & 8 & 9 & 9 & 9 & 9 & 8 & 9 & 8 & 7 \\ 9 & 8 & 9 & 8 & 8 & 9 & 9 & 8 & 6 & 4 \\ 8 & 8 & 9 & 8 & 8 & 8 & 8 & 6 & 4 & 3 \\ 9 & 9 & 9 & 8 & 9 & 8 & 6 & 4 & 3 & 3 \\ 8 & 9 & 9 & 8 & 8 & 6 & 4 & 3 & 2 & 2 \\ 9 & 8 & 8 & 8 & 6 & 5 & 3 & 3 & 1 & 1 \\ 8 & 8 & 8 & 5 & 5 & 3 & 2 & 2 & 1 & 0 \\ 8 & 6 & 5 & 4 & 2 & 3 & 2 & 1 & 0 & 0 \\ 7 & 5 & 4 & 2 & 2 & 2 & 1 & 0 & 0 & 0 \\ 4 & 4 & 3 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$ correspondant à l'image :



On souhaite appliquer à cette image les filtres détecteurs de contours de Sobel suivants, dont les réponses impulsionnelles sont les suivantes :

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \text{ et } S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

1. Compléter l'application du filtre horizontal S_x donnant $I_x = I \star S_x$ suivant :

$$I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 3 & -1 & 1 & -2 & -1 & -3 & -10 & -5 & 0 \\ 0 & -3 & 1 & 1 & \mathbf{?} & 2 & 1 & -4 & -10 & -13 & \mathbf{?} & 0 \\ 0 & -1 & 2 & -1 & -3 & 1 & -2 & \mathbf{?} & -14 & -11 & -4 & 0 \\ 0 & 1 & 2 & -3 & -2 & -2 & -10 & \mathbf{?} & -12 & -6 & -1 & 0 \\ 0 & 1 & 1 & -3 & -4 & -7 & \mathbf{?} & -12 & -9 & -5 & 0 & 0 \\ 0 & -1 & -1 & -4 & -8 & \mathbf{?} & -13 & -8 & -7 & -7 & -1 & 0 \\ 0 & -3 & \mathbf{?} & -8 & -11 & -8 & -9 & -6 & -6 & -7 & -2 & 0 \\ 0 & -6 & \mathbf{?} & -10 & -11 & -4 & -4 & -7 & -6 & -4 & -1 & 0 \\ 0 & -6 & \mathbf{?} & -10 & -9 & -2 & -2 & -7 & -5 & -1 & 0 & 0 \\ 0 & \mathbf{?} & -6 & -9 & -8 & -3 & -1 & -5 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

N.B : On suppose ici qu'on effectue un padding par recopie sur l'image I.

2. Compléter l'application du filtre vertical S_y donnant $I_y = I \star S_y$ suivant :

$$I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -3 & -3 & 0 & 1 & -3 & -8 & -11 & 0 \\ 0 & -3 & -1 & -1 & \mathbf{?} & -4 & -3 & -4 & -10 & -15 & \mathbf{?} & 0 \\ 0 & 1 & 2 & 1 & 1 & 1 & -4 & \mathbf{?} & -14 & -11 & -6 & 0 \\ 0 & 1 & 2 & 1 & 0 & -2 & -8 & \mathbf{?} & -12 & -8 & -5 & 0 \\ 0 & -1 & -3 & -3 & -4 & -9 & \mathbf{?} & -10 & -7 & -7 & -8 & 0 \\ 0 & -1 & -3 & -6 & -10 & \mathbf{?} & -11 & -8 & -5 & -5 & -7 & 0 \\ 0 & -5 & \mathbf{?} & -12 & -15 & -14 & -9 & -6 & -6 & -5 & -4 & 0 \\ 0 & -6 & \mathbf{?} & -14 & -13 & -10 & -6 & -5 & -6 & -4 & -1 & 0 \\ 0 & -14 & \mathbf{?} & -8 & -7 & -6 & -6 & -5 & -3 & -1 & 0 & 0 \\ 0 & \mathbf{?} & -6 & -3 & -2 & -3 & -3 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3. Compléter le calcul de la norme du gradient $||\overrightarrow{G}|| = \sqrt{I_x^2 + I_y^2}$ (arrondi à l'entier le plus proche) :

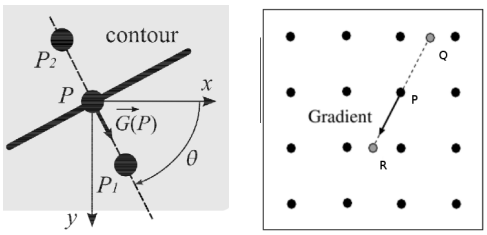
$$||\overrightarrow{G}|| = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 3 & 3 & 3 & 2 & 1 & 4 & 13 & 12 & 0 \\ 0 & 4 & 1 & 1 & \mathbf{?} & 4 & 3 & 6 & 14 & 20 & \mathbf{?} & 0 \\ 0 & 1 & 3 & 1 & 3 & 1 & 4 & \mathbf{?} & 20 & 16 & 7 & 0 \\ 0 & 1 & 3 & 3 & 2 & 3 & 13 & \mathbf{?} & 17 & 10 & 5 & 0 \\ 0 & 1 & 3 & 4 & 6 & 11 & \mathbf{?} & 16 & 11 & 9 & 8 & 0 \\ 0 & 1 & 3 & 7 & 13 & \mathbf{?} & 17 & 11 & 9 & 9 & 7 & 0 \\ 0 & 6 & \mathbf{?} & 14 & 19 & 16 & 13 & 8 & 8 & 9 & 4 & 0 \\ 0 & 8 & \mathbf{?} & 17 & 17 & 11 & 7 & 9 & 8 & 6 & 1 & 0 \\ 0 & 15 & \mathbf{?} & 13 & 11 & 6 & 6 & 9 & 6 & 1 & 0 & 0 \\ 0 & \mathbf{?} & 8 & 9 & 8 & 4 & 3 & 5 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4. Proposer un seuil de binarisation pour extraire le contour de l'image à partir de $||\overrightarrow{G}||$.

Exercice 2 — Suppression de non-maxima

Le problème de détection de contours se formule comme une recherche de maxima locaux du module du gradient dans la direction du gradient. Après application d'un filtre discret approximant le gradient, une étape de post-traitement couramment utilisée consiste à effectuer une suppression de non maxima.

On suppose connu le vecteur gradient $\overrightarrow{G(P)}$ en un point P (pixel de l'image). On veut déterminer si P est un maximal local de $||\overrightarrow{G(P)}||$ dans la direction du gradient (figure (a)).



(a) (b)

On va considérer les deux algorithmes suivants pour effectuer cette tâche :

1. Discrétisation de l'orientation. Proposer un algorithme permettant de déterminer la direction discrète à assigner au point P (modèle à 8 voisins).
2. Interpolation sub-pixel. On propose d'estimer la valeur du module du gradient pour les deux voisins dans la direction du gradient.
   Indication : On déterminera les coordonnées des points Q et R (figure (b)), et on estimera le module du gradient en ces points par interpolation linéaire.

Exercice 3 — Pyramide multi-résolution et crénelage (*aliasing*)

Dans le cadre de la construction d'une pyramide multi-résolution construite avec filtrage gaussien, on souhaite déterminer les paramètres du filtre de sorte à ne pas introduire d'effets d'*aliasing* lors du sous-échantillonnage. On supposera que l'image de résolution maximale aura été correctement échantillonnée.

Rappel : La TF d'une gaussienne est une gaussienne d'écart-type inverse ($\sigma_f = \frac{1}{\sigma_s \pi}$) :

$$TF\left[e^{-b^2(t^2+u^2)}\right] = \frac{\pi}{b^2} e^{-\frac{\pi^2(f^2+g^2)}{b^2}}$$

1. En utilisant un filtre gaussien, est-il théoriquement possible d'annuler totalement l'effet de l'*aliasing* ? Expliquer.
2. En supposant que le filtre gaussien a en pratique une fréquence de coupure de 3σ, déterminer l'écart-type du masque gaussien à appliquer dans le domaine spatial, de sorte que la convolution $I \star G$ n'introduise pas de repliement spectral lors du passage au niveau de résolution suivant.

5.2 TME 5 : comparaison de filtres détecteurs de contours

Dans ce TP, on demande d'effectuer une expérimentation de différentes méthodes de détection de contours, en évaluant les points suivants :

1. Comparaison de filtres discret du premier et du second ordre.
2. Influence du lissage comme pré-traitement à la détection de contours.
3. Suppression de non-maxima et évaluation de la chaine complète en terme de robustesse et localisation.

Exercice 1 — Comparaison de filtres discret du 1^{er} et du 2^{nd} ordre

Récupérer la fonction `convolution(I,m)` qui effectue le produit de convolution discret 2d entre I et m, où m est le symétrique de la réponse impulsionnelle du filtre.

1. **Filtre de Sobel**. On rappelle que le filtre de Sobel est un filtre discret permettant d'approcher le gradient par différences finies avec les masques directionnels suivants :
   $G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ et $G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
   On demande ici :
   (a) de mettre en place les fonctions pour calculer $I_x = I \star G_x$ et $I_y = I \star G_y$;
   (b) de mettre en place les fonctions pour calculer le module du gradient ainsi estimé ;
   (c) de mettre en place la fonction permettant de seuiller ce module afin de produire un filtre détecteur de contours du premier ordre.
2. **Filtre laplacien**. On rappelle que le filtre laplacien est un filtre discret permettant d'approcher les dérivées secondes de l'image par différences finies avec le masque suivant :
   $L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$.
   On demande ici :
   (a) de mettre en place la fonction pour calculer $I_L = I \star L$.
   (b) de mettre en place la fonction pour déterminer les passages par 0 de I_L. Pour cela, on procèdera comme suit :
   — Prendre une fenêtre centrée 3×3 sur le pixel (i, j), et calculer $max(I_L)$ et $min(I_L)$.
   — Le passage par O sera détecté si $max(I_L) > 0$, $min(I_L) < 0$ et $max(I_L) - min(I_L) > S$.
3. **Comparaison des filtres du premier et du second ordre**.
   Évaluer les deux types de détecteurs de contour sur l'image `lena.gif`.
   On trouvera expérimentalement un seuil de binarisation permettant de détecter senseiblement les mêmes contours avec les filtres du premier et du second ordre (*e.g.* seuils de 70 et 70). Quelle est la différence principale entre les deux détecteurs ?

Exercice 2 — Suppression de non maxima

On va maintenant s'intéresser au problème de la suppression de non maxima, pour les filtres du premier ordre. Afin de détecter les maxima locaux du module du gradient dans la direction du gradient, on utilisera la fonction `orientation(Ix,Iy,IG)` fournie, qui à partir de l'image du module du gradient I_G et des dérivés directionnelles I_x et I_y retourne l'image de l'orientation des gradients. L'image est discrétisée en 4 directions 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3\pi}{4}$, codées respectivement par 1,2,3,4 (0 encodant un gradient nul en ce pixel). On demande ici :

1. De visualiser en couleur le résultat de la fonction `orientation(Ix,Iy,IG)` sur l'image `tools.gif`. Interpréter le résultat obtenu et vérifier les voisins à inspecter en fonction de l'orientation en chaque pixel.
2. De mettre au point une fonction `nms(Ig,Ior)`, qui à partir de l'image du module du gradient I_g et de son orientation I_{or}, renvoie une image du module du gradient où seuls les extrema locaux sont conservés. Quelle est l'épaisseur des contours qui seront ensuite détectés ?
3. De tester la fonction `nms` aux images `tools.gif` et `lena.gif`, après application ou non d'un filtre gaussien.
4. Conclure sur l'influence du lissage sur la chaine complète de détection de contours (lissage + dérivation + `NMS`). Montrer les différents niveaux de granularité des structures qui sont détectées en fonction de σ.

Exercice 3 — Influence du lissage dans la détection de contours

Les détecteurs de contours précédents sont des filtres fréquentiels passe-haut, ils ont donc tendance à accentuer le bruit. Pour surmonter ce problème, on propose d'effectuer un lissage gaussien préalable à la détection de contours. Vous pourrez utiliser la fonction `gauss(sigma)` fournie, qui génère un masque gaussien de taille adaptée à l'écart-type σ (exprimé en pixels).

1. Effectuer un lissage gaussien de l'image en calculant le produit de convolution suivant $I \star G_\sigma$.
2. Effectuer une détection de contours en appliquant les filtres de Sobel et laplacien à l'image lissée.
3. Évaluer les deux types de détecteurs de contour après lissage sur l'image `lena.gif` avec un lissage de $\sigma = 2$. On trouvera expérimentalement un seuil de binarisation permettant de détecter sensiblement les mêmes contours avec les filtres du premier et du second ordre (*e.g.* seuils de 10 et 200 pour le Laplacien et le module du filtre de Sobel, respectivement). Quelle est la différence principale entre les deux types de détecteurs appliqués après lissage ?
4. Bonus : Faire varier σ sur un intervalle $[\sigma_1, \sigma_2]$ (à déterminer par rapport à la taille du masque correspondant) et analyser les résultats obtenus après application des deux filtres détecteurs de contour. Quelle est l'influence du lissage sur le bruit ? Quelle est l'influence du lissage en terme de localisation des contours ? Illustrer.
5. Bonus : Appliquer comme premier traitement un filtre réhausseur de contraste à la place du filtre gaussien. Quelle en est la conséquence ?
6. Bonus : Calculer la décomposition multi-résolution de l'image de départ (pyramide - jusqu'à une taille 8×8), et appliquer les filtres détecteurs de contours à chaque niveau de résolution. Que constatez-vous ?