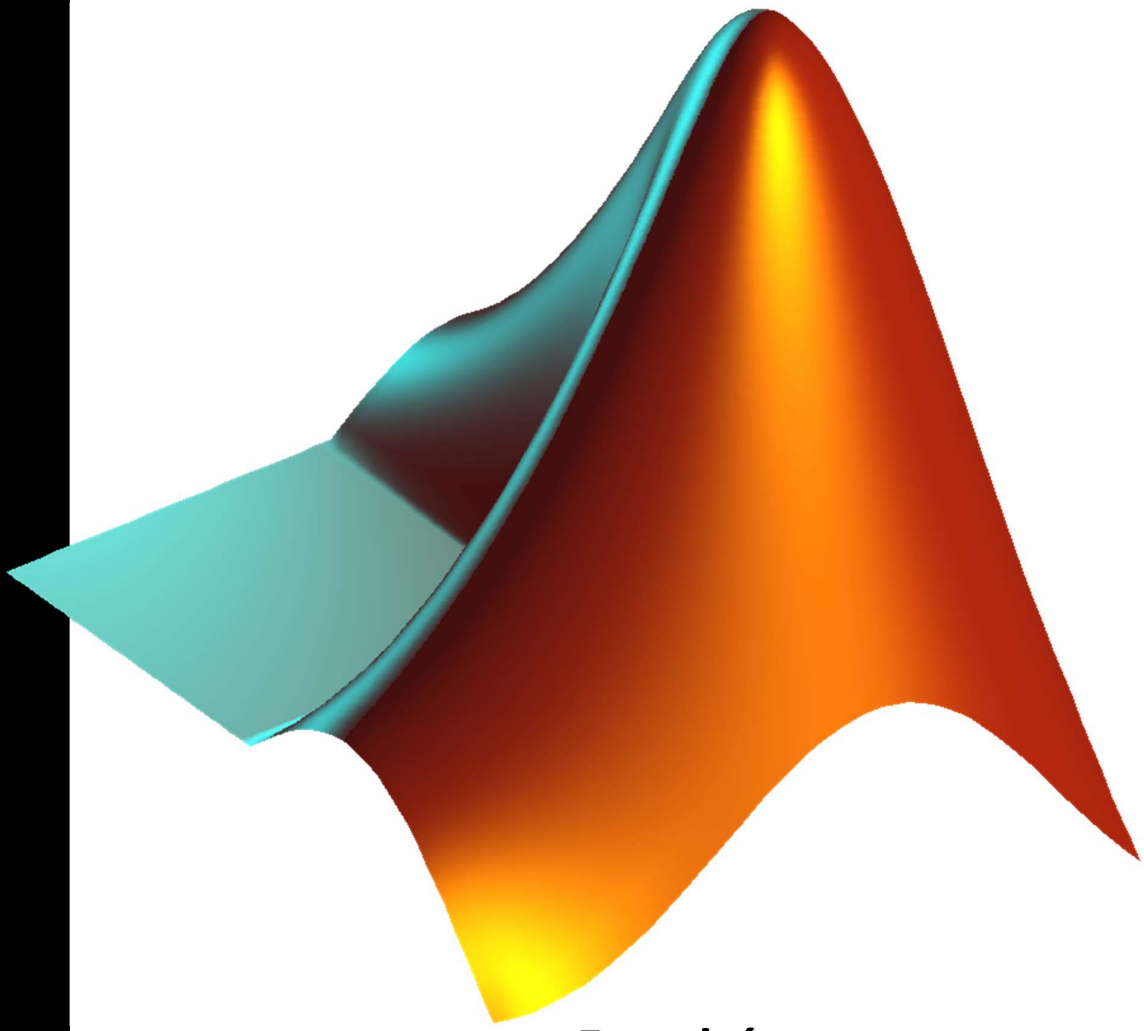
UNIVERSITÉ PARIS 13

Université de Paris 13_Institut Galilée

Travaux pratique Traitement d'image

Mini-Projet 15



Réalisé par :

- MESSILI Imad
- N° : 11710395

Encadré par :

M. LADAYCIA Abdelhamid

Année universitaire 2017/2018

Image synthétique :

Question 1. Générer une image de taille 330 x 470 composée d'un fond uni de couleur noire et d'un disque de rayon égal à 25 pixels et de couleur gris clair.

```
clc
clear all
close all
%%
%b) le cercle :
X=zeros(330,470); % une matrice de zéros de taille 330*470
x=0:329;
y=0:470;
[X1,Y]= ndgrid (x,y); % trasformer l'image en une matrice bi-dimentionnelle
avec des coordonées.
X((X1-25).^2+(Y-25).^2 < 36^2/2)=0.9; % équation du disque.
figure(1)
imshow(X) % Affichage de l'image originelle
title('Image 1 : Image synthetisee disque');
```

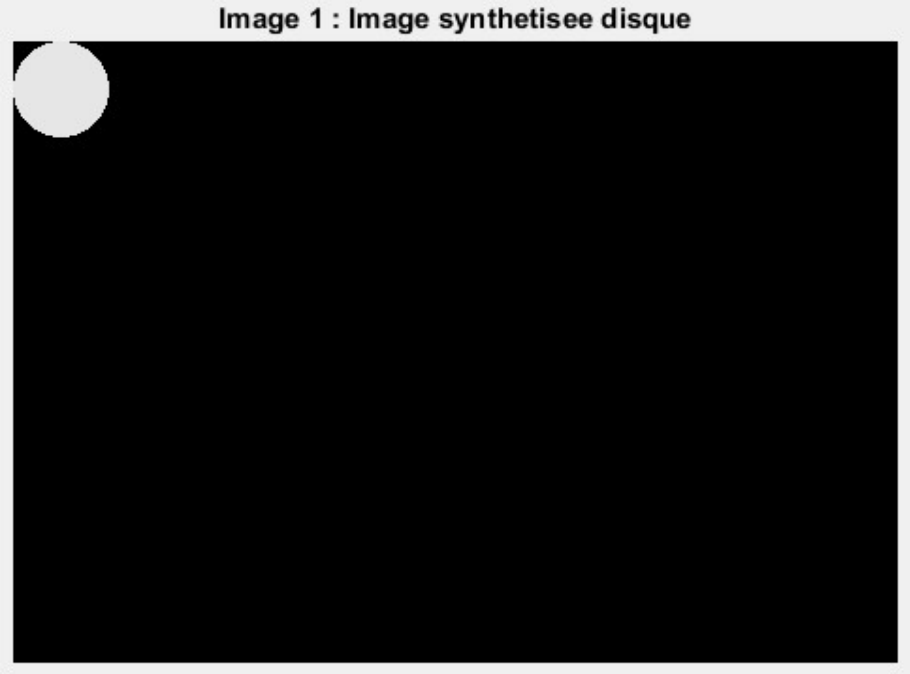
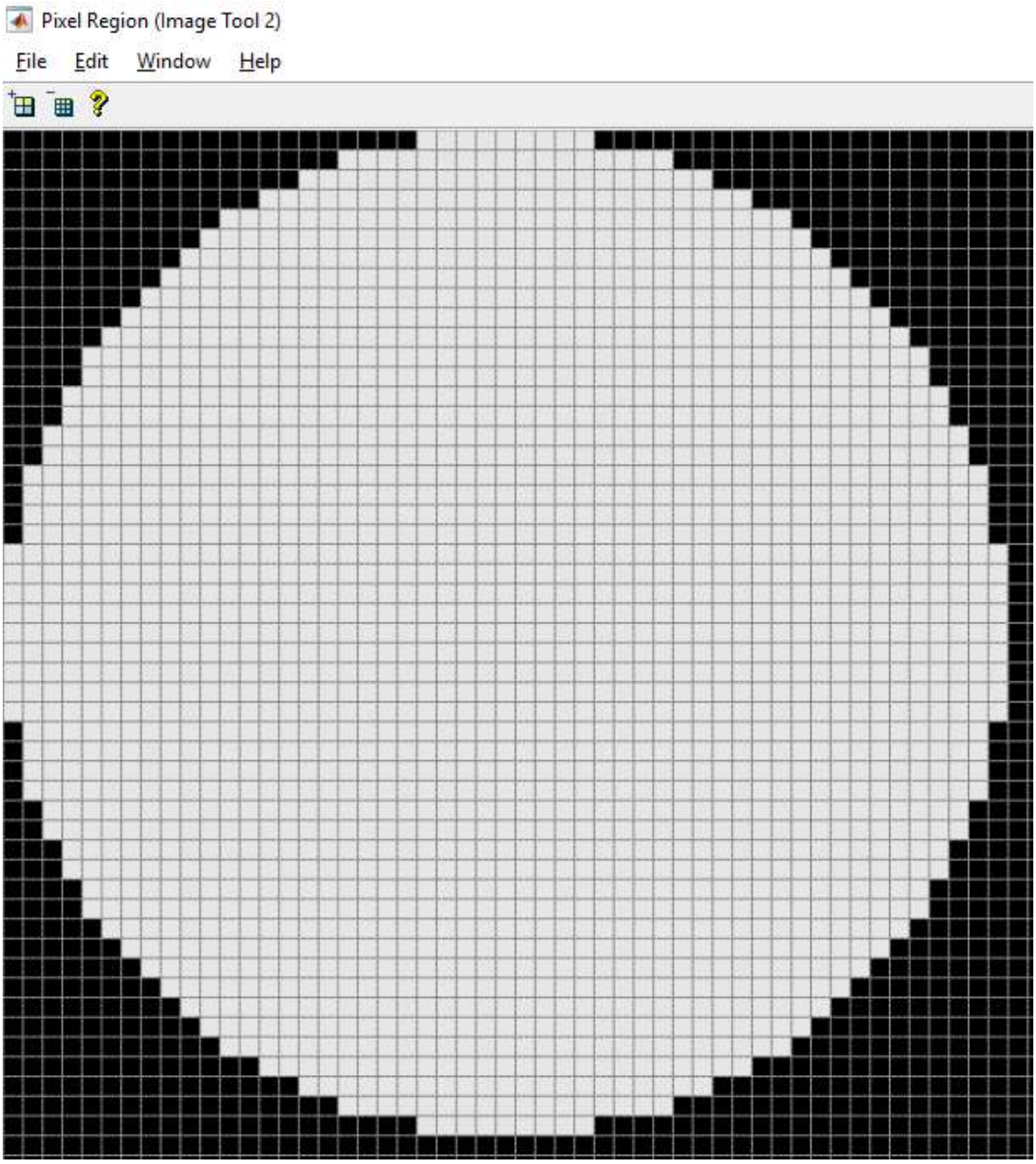


Image 1 : Image synthétisée

Pour voir les pixels de notre disque on peut utiliser l'outil que Matlab nous fournit

```
imtool(X) % affichage de l'image avec l'outil imtool
```



Afin de s'assurer que notre disque a un rayon de 25 pixels un petit script peux nous aider à le faire.

```
BW = edge(X); % détecte les contours sur une image en niveau de gris.
[r1,c1] = find(BW); % pour trouver les countours.
subplot(1,2,1);
spy(BW); % détecte l''eparpillement des pixels dans une matrice.
x2 = max(r1); % la valeur maximale sur l'axe des 'x'
x1 = min(r1); % la valeur minimale sur l'axe des 'x'
x = x2 - x1   % la différence entre les deux extrimitées.
y2 = max(c1); % la valeur maximale sur l'axe des 'y'
y1 = min(c1); % la valeur minimale sur l'axe des 'y'
y = y2 - y1   % la différence entre les deux extrimitées.
subplot(1,2,2);
imshow(BW)
rayon = (x+y)/4 % le rayon du cercle en pixels
```

Résultats :
x = 50
y = 50
rayon = 25 « le résultat est bon et notre rayon a 25 pixels.

Image naturelle en niveaux de gris

Question 2. Ouvrir l'image image15.jpg et la convertir en niveaux de gris. Vérifier qu'elle est bien codée sur 8 bits et l'afficher.

```
X= imread('image15.jpg'); % Lecture de l'image.
Y=rgb2gray(X); % convertion de l'image de RGB au niveau de gris.
figure(2)
imshow(Y) % affchage de la nouvelle image.
title('Image 2 : Image naturelle en niveaux de gris')
```

Vérifier qu'elle est bien codée sur 8 bits :

```
>> whos X
  Name        Size              Bytes  Class    Attributes

  X           256x384x3         294912 uint8
```

Affichage de l'image :

Image 2 : Image naturelle en niveaux de gris

Question 3. Requantifier l'image sur 6 bits. Afficher l'image quantifiée et commenter. Le bruit de quantification est-il visible ? Si oui, dans quelle zone semble-t-il plus présent ? Tracer sur la même figure (subplot) l'histogramme de l'image originale et celui de l'image requantifiée. Commenter. Afficher la valeur absolue de la différence entre l'image originale renormalisée et l'image quantifiée renormalisée sous la forme d'une image. Commenter.

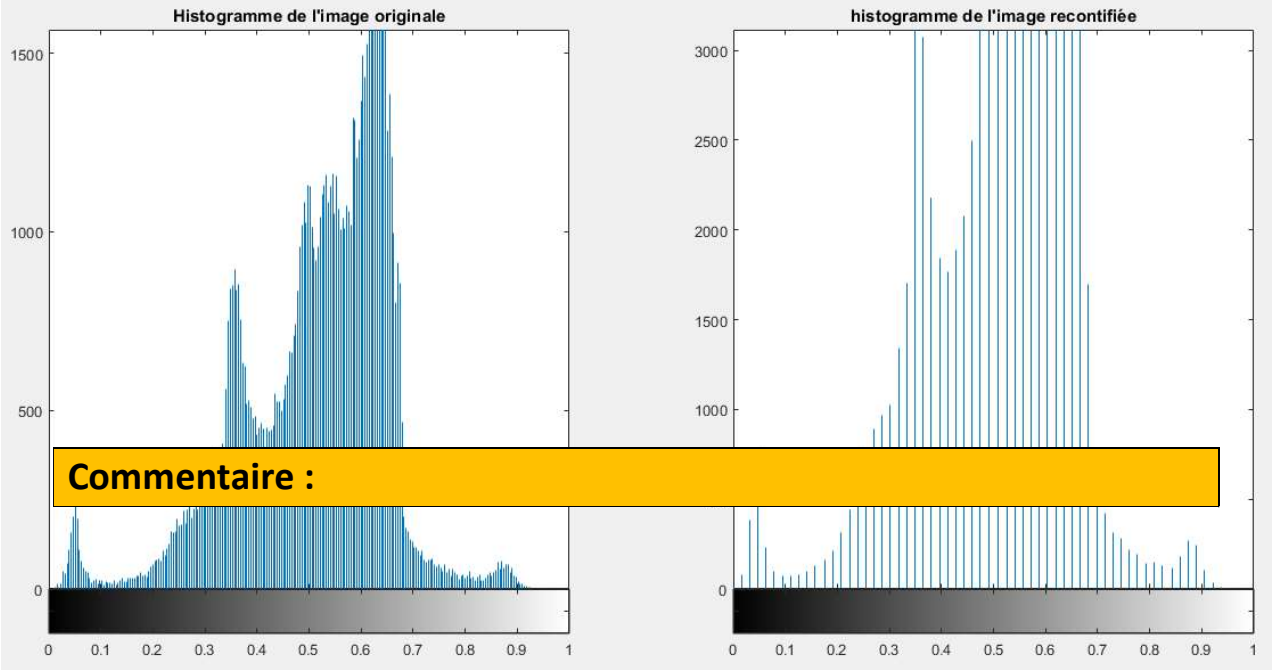
```
%% quantifier l'image en 6 bits :
%%
Y=double(Y)/255; %convertir l'image en double et la normalisée
n=64;
d=255/n; %le pas de quantification
Y1=floor(Y*255/d)/(n-1); % l'equation de quantification.
figure(3)
subplot(1,2,1)
imshow(Y)
title('8 niveaux')
subplot(1,2,2)
imshow(Y1)
title('Image 3 : Image requantiee 6 niveaux')
```

Commentaire :

La requantification de l'image en 6 niveaux, n'ajoute pas une très grande dégradation sur la qualité de l'image. Cependant un bruit dû à la baisse du nombre limité d'échantillons pris est constatable.

Le bruit de quantification est bien constatable sut la partie supérieure de l'image (le ciel).

```
%% les Histogrammes:
%%
figure(5)
subplot(1,2,1)
imhist(Y)
title('Histogramme de l''image originale')

subplot(1,2,2)
imhist(Y1)
title('histogramme de l''image recontifiée')
```

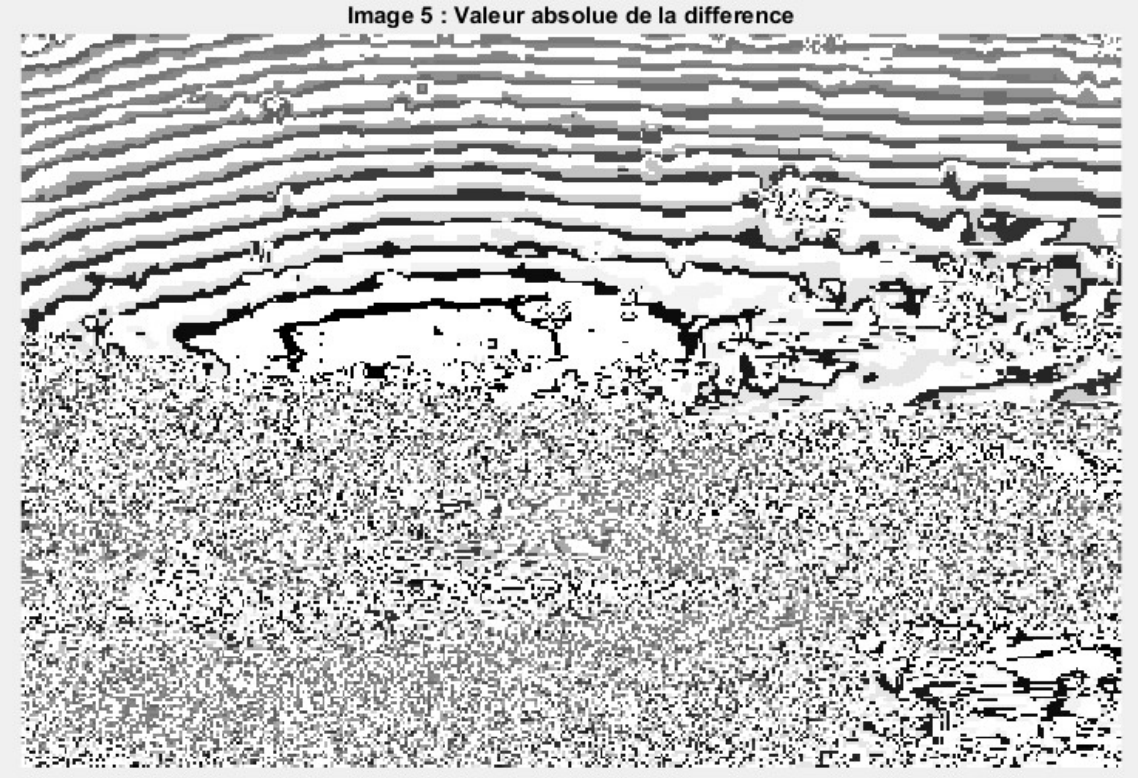


Commentaire :

Le nombre d'échantillons dans l'histogramme de l'image originale est le double du nombre d'échantillons dans l'histogramme de l'image requantifiée.

Cependant et malgré la requantification et la perte de presque la moitié des échantillons l'histogramme garde sa forme et son allure.

```
%% valeur absolue de la différence :
%%
Y2=abs(Y1-Y);
figure(6)
imshow(Y2*255)
title('Image 5 : Valeur absolue de la difference')
```



Commentaire :

Le bruit de quantification se reparti sur la zone supérieure de l'image (le ciel) d'une manière bien déterminée (en forme de vagues), ce qui le rend apparent à l'œil nue. Alors que le bruit de quantification se reparti sur la zone inferieure de l'image d'une manière aléatoire.

Question 4. Afficher sous forme d'image le module de la transformée de Fourier de votre image (en échelle linéaire ou logarithmique selon ce que vous souhaitez observer). Commenter (basses et hautes fréquences, directions privilégiées, etc.…).

```
% fft de l'image :
Y3=(fft2(Y)); % la fast Fourier transform sans arrangement
Y4= fftshift(Y3); % la fast Fourier transform avec arrangement
Y5= abs(Y4);

%%  echelle lineaire
figure(8)
Y6=mat2gray((Y5));

%% en echelle logarithmique :
Y7=10*log10(10+Y5);
Y8=mat2gray(Y7);

figure(10)
imshow([Y6 Y8])
title('Image 6 : Transformee de Fourier de l''image')
```

Image 6 : Transformée de Fourier de l'image

Commentaires :

Vu que l'affichage en échelle linéaire ne permet pas d'extraire assez d'information on préfère se tournée vers le logarithmique.

- L'image contient plus de basse fréquences que de hautes fréquences, et ceci peut être expliqué par le manque de contour dans l'image et l'homogénéité des couleurs (ciel et mer).
- La direction verticale est privilégié.

Question 5.

Renormaliser l'image et y appliquer un bruit poivre et sel compromettant p = 55% des échantillons. Afficher l'image bruitée et commenter. Appliquer sur l'image bruitée un filtre moyenneur de taille 7*7. Afficher l'image filtrée et commenter. Le filtrage vous semble-t-il adapté pour débruiter l'image ? Sinon, quel filtrage privilégieriez-vous ? Calculer le PSNR et commenter.

```
%% Appliquer un bruit sel et poivre sur l'image:
%%
Yn=imnoise(Y,'salt & pepper',0.55); % application du bruit
figure(11)
imshow([Y,Yn]); % affichage des 2 images bruitée et non bruitée.
title('Image 7 : Image bruitee Original and Noised Picture Salt and
pepper')
```

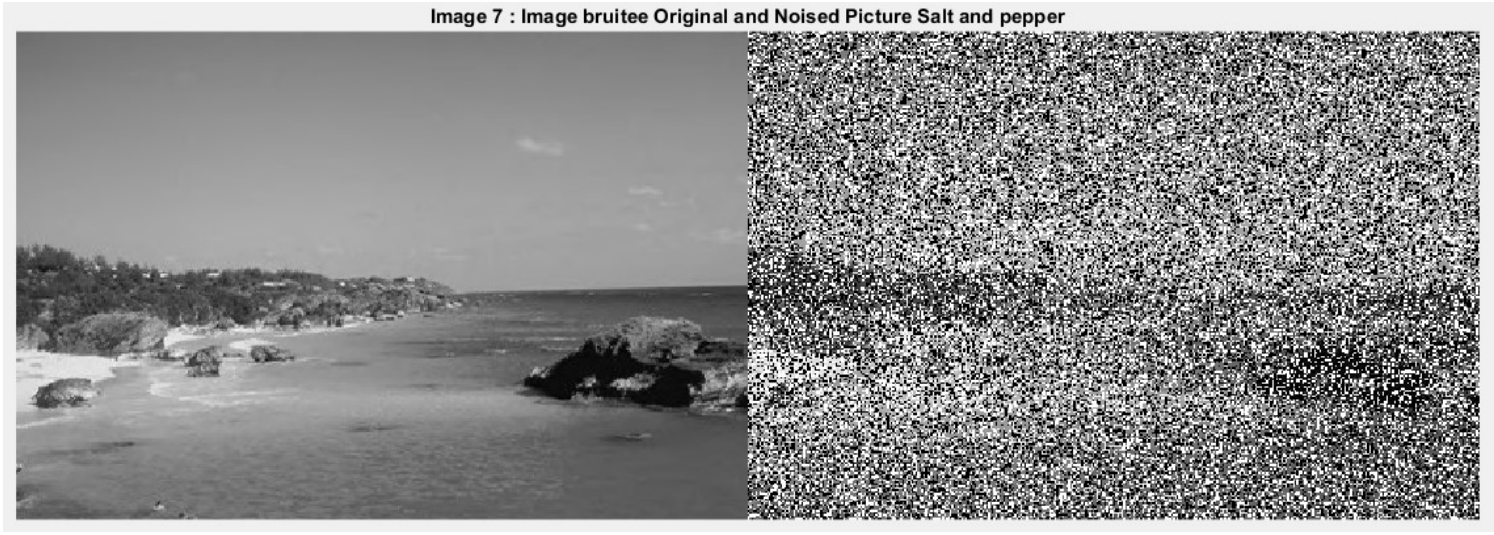


Image 7 : Image bruitée

Commentaires:

Le bruit poivre et sel (impulsionnel) est un bruit aléatoire, dans notre cas le bruit occupe 55% des pixels (bruit puissant), ce qui rend l'image non reconnaissable.

Appliquer un filtre moyenneur :

```
h=fspecial('average',[7 7]);
% application du filtre moyenneur :
Yfil=imfilter(Yn,h,'replicate');
figure(12)
imshow(Yfil)% Affichage des 3 signaux.
title('Image 8 : Image filtré avec un filtre moyenneur')
```

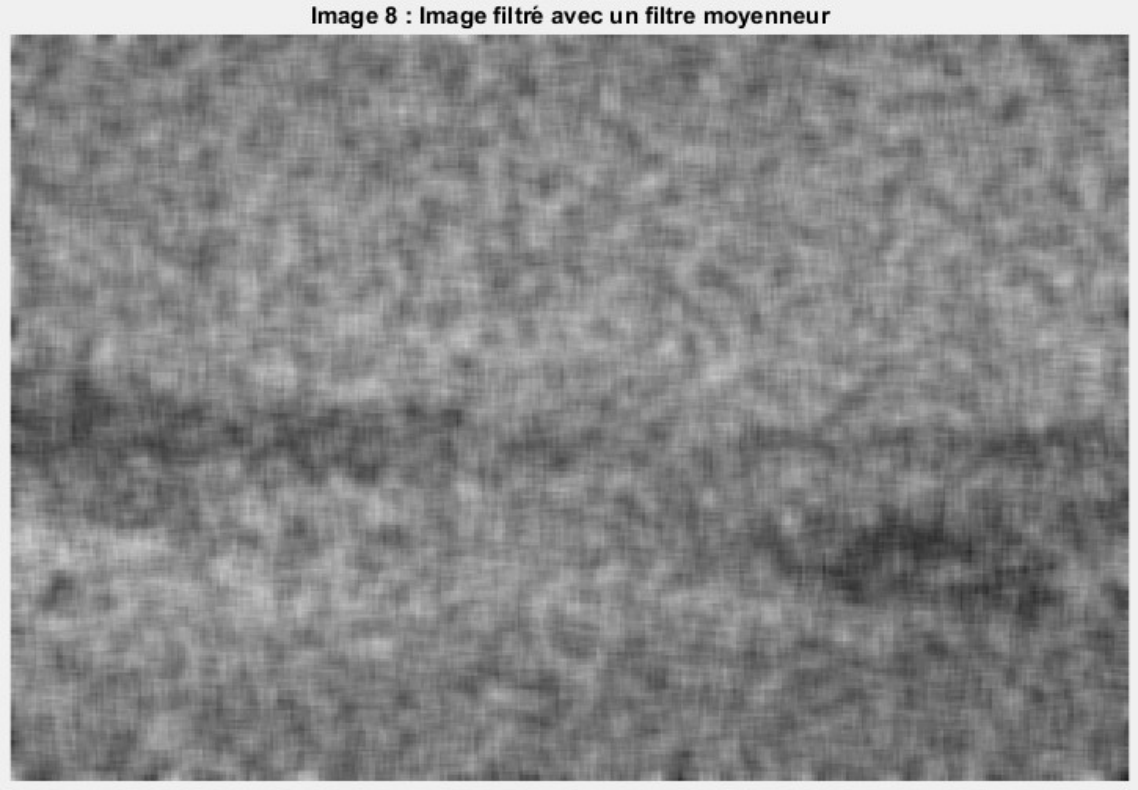


Image 8 : Image filtrée

Commentaire :

Après le filtrage l'information est totalement perdue et on ne peut rien distinguer.

- Non le filtre moyenneur ne semble pas adapté à ce genre de bruit.

Le filtre le plus adapté au bruit sel et poivre et le filtre médian.

```
%% 4- Application d'un filtre médian :
%%
Ymed=medfilt2(Yn, [7 7]);
figure(13)
imshow(Ymed)% Affichage des 3 signaux.
title('signal filtré Médian');
```

Commentaire :

Malgré l'importante puissance du bruit poivre et sel, un filtre médian avec les mêmes dimensions (7x7) que le moyenneur nous fournit une image filtré mieux reconnaissable et moins bruitée.


```
%% 6-Calculer le PSNR pour les deux simulations precedemment realisees.
%%
PSNR1=-10*log10(std2(Y-Yfil))% PSNR du filtre moyenneur
PSNR2=-10*log10(std2(Y-Ymed))% PSNR du filtre médian
```

PSNR1 = 9.9298

PSNR2 = 11.1201

Commentaire :

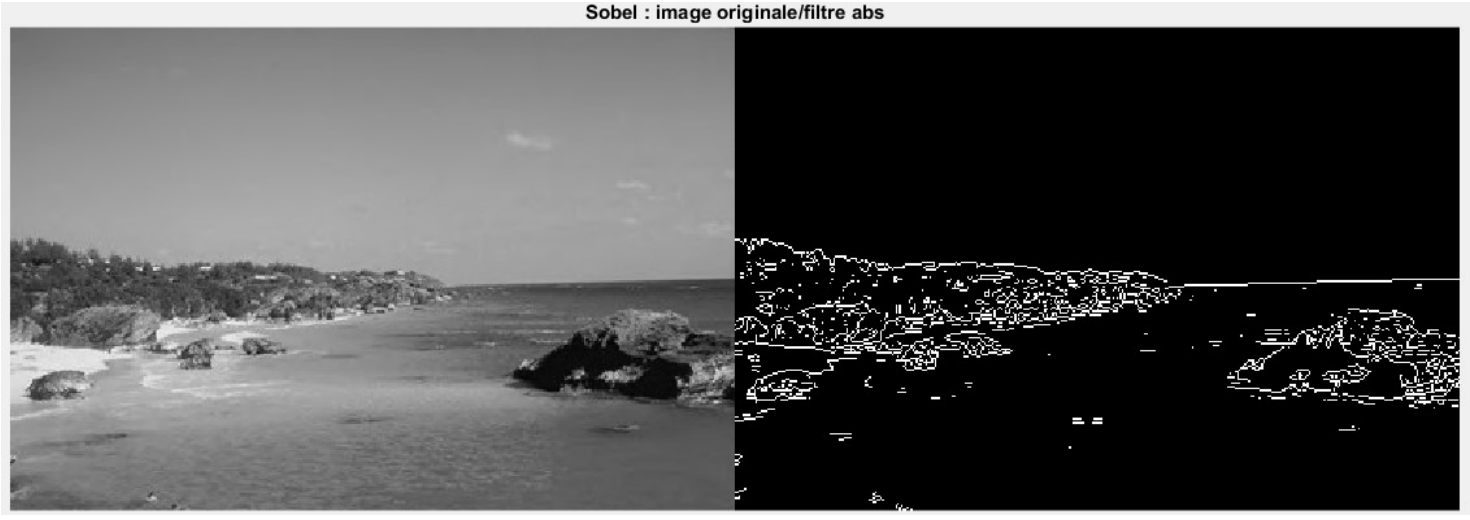
Le PSNR obtenue avec le filtre médian est plus important que celui obtenu avec le filtre moyenneur. Le filtre médian est mieux adapté au bruit poivre et sel

Question 6. Choisir les contours que vous souhaitez faire apparaitre et ceux que vous ne souhaitez pas faire apparaitre (et les décrire). Tester différentes techniques (filtre gradient, filtres de Sobel, filtre LOG) et choisir celle qui donne les meilleurs résultats par rapport aux objectifs que vous vous étiez fixés. Décrire la méthode finalement utilisée et afficher l'image filtrée obtenue mettant en évidence les contours.

Les contours qu'on souhaite faire apparaitre :

Les contours du ciel

```
%% filtre Gradient :
%%
X=imread('image15.jpg');
X=double(X)/255;
X=rgb2gray(X);
h1=[0 1 2;-1 0 1;-2 -1 0];
Yg=imfilter(X,h1);
figure(14)
imshow([X mat2gray(abs(Yg))])
title('Gradient :image originale/filtre abs');


%% filtre Log :
%%
% h2=fspecial('LOG',[7 7],0.5);
% Yl=imfilter(X,h2);
Yl=edge(X,'log');
figure(15)
imshow([X mat2gray(abs(Yl))])
title('Log : image originale/filtre abs');


%% filtre Sobel :
%%
Ys=edge(X,'sobel');
figure(16)
imshow([X mat2gray(abs(Ys))])
title('Sobel : image originale/filtre abs');
```



Log : image originale/filtre abs

Gradient :image originale/filtre abs

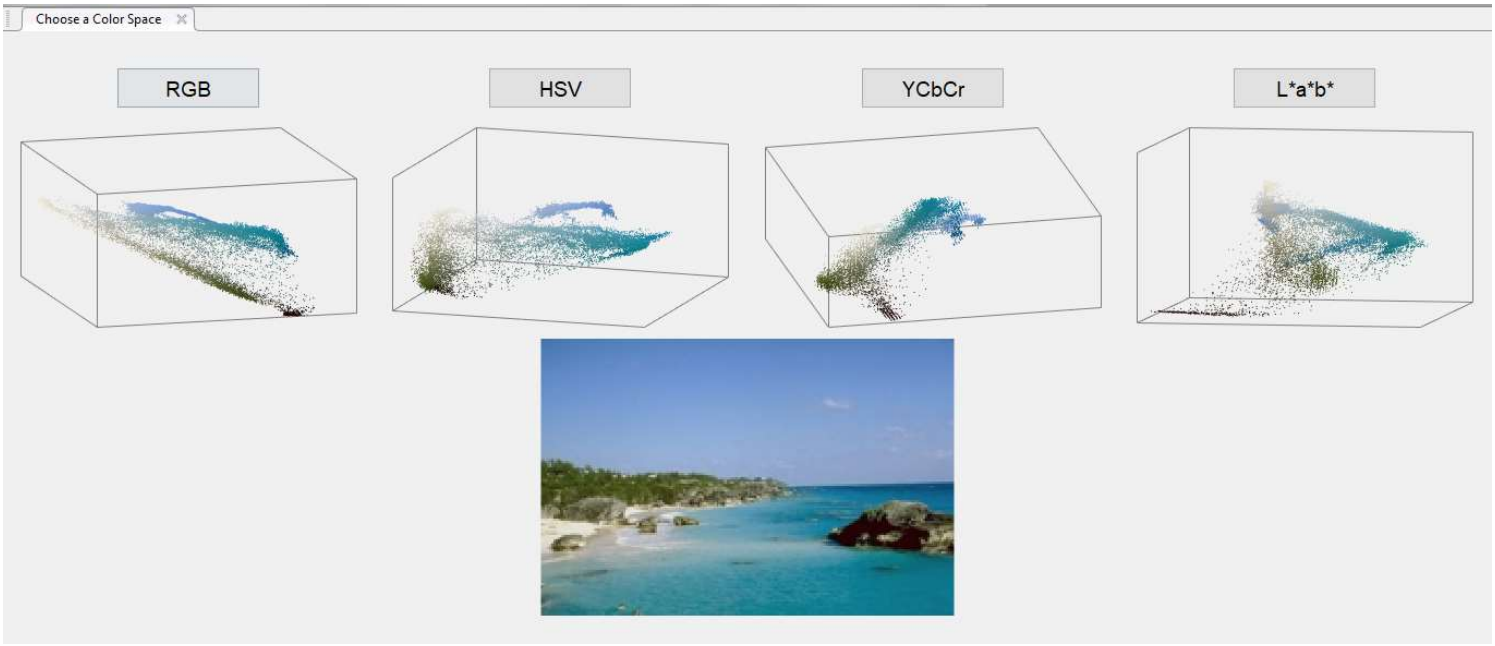
Le filtre Sobel est le filtre le mieux adapté à notre souhait car il décrit les contours entre le ciel et le reste des composantes de l'image.

Image naturelle en couleurs

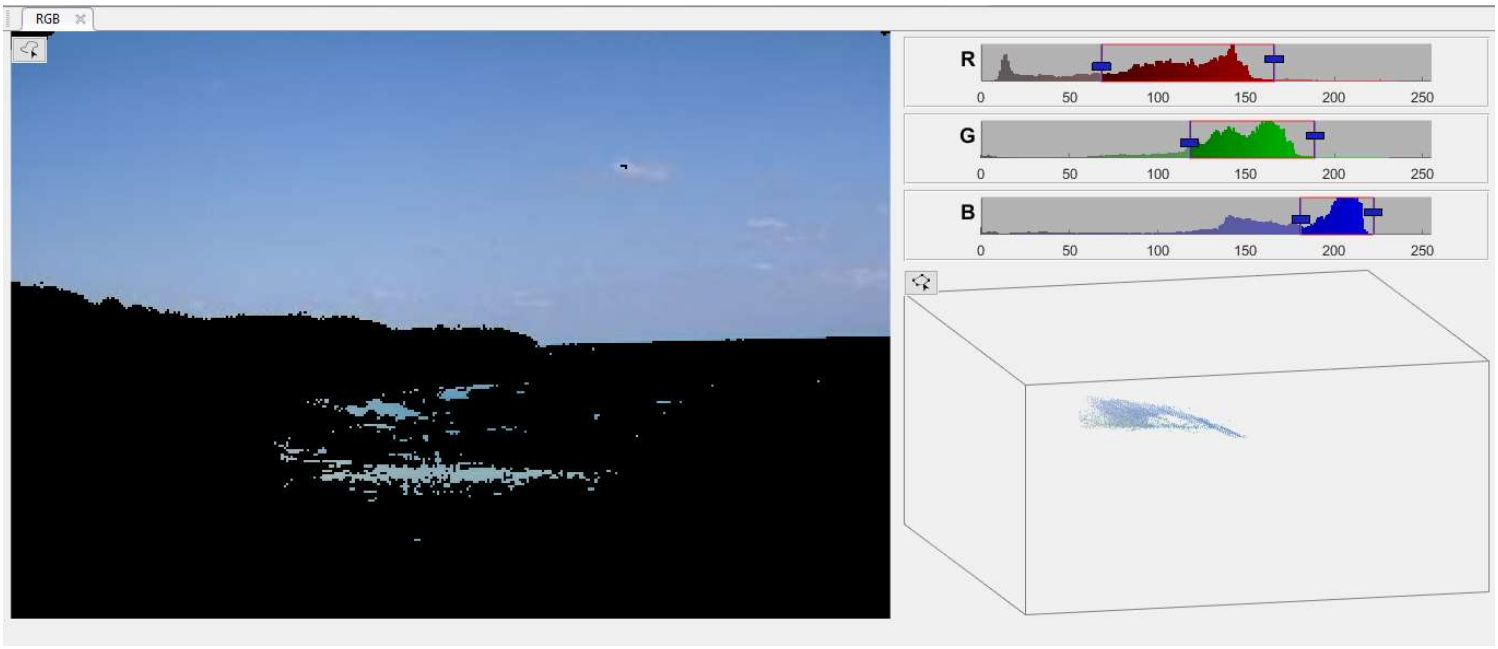
Question 7.

Ouvrir l'image image15.jpg, la laisser en couleurs et l'afficher. Proposer une méthode pour que les pixels correspondant au ciel (ou à la mer, au choix) deviennent de couleur orange. La décrire et afficher le résultat.

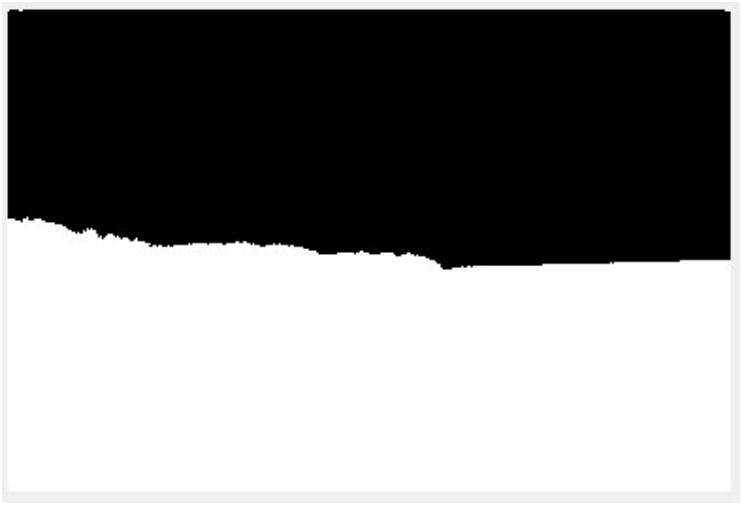
Avant tout il faut récupérer les plages concernées par le changement pour chaque composante. Et cela on va utiliser un outil que matlab nous fournit et qui se trouve dans l'anglet APPS (colorthreshold).



On choisit RGB et on choisit nos plages afin d'isoler le ciel :



```
%% Image naturelle en couleurs
%% Lecture de l'image en couleur et l'afficher :
X= imread('image15.jpg');;
X=double(X);
figure(17)
imshow(X/255)
%% récuperation des région qui représenent le ciel dans les 3 composntes :
red=X(:,:,1);
R_bin= (red==0 & red<=75); % composante rouge
blue=X(:,:,2);
B_bin= (blue>100 & blue<=220); % composante bleue
green=X(:,:,3);
G_bin= (green>75 & green<=175); % composante rouge
X_bin=cat(3,R_bin,G_bin,B_bin); % reconstitution de l'image aux 3
composantes
X_bin=rgb2gray(double(X_bin));
X_bin=double(X_bin);
X_bin= imbinarize(X_bin); % convertir l'image de niveau de gris en image
binaire

X_bin=imfill(X_bin,'holes'); % eliminer les imperfections et remplir les
vides
X_bin=imcomplement(X_bin); % image complémentaire de l'image binaire
ontenue
X_bin=imfill(X_bin,'holes'); % deuxième tentative d'eliminer les
imperfections
label=bwlabel(X_bin); % diviser l'image en plusieurs labels afin de pouvoir
en choisir celui qui nous interesse.
lab=imcomplement(label==1); % complementer l'image obtenue du label numéro
1.
figure(19)% afficher le label 1
imshow(lab)



%% Changer la couleur du ciel (orange):
%% extraction de chaque composante à part :
R=X(:,:,1);
G=X(:,:,2);
B=X(:,:,3);



%%
Y1=(R).*lab; % multiplication de la premiere composnate par l'image filtré
Y1(Y1==0)=255; % chagement des valeurs des pixels correspondant au ciel
Y2=(G).*lab; % multiplication de la deuxième composnate par l'image filtré
Y2(Y2==0)=165; % chagement des valeurs des pixels correspondant au ciel
Y3=(B).*lab; % multiplication de la troisième composnate par l'image filtré
Y3(Y3==0)=0; % chagement des valeurs des pixels correspondant au ciel
Y=cat(3,Y1,Y2,Y3); % regrouper les trois composantes
figure(20) % afficher l'image finale (ciel orange)
imshow(Y/255)
```



Label 1 filtre à appliquer sur l’image originale

l'image finale (ciel orange)