Traitement du signal

Rapport

Julien Lecat & Ulysse Dahiez

Exercice 1 :

Nous avons dans un premier temps, réaliser ces image(figure 1),ensuite nous avons utilisé la méthode pour remplir les pixel dans une matrice, des pixels noir. Par la suite, nous avons ajouter le carré blanc au milieu de l’image. Ce qui ,nous a donné la figure 1.

Pour se faire nous avons utilisé : X = zeros(512, 512);

Cette fonction nous permet de remplir de zéro une matrice de 512 par 512. Donc de réalisé un carré noir.

Ensuite il faut ajouter un carré blanc, grâce à la fonction suivante.

for x=206:306

for y=206:306

X(x,y) = 255;

end

end

Pour continuer nous avons appliqué la transformé de fourrier avec la fonction suivante :

Dct = dct2(X);

Nous avons ici la variable DCT qui est la transformé de fourrier de X.

Ensuite nous avons du reconstruire l’image initial suivant différents cas. Nous avons donc pour cela utilisé Dcts = sort(m2,'descend');

Qui nous permet de trier la matrice.

Ensuite nous prenons toutes les valeurs plus grands que 80% 50% et 20%.

Dct(abs(Dct) < Dcts(pct80)) = 0;

Il nous reste donc à afficher l’image grâce au fonctions suivantes :

figure

colormap(gray(64));

plot(Dcts)

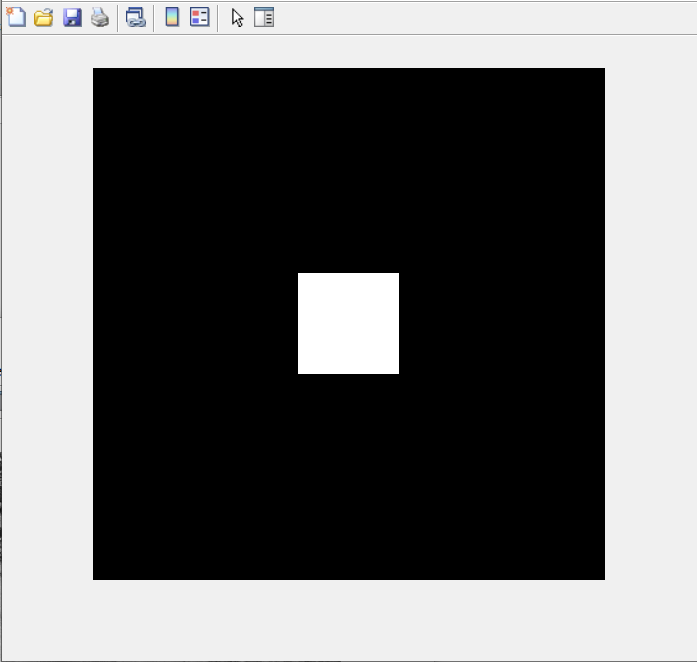
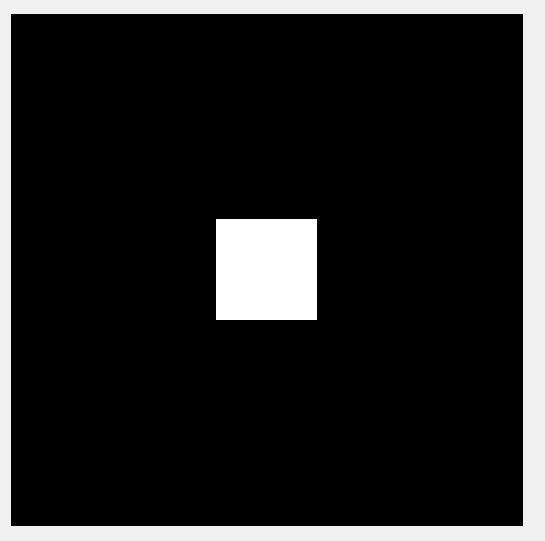
Nous permet d’afficher la courbe de la transformer de fourrier trier.

figure

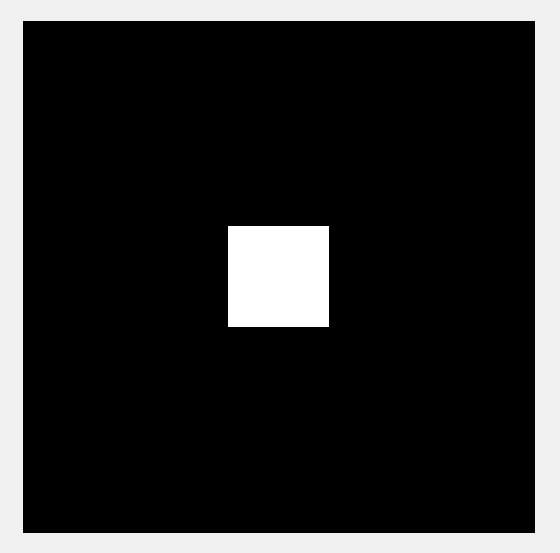
colormap(gray(64));

imshow(idct2(Dct));

Nous permet d’afficher l’image en ne prenant en compte que les valeurs plus grands que nos coefficients

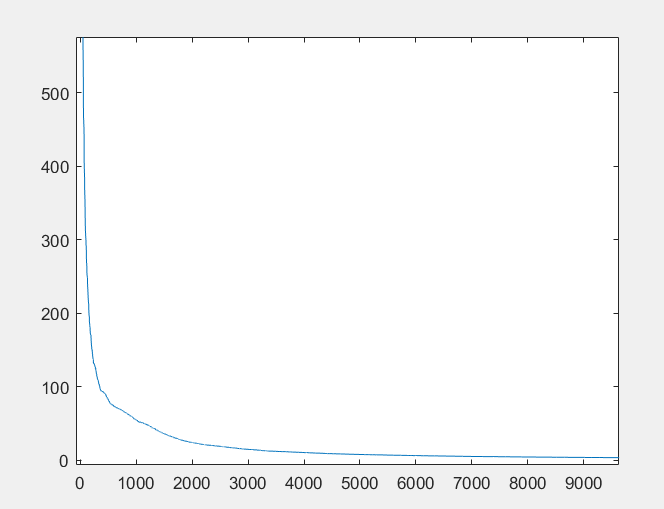
 

*Figure 1 : image de 512\*512 avec un carré blanc. Figure 2 : figure 1 compressé à 80%*

* Une image contenant texte, moniteur, afficher, ciel nocturne

Description générée automatiquement*

*Figure 3 : figure 1 compressé à 50% Figure 4 : figure 1 compressé à 20%*



*Figure 9 : Courbe de la Transformé de fourrier discrète.*

Nous pouvons en conclure que la compression par DTC ne déforme pas visuellement une image jusqu’à 20%. En effet, nous avons pu remarquer que, lorsque nous appliquons la DTC, nous obtenons une courbe, qui décroit fortement et se stabilise proche de 0. Nous pouvons en conclure de cette courbe, que les informations les plus pertinentes se trouve au début de cette même courbe. C’est pour cela que l’image n’est pas détériorée à 80% et à 50%.

Le contraste entre le noir et le blanc nous permet de bien voir les défauts (figure 4). Alors que sur les compressions à 80% et 50% aucun changement n’est visible à l’œil nu.

Exercice 2 :

Le but de cet exercice, nous devions regarder les différentes numérisations d’image et de les restituer.

Nous avons pour ce faire réaliser les même étapes que précédemment, mais en incluent l’image que l’on nous a donnée figure 1.

Grace à la fonction image1 = imread("image1.ascii.pgm");

Une image contenant primate, mammifère, yeux, fermer

Description générée automatiquement Une image contenant texte, noir, fixant

Description générée automatiquement

*Figure 5 : image non compréssé figure 6 : figure 5 compressé à 20%*

*Une image contenant texte, primate, fixant

Description générée automatiquement Une image contenant texte, fixant

Description générée automatiquement*

*Figure 7 : figure 5 compressé à 1% Figure 8 : figure 5 compressé à 0.5%*

Nous pouvons en conclure que pour minimiser les pertes au maximum, nous pouvons compresser une image détaillée jusqu’à 1%, comme sur la figure 7. L’image est certes un peu modifiée mais reste reconnaissable, lorsque nous descendons plus, les détailles de l’image se perdent (figure 8). Cependant comme le panel de couleurs est plus large que sur l’exercice 8, la compression à 20% n’a pas eu d’effet visible sur l’image. De plus que nous voyons pas de différence à l’œil nu, mais que l’on a quand même réduit la taille de l’image, ce qui nous permet de minimiser le stockage.



Affichage de l’image compressé

Application de la compression

Création de la variable de compression

Application de la transformé de fourrier

Insertion de l’image et affichage de l’image.