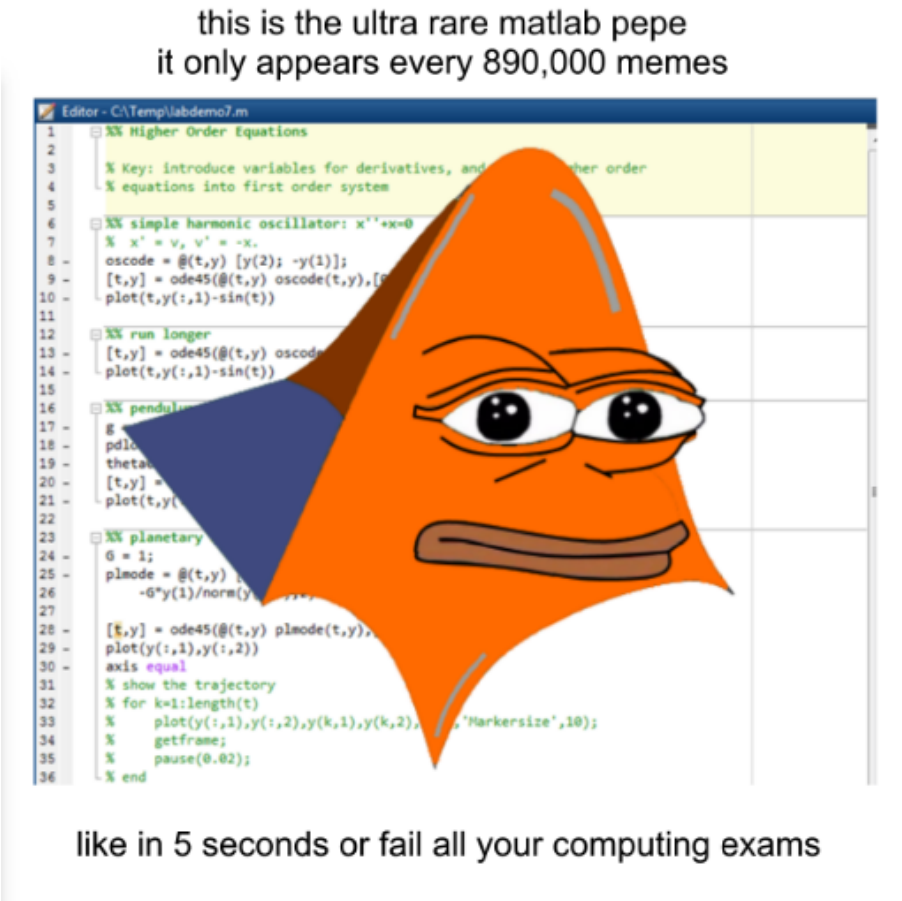
EPHEC

TRAITEMENT DE SIGNAL PRATIQUE

Le traitement de signal pour les nuls



January 4, 2019

Contents

Accès au code et fichiers

https://we.tl/t-UwuvdxfcsO

1 Traitement du son

1.1 Dualité temps/fréquence

Rappel 1:

Pour rappel, le dualité temps/fréquence désigne le fait que l'on ne peut pas être précis dans les deux domaines en même temps. Si nous voulons une précision temporelle, nous aurons une imprécision fréquentielle (**étalement de spectre**) et si nous voulons une précision fréquentielle, nous aurons une imprécision temporelle (**longue durée**).

1.1.1 Création du signal

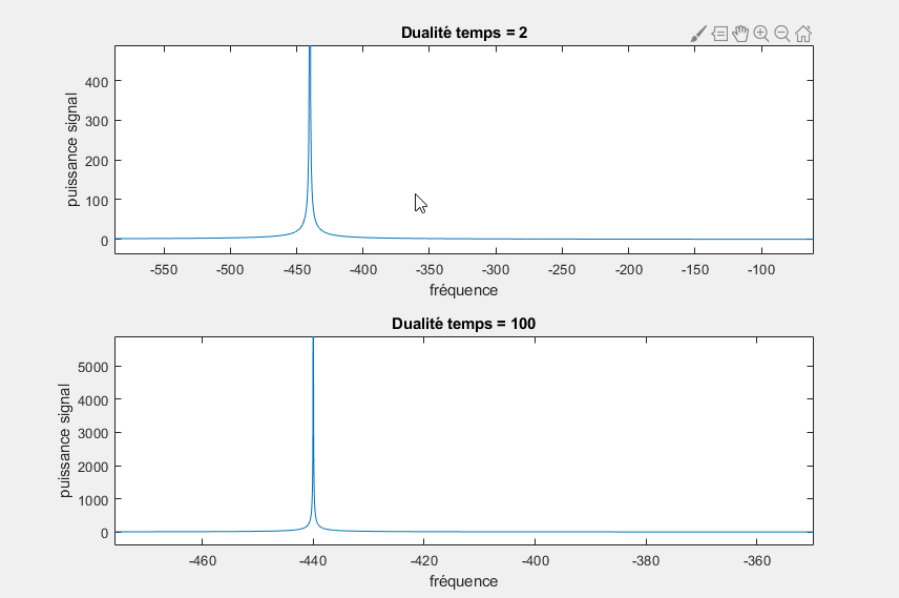
```
% Fs: Frequence Sample
% On applique Shannon—Nyquist, donc Fs ≥ 2*f
Fs = 2000;
% p1,p2 : periodes
% Temps en seconde.
p1 = 2;
p2 = 100;
% t1, t2
% Vecteur de temps
% Fenetre d'echantillonage
% Commence a 0, avec un echantillonage de 1/Fs, jusqu'a la periode(* Fs).
t1=0:1/Fs:p1;
t2=0:1/Fs:p2;
% s1, s2 : calcul du signal
% on applique la formule : A*sinus(2*pi*f*t)
% A*sin(w*t) ou w=(2*pi*f) (par rapport a math)
s1=sin(2*pi*440*t1);
s2=sin(2*pi*440*t2);
```

1.1.2 Démonstration avec sptool

Il suffit de créer le spectre avec SPTool et de l'importer et faire un plot pour afficher la dualité.

1.1.3 Démonstration sans sptool

```
% y1, y2 : transformee de fourier rapide
% fft : effectue la transformee
% fftsfift : centre les valeurs au niveau de 0.
y1=fftshift(fft(s1));
y2=fftshift(fft(s2));
% f1, f2 : frequence
% permet de representer les graphiques
f1=-Fs/2:1/p1:Fs/2;
f2=-Fs/2:1/p2:Fs/2;
% representation graphique
% subplot permet d'afficher plusieurs graphiques
subplot(2,1,1);
plot(f1,abs(y1));
title('Dualite temps = 2')
xlabel('frequence');
ylabel('puissance signal');
subplot(2,1,2);
plot(f2,abs(y2));
title('Dualite temps = 100')
xlabel('frequence');
ylabel('puissance signal');
```



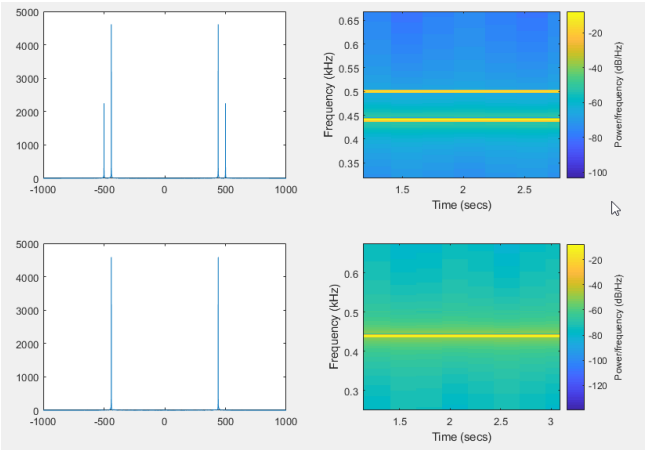
1.2 Création d'un signal et suppression d'un bruit avec filtre passe-bas

1.2.1 Code de création du signal

```
% f1, f2 : frequences
% signal a 440hz
% bruit a 500hz
f1=440;
f2=500;
% Fs : frequence echanti
% Shannon : 2*f =< Fs
Fs=2000;
% A1, A2 : Amplitude
A1=1;
A2=0.5;
% p : periode
% duree
p=5;
% t1
% Vecteur de temps
t=0:1/Fs:p;
% s1, s2 : signaux
% creation des signals
s1= A1*sin(2*pi*f1*t);
s2= A2*sin(2*pi*f2*t);
% sf : signal final
% addition
sf = s1+s2;
```

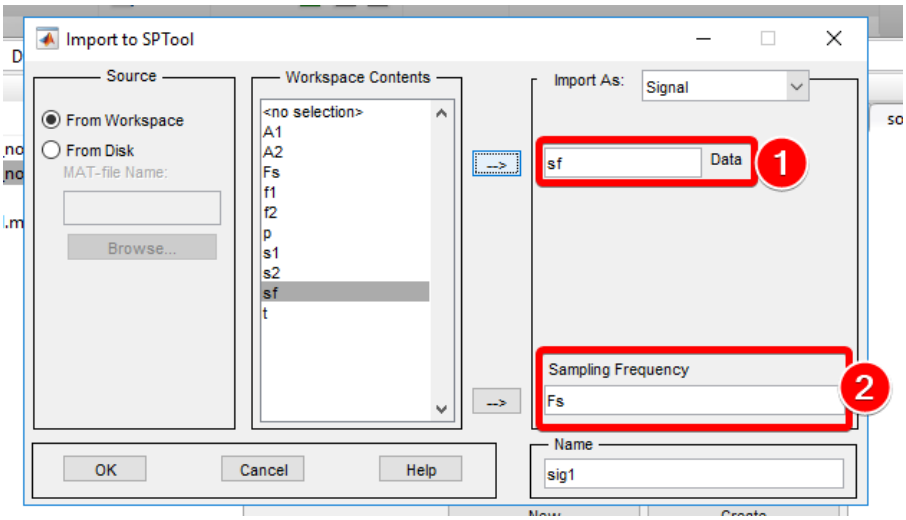
1.2.2 Démonstration sans sptool

```
% Fc : Frequence coupure
% > 440 Hz car 500 Hz = freq non voulue
Fc=460;
% Wn : Fenetre de "coupe"
% souvent fc/(fs/2)
% depend si low, high ou stop
Wn=Fc/(Fs/2);
% b,a : coeficients du filtre
% butter : filtre butterworth
% (n,Wn,type) : ordre, fenetre, [low,high,stop]
[b,a]=butter(50,Wn,'low');
sfiltred=filter(b,a,sf);
```

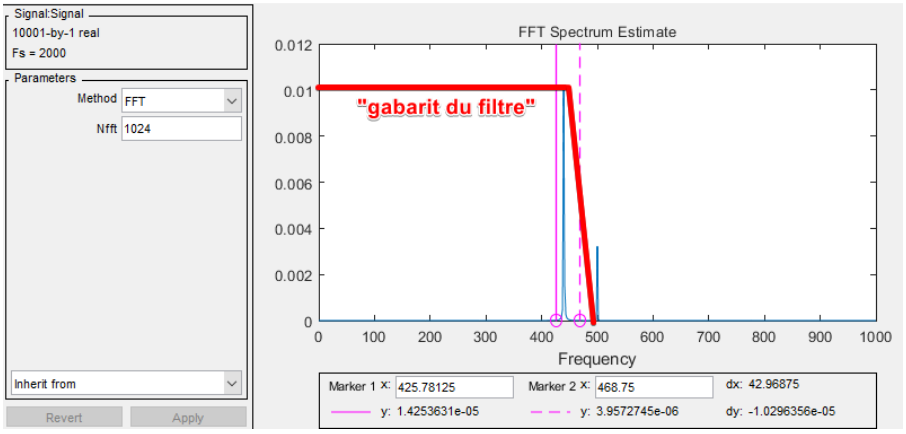


1.2.3 Démonstration avec sptool

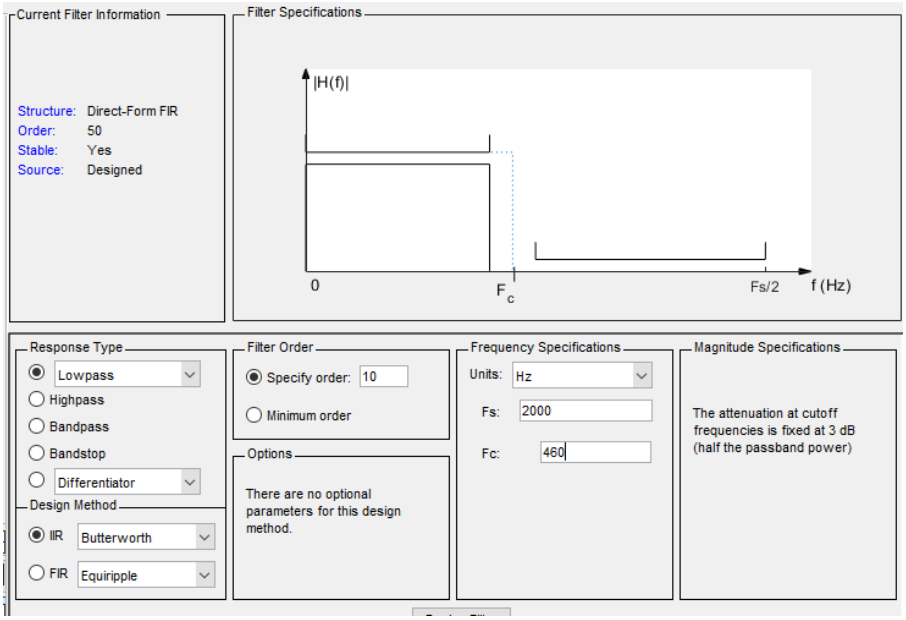
1. Importer la "data" **sf** qui est notre signal créé et la fréquence d'échantillonage **Fs**



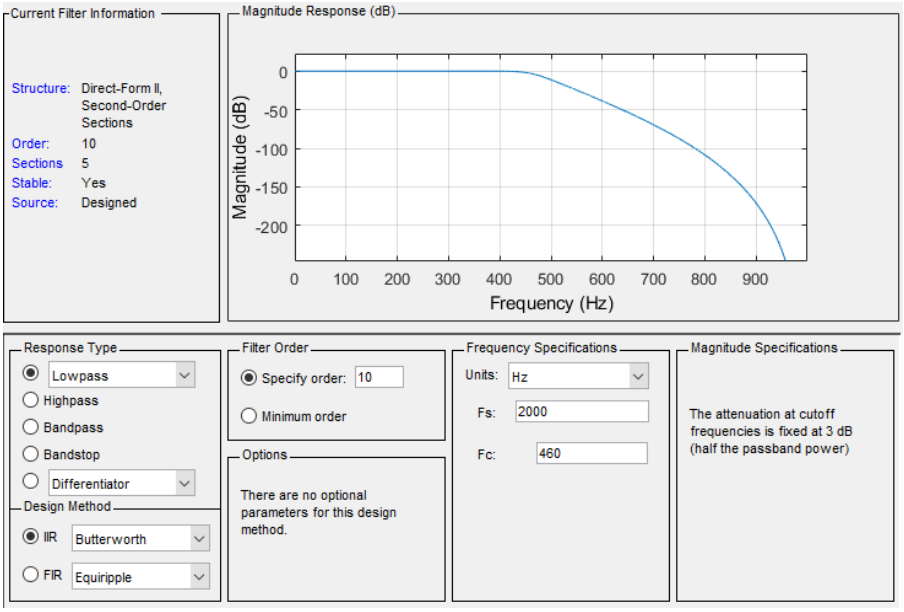
2. création du spectre "**fft**" et représentation du gabarit pour la pression de la fréquence bruitée.



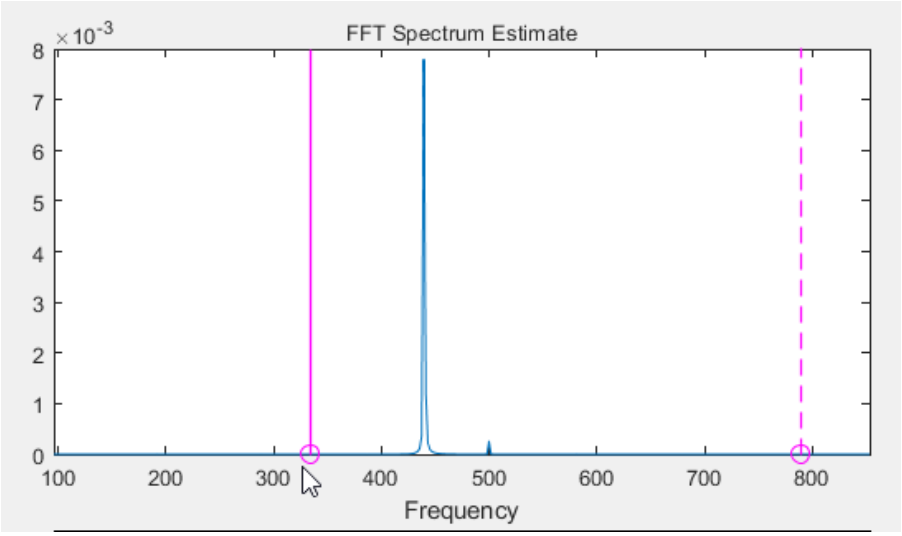
3. Création du filtre **"passe-bas" butteworth** d'**ordre** 10 avec une fréquence de coupue supérieur à la fréquence gardée.



4. Gabarit du filtre réel



5. Représentation de dirac du signal filtré. On observe que la "pic" à 500 a bien été atténuée.



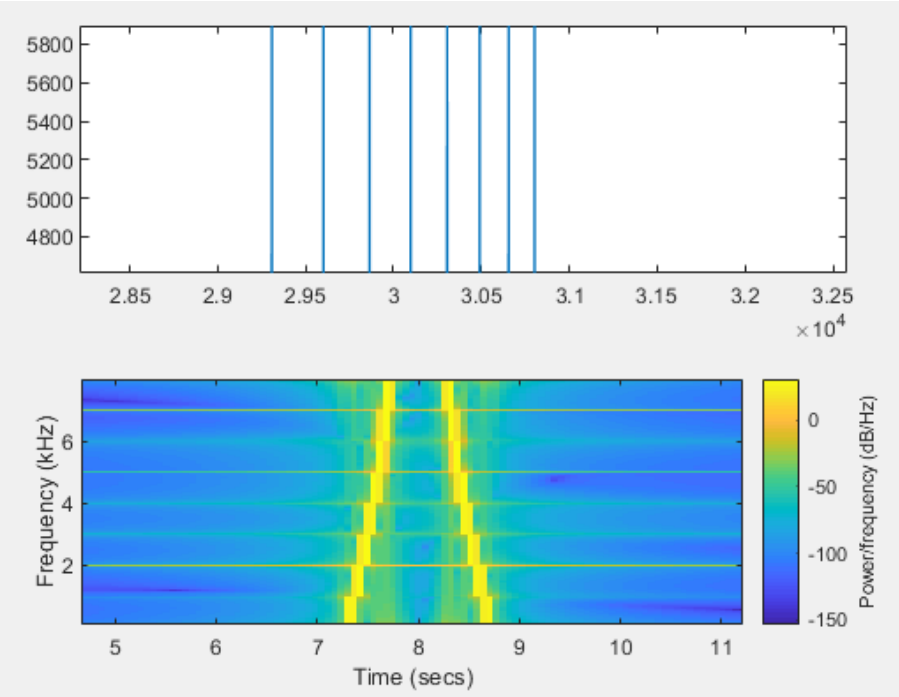
1.3 Augmentation d'une octave

```
% f : freq
f = 150;
% Fs : freq sample
Fs = 8000;
% p : periode
p = 2;
% t :
t = 0:1/Fs:p;
```

1.3.1 Boucle et affichage

```
% y : array
y = [];
for i=1 : 8
w = 2*pi*f; % omega = 2*pi*f
x = sin(w*t); % calcul du sinus = A*sin(wt)
f = f*2^(1/6); % passer a l'octave suivante
y = [y,x]; % concatenation en 1 ligne
end

% spectre FFT
f = fftshift(fft(y));
x = abs(f);
dt = 1/p;
tf = 0:dt:(Fs*8)+7*dt; % 0:dt:((Fs*echantillons) + ((echantillon-1)*dt)))

% affichage plot et spectogram
subplot(2,1,1)
plot(tf,x);
subplot(2,1,2)
spectrogram(f,1024,[],[],Fs,'yaxis');
```

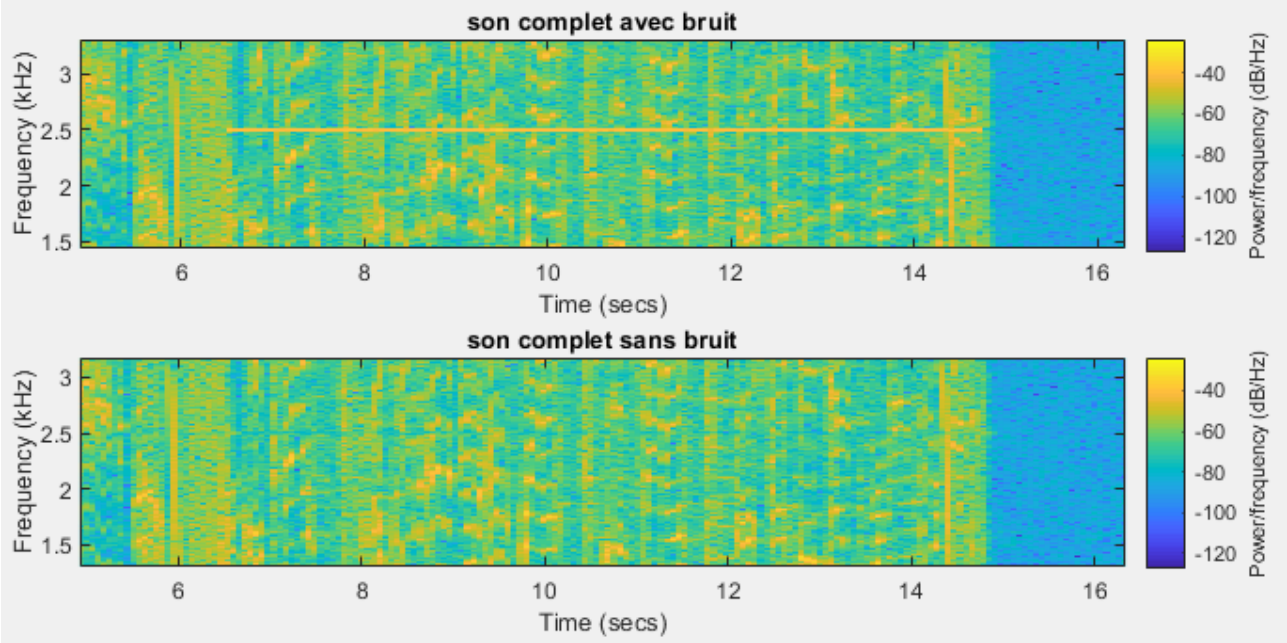
1.3.2 Résultat



1.4 Suppression de bruit (alien) - filtre stopband

1.4.1 Code

```
[son,Fs] = audioread('alien.wav');
% spectogram pour observer le bruit
% bruit_start at 6.5 & end at 14.8
subplot(2,1,1);
spectrogram(son,1024,[],[],Fs,'yaxis');
title('son complet avec bruit');
% decoupe
% p1, p2 : partie 1 et partie 2
p1 = 6*Fs;
p2 = 15*Fs;
bruit = son(p1:p2);
% Fc : freq coupure
Fc=2500;
% Fenetre
% Freq de coupure +/-
Wn1=(Fc-10)/(Fs/2);
Wn2=(Fc+10)/(Fs/2);
% matrice avec les 2 fenetres
Wn = [Wn1 Wn2];
% creation d'un butterworth d'ordre 5
% filtre stopband
[b,a]=butter(5,Wn,'stop');
sans_bruit=filter(b,a,bruit);
% on remplace les partie bruite avec les partie non bruite
son(p1:p2)=sans_bruit;
subplot(2,1,2);
spectrogram(son,1024,[],[],Fs,'yaxis');
title('son complet sans bruit');
```

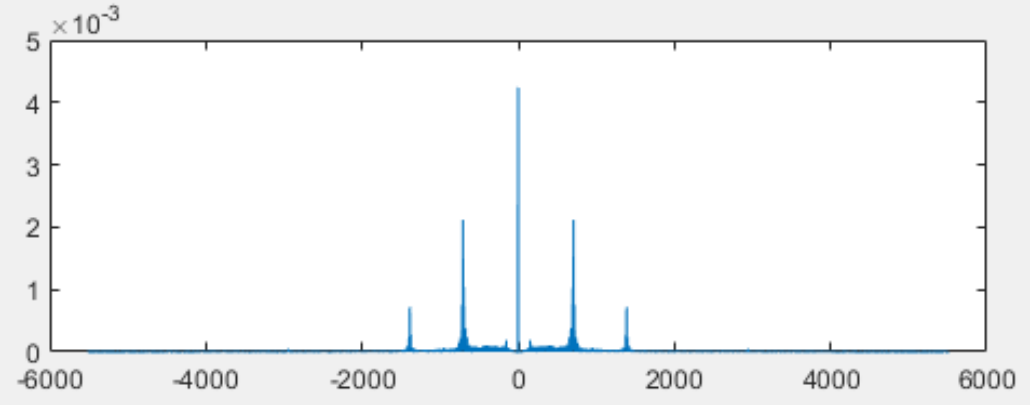


1.5 Suppression bruit (MORSE) - avec affichage fft

1.5.1 Code

```
[s,Fs] = audioread('MORSE2.WAV');
y = fftshift((1/length(s))*fft(s));
f = -Fs/2:(Fs/length(s)):Fs/2-(Fs/length(s));
subplot(2,1,1);
title('representation spectrale')
plot(f,abs(y));
```

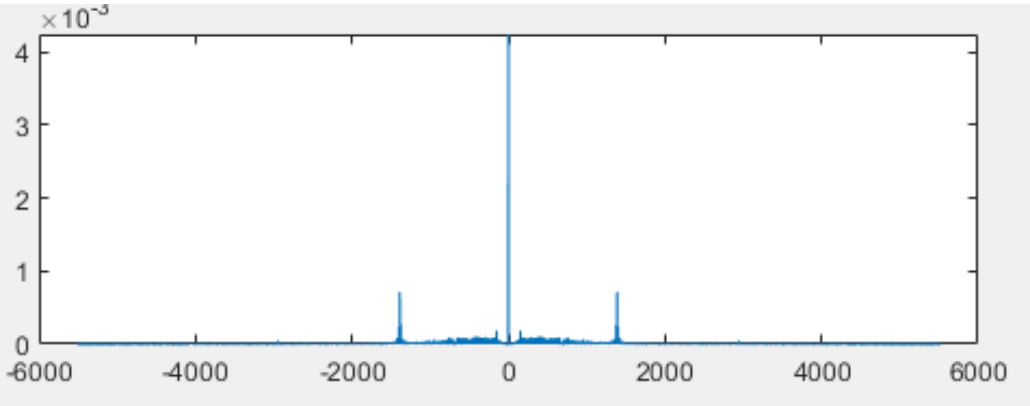
1.5.2 plot - affichage bruit



1.5.3 filtre stopband

```
Fc = 700;
Wn1 = (Fc-67)/(Fs/2);
Wn2 = (Fc+67)/(Fs/2);
Wn = [Wn1 Wn2];
[b,a] = butter(2,Wn,'stop');
Morse = filter(b,a,s);
```

1.5.4 plot - affichage sans bruit



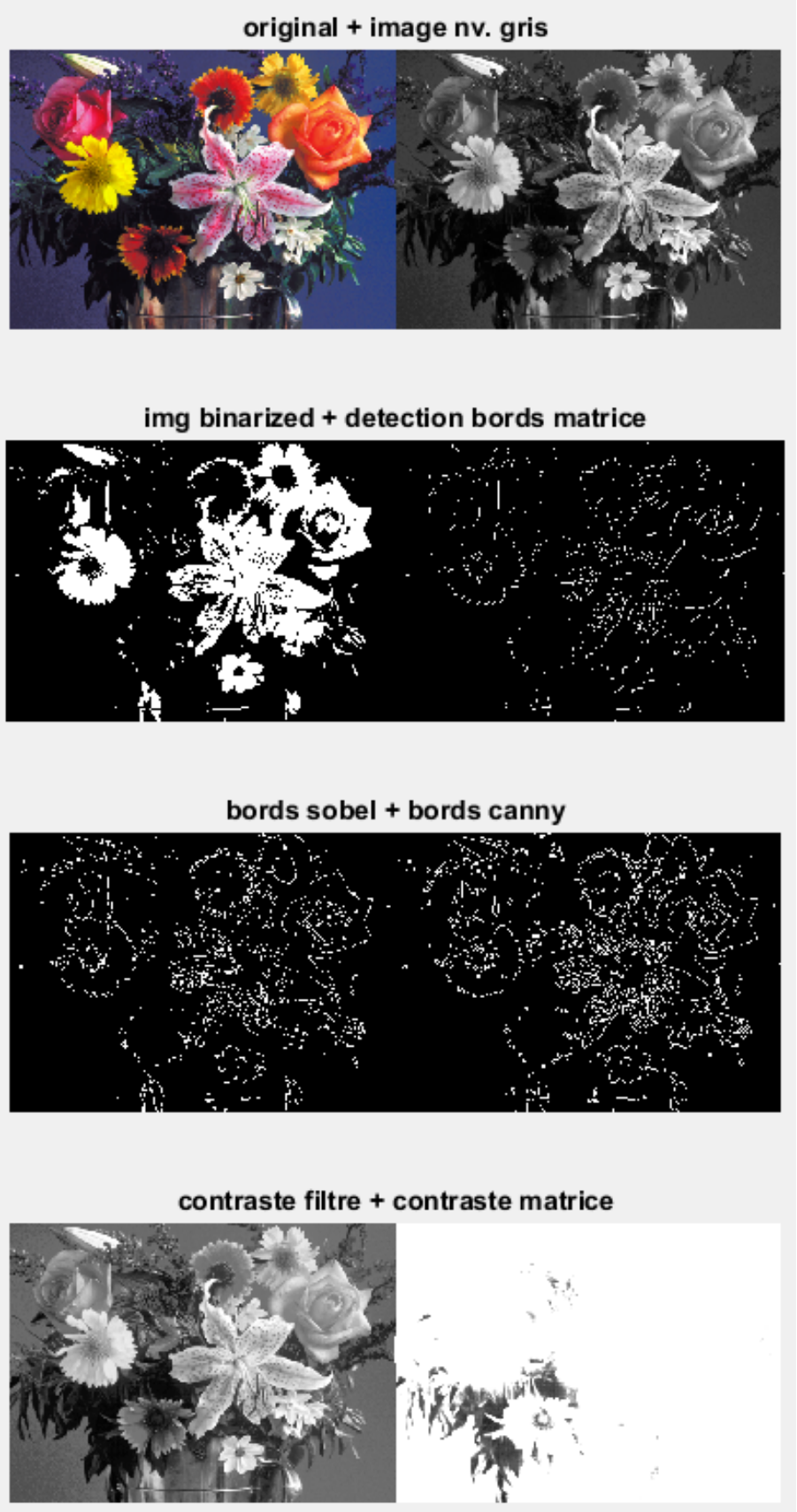
2 Traitement d'image

2.1 Détection de bords

2.1.1 Code

```
img = imread('assets/flowers.tif');
% conversion image en gris
% rgb2gray
img_gray = rgb2gray(img);
% detection bords
% imbinarerd : binarisation de l'image
img_bin = imbinarize(img_gray);
% binarisation solution 2
% methode depreciated (im2bw)
level = graythresh(img_gray);
img_bin2 = im2bw(img_gray,level);
% detection des bords
% edge : permet de trouver les bords sur par rapport a l'intensite d'une
% image
% sobel, canny
% detection avec matrice
F = [0 0 0; 1 0 0; 0 -1 0]; % detection de bords
img_edged_sobel = edge(img_bin,'sobel');
img_edged_canny = edge(img_bin,'canny');
img_edged_matrice = imfilter(img_bin,F);
% amelioration du contraste
% imadjust (img, height, width, intensity)
img_contrast = imadjust(img_gray,[],[],0.5);
% contraste avec matrice
C = [1 1 1; 1 4 1; 1 1 1]; % contraste hardcore
img_contrast_matrice = imfilter(img_gray,C);
```

2.1.2 Affichage

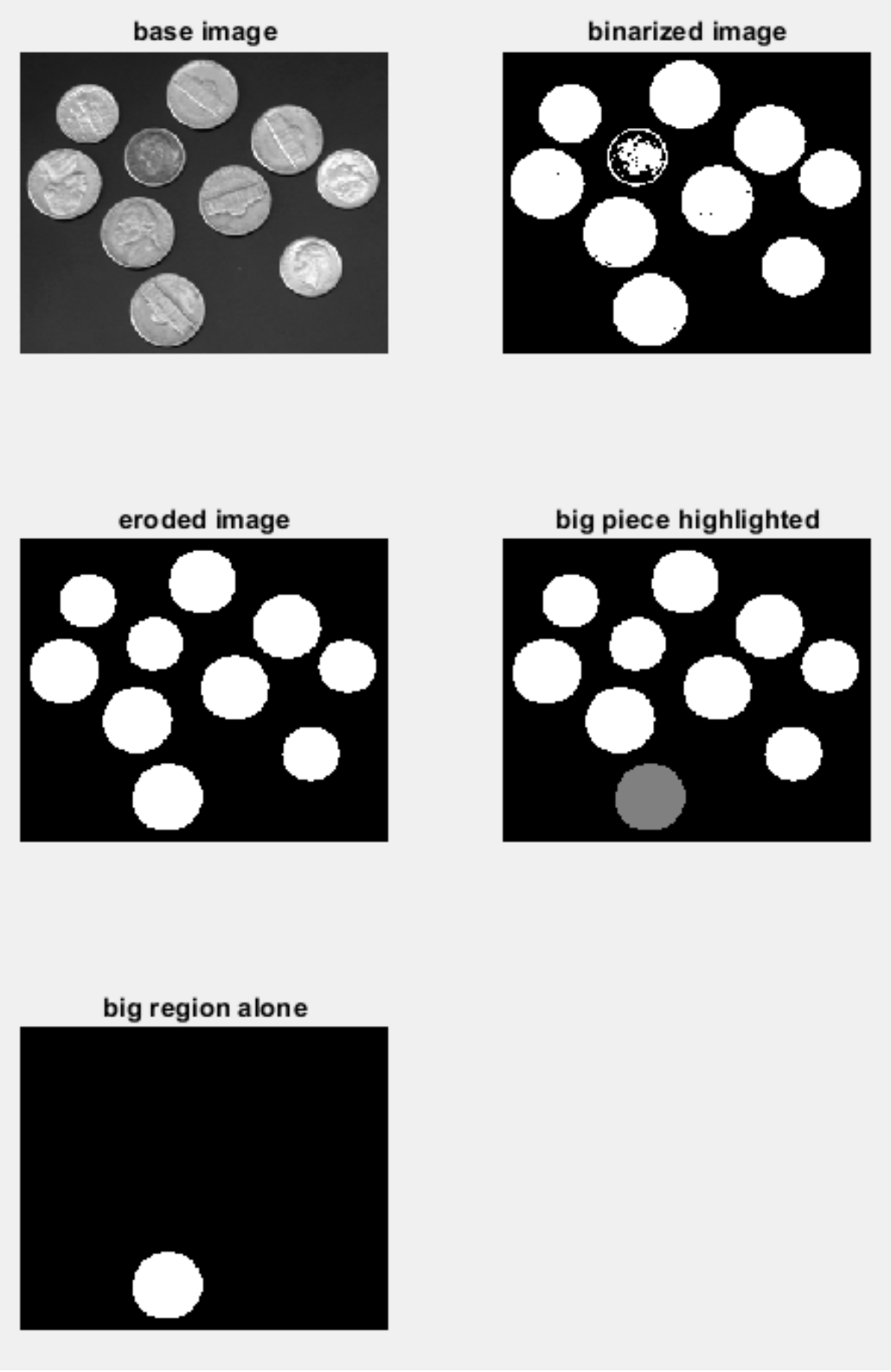


2.2 Labélisation et détection de pièces

2.2.1 Code

```
% imread : import image
img = imread('assets/pieces.png');
% [X, map] = imread('assets/pieces.png');
% img = ind2gray(X,map);
% level : graythresh
% designe un seuil entre 0 et 1
% permet de binariser une image
 level = graythresh(img);
% im2bw (depreciated)
% converti en image binaire via le graythresh
% base sur img(base) et level(graythresh)
img_bin = im2bw(img,level);
% imbinarize should be used instead of im2bw
% img_bin = imbinarize(img);
% remplir les zones vides
img_fill = imfill(img_bin,'holes');
img_fill = bwareaopen(img_fill,30);
% erode
se = strel('disk',2,4);
img_se = imerode(img_fill, se);
% labelisation
% avec im2bw
% L : regions
% n : nombre (valeur)
[L, n] = bwlabel(img_se);
disp('nombre de pieces:');
disp(n);
% regionprops : mesure la propriete des regions labelisee.
labels = regionprops(L);
% boucle pour trouver la surface la plus grande.
maxArea = 0;
for i=1:size(labels)
    if maxArea < labels(i).Area
        maxArea = labels(i).Area;
        index=i;
    end
end
% recupere le nombre de labels
% pieces : calcule le nombre de regions
pieces = L;
% @big : plus grosse region
% find()
% recupere l'index le plus grand
big = find(L == index);
for i=1:size(big)
    pieces(big(i))=0.5;
end
% bwareaopen : nettoyage
% afficher la piece avec la plus grande region
big_piece_selected = bwareaopen(pieces, labels(index).Area);

% affichage console % disp()
% ------------------------------------------------------------
% nombre de pieces:
%    10
% ------------------------------------------------------------
```

2.2.2 Affichage



2.3 Filtre moyenneur

2.3.1 Via matrice

```
img = imread('assets/LENNA.BMP');

% M1 : Passe-haut
M1 = [1 1 1; 1 5 1; 1 1 1];
% M2 : Passe-bas - img bruitee
M2 = [0 -1 0; -1 5 -1; 0 -1 0];
% M3 : Contraste +
M3 = [0 -1 0; -1 5 1; 0 -1 0];
% M4 : flou
% filtre moyenneur
% valeur elevee = flou eleve
M4 = ones(3,3)/9;
% M5 : detection de bords
M5 = [-1 0 1; -1 0 1; 0 1 0; 1 -5 1; 0 1 0];
% M6 : renforcement bords
M6 = [ 0 0 0; -1 1 0; 0 0 0 ];
% M7 : repoussage pixel
M7 = [ -2 -1 0; -1 1 1; 0 1 2];

% applications des filtres
img_M1 = imfilter(img,M1);
img_M2 = imfilter(img,M2);
img_M3 = imfilter(img,M3);
img_M4 = imfilter(img,M4);
img_M5 = imfilter(img,M5);
img_M6 = imfilter(img,M6);
img_M7 = imfilter(img,M7);
```

original, passe-haut

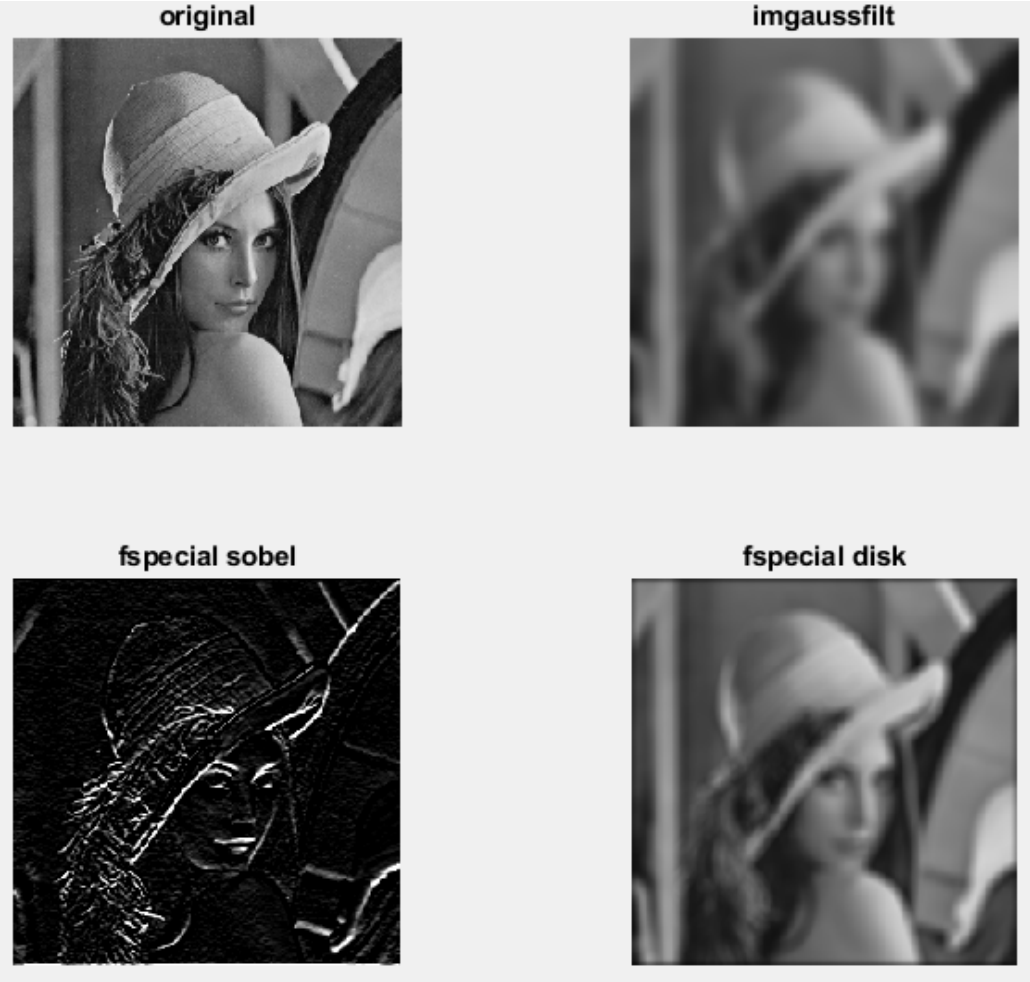
passe-bas, Contraste +

flou, detection bords

renforcement bords, repoussage

2.3.2 Via fspecial

```
img = imread('assets/LENNA.BMP');
% imgaussfilt: flou gaussien
% img, sigma
img_blur = imgaussfilt(img,5);
subplot(1,3,1);
imshow(img_blur);
% fspecial
% sobel : countour
S = fspecial('sobel');
img_sobel = imfilter(img,S);
subplot(1,3,2);
imshow(img_sobel);
% disc : flou moyenneur 'disc'
D = fspecial('disk',5);
img_disk = imfilter(img,D);
subplot(1,3,3);
imshow(img_disk);
```



2.4 via une boucle + dots

2.4.1 Code

```
img = imread('assets/LENNA.BMP');
Z = img;
Z(1:15:256,1:15:256) = Z(1:15:256,1:15:256)*0;
% matrice filtre moyenneur
M = ones(3,3)/9;
img_M = imfilter(Z,M);
% ligne, colonne
[l, c] = size(Z);
dots=[];
for n=2:l-1
    for m=2:c-1
        dots(1) = Z(n-1,m-1);
        dots(2) = Z(n-1,m);
        dots(3) = Z(n,m-1);
        dots(4) = Z(n,m);
        dots(5) = Z(n+1,m);
        dots(6) = Z(n,m+1);
        dots(7) = Z(n+1,m+1);
        dots(8) = Z(n+1,m-1);
        dots(9) = Z(n-1,m+1);
        moyenne=mean2(dots);
        if(moyenne - Z(n,m)>50)
            Z(n,m) = moyenne;
        end
    end
end
% img_M = imfilter(Z,moyenne);
```

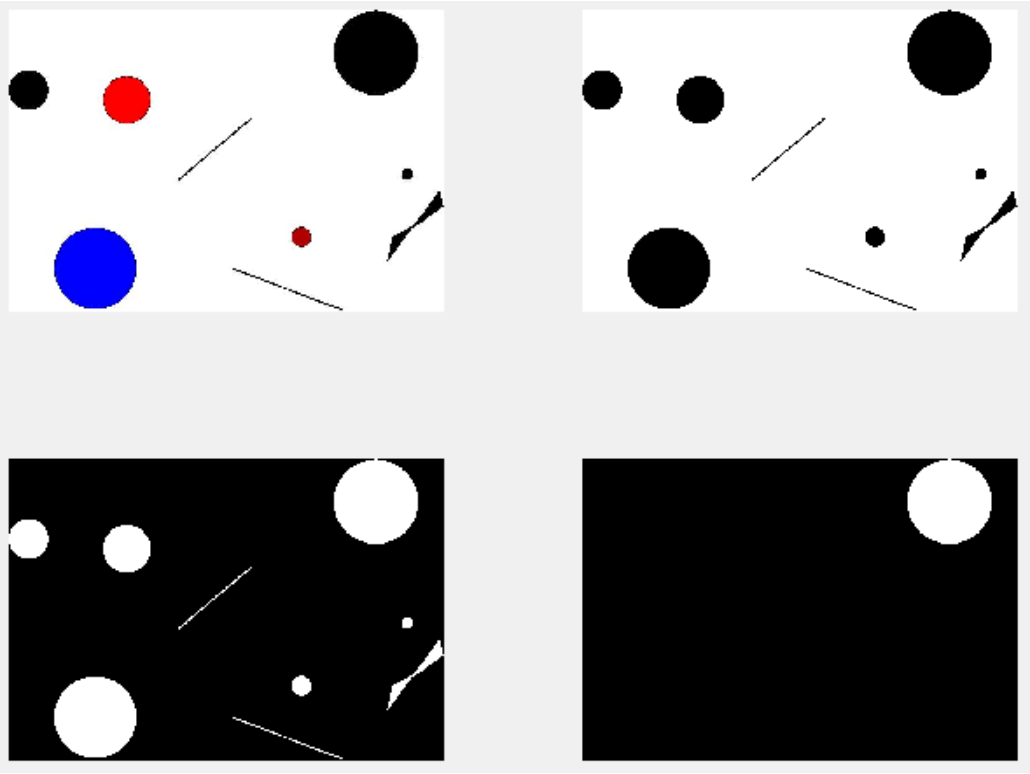
2.4.2 Affichage

2.5 Labélisation et suppression

2.5.1 Code

```
img = imread('assets/bwlabel_exc.jpg');
subplot(2,2,1);
imshow(img);
% On converti l'image en noir & blanc
imgBw = im2bw(img);
subplot(2,2,2);
imshow(imgBw);
% labelise
imgReverse = ¬imgBw;
imgLabel = bwlabel(imgReverse);
L = regionprops(imgLabel,'all');
% erosion
se = strel('disk',4);
imgErod = imerode(imgLabel,se);
subplot(2,2,3);
imshow(imgReverse);
% select biggest region
maxArea=0;
for i=1 : size(L)
    if maxArea < L(i).Area
        maxArea = L(i).Area;
        nbrDePieces=i;
    end
end
% bwareaopen
imopen = bwareaopen(imgLabel, L(nbrDePieces).Area);
subplot(2,2,4)
imshow(imopen);
```

2.5.2 Affichage

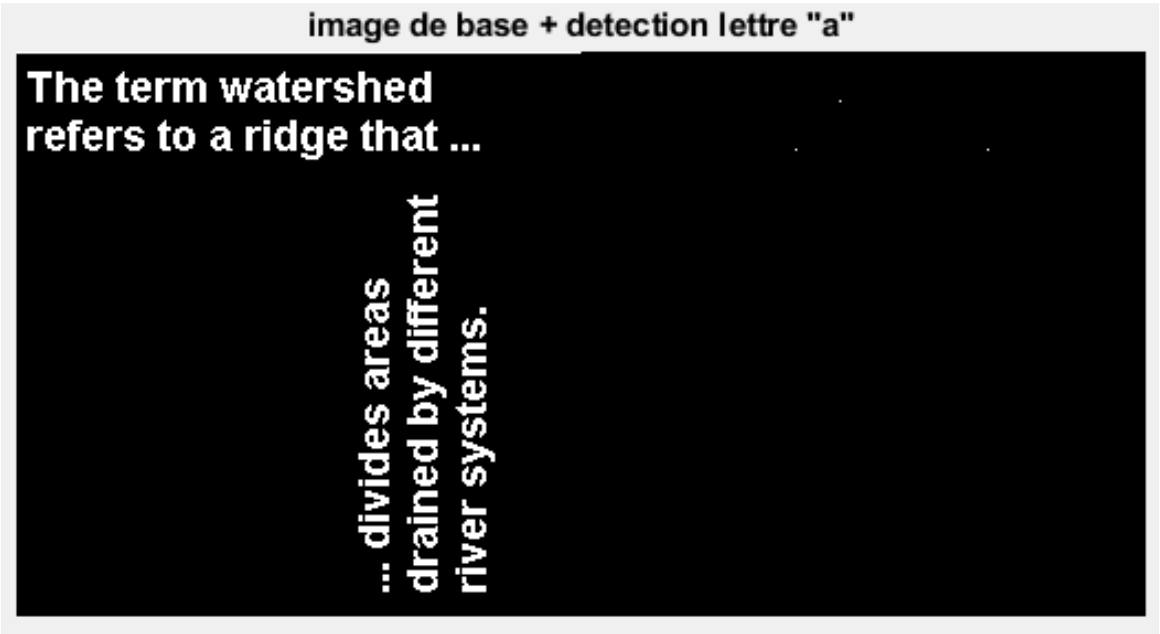


2.6 détection emplacement d'une lettre - OCR

2.6.1 Code

```
% importation text, a
text = imread('assets/text.png');
a = imread('assets/a.png');
text = im2bw(text);
% a = im2bw(a);
% extraction du "a" dans le "text"
a = text(32:45,88:98);
% correlation
C = real(ifft2(fft2(text) .* fft2(rot90(a,2),256,256)));
% [] permet de choisir la bonne taille d'affichage.

val = max(C(:));
disp('thresh de C :');
disp(val);
% Utiliser threshold un peu inferieur au max.
 thresh = 60;
% affiche les pixel au dessus du threshold.
imshowpair(text,C > thresh,'montage')
title('image de base + detection lettre "a"');
```

2.6.2 Affichage

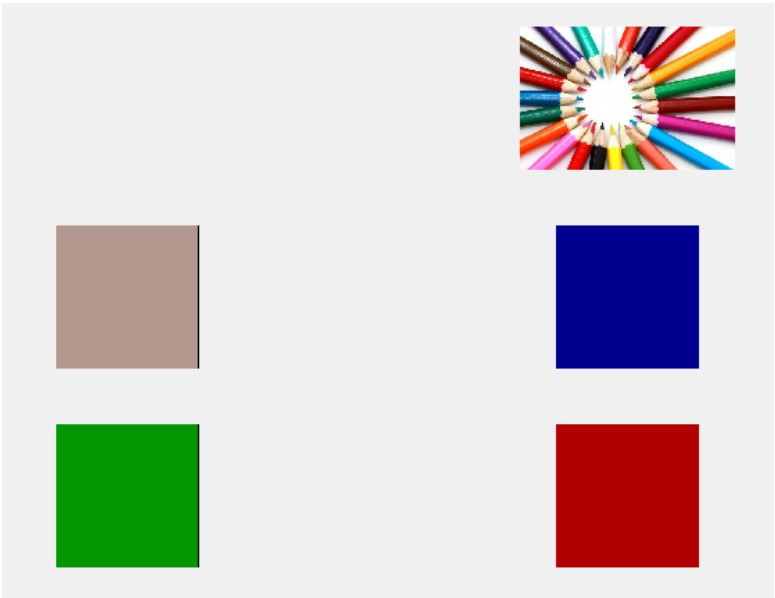


2.7 Moyenne des couleurs

2.7.1 Code

```
img = imread('assets/crayons.jpg');
% 1. cr ation de carr s avec les moyennes de couleur.
red = img(:,:,1);
green = img(:,:,2);
blue= img(:,:,3);
% mean RGB
% mean : permet de faire une moyenne
% on divise par 255 pour avoir la couleur moyenne de chaque couleur
meanR = mean2(red)/255;
meanG = mean2(green)/255;
meanB = mean2(blue)/255;

% creation d'un carre
% avec la moyenne de toute les couleurs
carre1 = ones(300,300);
carre1(:,:,1)=meanR;
carre1(:,:,2)=meanG;
carre1(:,:,3)=meanB;

carre2 = ones(300,300);
carre2(:,:,1)=0;
carre2(:,:,2)=0;
carre2(:,:,3)=meanB;

carre3 = ones(300,300);
carre3(:,:,1)=0;
carre3(:,:,2)=meanG;
carre3(:,:,3)=0;

carre4 = ones(300,300);
carre4(:,:,1)=meanR;
carre4(:,:,2)=0;
carre4(:,:,3)=0;
```

2.7.2 Affichage



2.8 Traitement image pout.tif- héééé

2.8.1 Code

```
img = imread('assets/pout.tif');
%subplot(2,2,1)
%imshow(img);
img_bw = imbinarize(img);
%subplot(2,2,2)
%imshow(img_bw);
[w, h] = size(img_bw);
% remplacement chaque pixel par la valeur du pixel de gauche moins celui du
% dessous
for x=2 : w - 1
    for y = 2 : h - 1
        img_replace(x,y) = img_bw(x-1,y) - img_bw(x,y+1);
    end
end
subplot(3,1,1);
imshowpair(img_bw,img_replace,'montage');
title 'orignal(bw) -> replacement pixel';

H = [0 -1 0 ; -1 5 -1 ; 0 -1 0];
G = [1 1 1; 1 4 1; 1 1 1];
F = [0 0 0; 1 0 0; 0 -1 0];
contraste = imfilter(img_bw, G);
subplot(3,1,2)
imshowpair(img_replace, contraste,'montage')
% bords
canny = edge(img_bw,'canny');
%subplot(3,1,2);
%imshowpair(img_replace,canny,'montage');
%title 'Remplacement pixel -> Canny';
sob = edge(img_bw,'sobel');
subplot(3,1,3);
imshowpair(canny,sob,'montage');
title 'Canny -> Sobel';
```

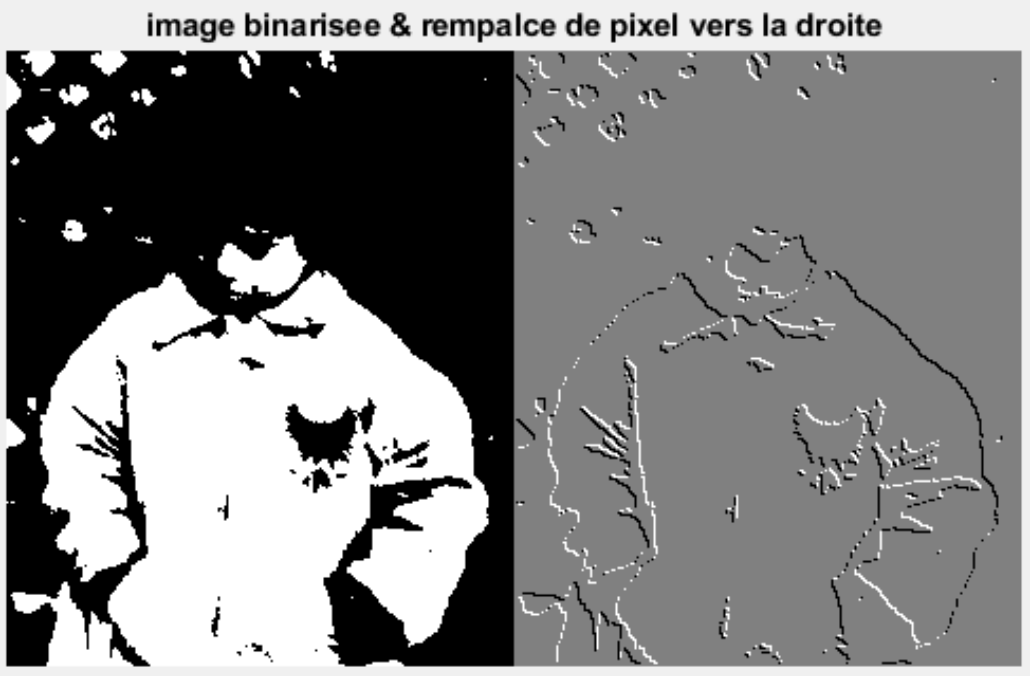
2.8.2 Affichage

2.9 Décalage pixels

2.9.1 Code

```
img = imread('assets/pout.tif');
% image deja en niveau de gris
% level & im2bw (deprecied)
% level = graythresh(img);
% img_bin = im2bw(img,level);
img_bin = imbinarize(img);
% w, h : width, hight
[w, h] = size(img_bin);
for x=2 : w-1
    for y=2 : h-1
        img_replace(x,y) = img_bin(x-1,y) - img_bin(x,y-1);
    end
end
imshowpair(img_bin,img_replace,'montage');
title('image binarisee & rempalce de pixel vers la droite');
```

2.9.2 Affichage



3 Théorie

Filtreutiles

3.0.1 Filtre passe haut

Amélioration du contraste

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

3.0.2 Filtre passe bas

Adoucit les détails

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

3.0.3 Détection de bord

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3.0.4 Repoussage

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$