Rapport du cours de Traitement d’images

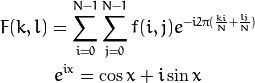
TRANSFORMÉE DE FOURIER

dans le cadre du cours de traitement d’images, il nous a été demandé d'implémenter et de tester la Transformée de Fourier définie dans OpenCV avec la fonction cvDTF.

dans les ligne qui suivent nous aurons à présenter :

1. la transformée de Fourier sur une image
2. l’inverse de la transformée de Fourier sur la même image
3. filtre passe bas et passe haut
4. Conclusion
5. **Transformée de fourier**

voici la formule de la transformée de fourier:



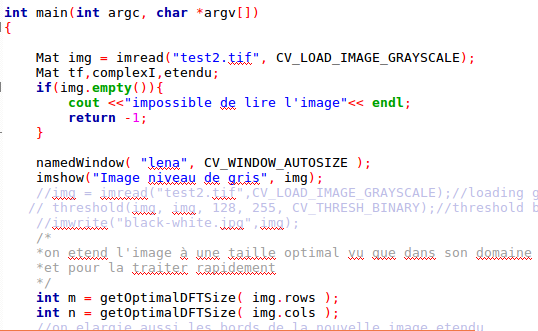
et voici l’image en entrée:



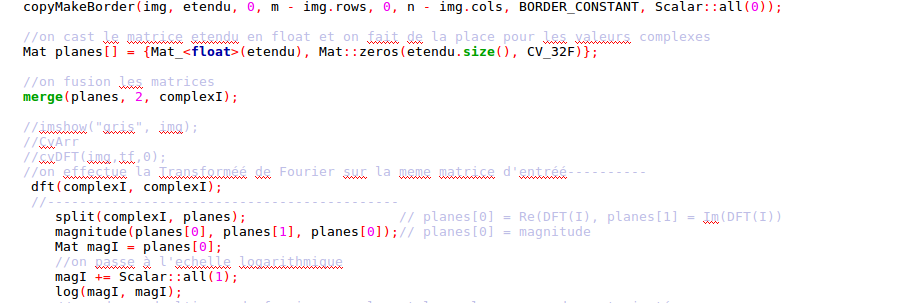
La transformée de Fourier décompose une image en ses composantes sinus et cosinus. En d'autres termes, il va transformer une image de son domaine spatial en son domaine fréquentiel. L'idée est que n'importe quelle fonction peut être approchée exactement avec la somme des sinus infinis et des fonctions cosinus. La transformation de Fourier est un moyen de le faire. Mathématiquement une transformation de Fourier d'images en deux dimensions est:

ci f est la valeur de l'image dans son domaine spatial et F dans son domaine fréquentiel. Le résultat de la transformation est des nombres complexes. L'affichage est possible soit via une image *réelle* et une image *complexe* , soit via une image de *magnitude* et de *phase* . Cependant, dans tous les algorithmes de traitement d'image, seule l'image de *magnitude* est intéressante car elle contient toutes les informations dont nous avons besoin sur la structure géométrique des images. Néanmoins, si vous avez l'intention d'apporter quelques modifications à l'image dans ces formulaires et que vous devez ensuite la retransformer, vous devrez conserver ces deux éléments.

voici le code de notre programme qui fait la transformée de fourier:



dans le code ci-haut nous commençons d’abord à charger et à redimensionner l’image car la performance d'un DFT(étant donné que nous utilisons des images numérique ,nous allons utiliser la transformée de fourier discrète) dépend de la taille de l'image. Il tend à être le plus rapide pour les tailles d'image qui sont multiples des nombres deux, trois et cinq. Par conséquent, pour obtenir des performances maximales, il est généralement recommandé d'ajouter des valeurs limites à l'image pour obtenir une taille avec de tels caractères. Le [**getOptimalDFTSize()**](http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#getoptimaldftsize) retourne cette taille optimale et nous pouvons utiliser la [**copyMakeBorder()**](http://docs.opencv.org/modules/imgproc/doc/filtering.html#copymakeborder) fonction pour étendre les frontières d'une image:

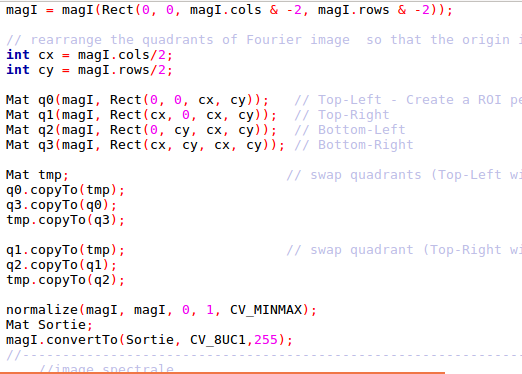


Le résultat d'une transformation de Fourier est complexe.Cela implique que pour chaque valeur d'image, le résultat est deux valeurs d'image (une par composant:composant imaginaire et réel). De plus, la gamme de domaines fréquentiels est beaucoup plus grande que sa contrepartie spatiale. Par conséquent, nous les stockons généralement au moins dans un format *flottant* . Par conséquent, nous allons convertir notre image d'entrée à ce type et l'étendre avec un autre canal pour contenir les valeurs complexes, ce qui fait sur l’image ci-haute. après cela on applique cette DFT avec la fonction dft de OpenCV

Il s'avère que la plage dynamique des coefficients de Fourier est trop grande pour être affichée sur l'écran. Nous avons des valeurs de changement faibles et élevées que nous ne pouvons pas observer de la sorte. Par conséquent, les valeurs élevées se transformeront toutes en points blancs, alors que les petites seront noires. Pour utiliser les valeurs d'échelle de gris pour la visualisation, nous pouvons transformer notre échelle linéaire en une échelle logarithmique:

magI + = Scalar :: all ( 1 ); *// passe au*   
journal d' *échelle* logarithmique ( magI , magI );

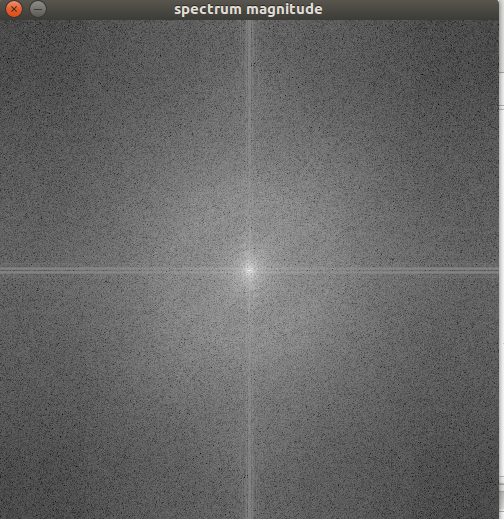
Etant donné, qu'à la première étape, nous avons élargi l'image, il nous faut maintenant recadrer les valeurs nouvellement introduites. À des fins de visualisation, nous pouvons également réorganiser les quadrants du résultat, de sorte que l'origine (zéro, zéro) corresponde au centre de l'image.



et ensuite nous normalisons avec le code suivant:

normaliser ( magI , magI , 0 , 1 , NORM\_MINMAX );

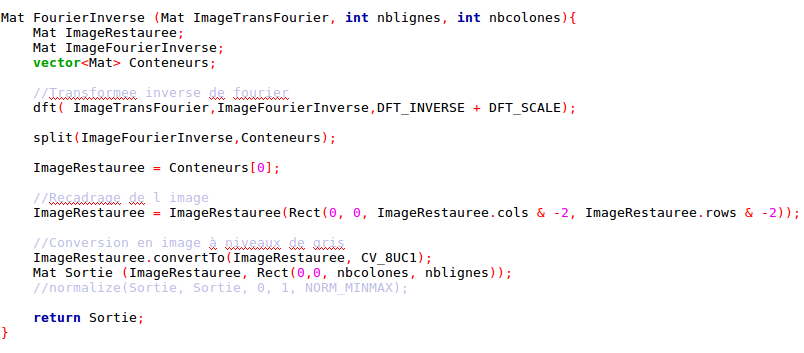
ce qui nous donne l’image suivante:



sur l’image ci-haut on peut facilement observer les bases fréquences ,ces sont les régions où l'intensité est homogène tandis que les hautes fréquences sont remarquées par la perpendiculaire de luminosité intense.

**2. l’inverse de la transformée de Fourier sur la même image**

pour l’inverse de la DFT nous aurons juste à changer un paramètre au niveau de la fonction dft d’openCV :



vous remarquerez le paramètre DFT\_INVERSE qui permet de faire l’inverse de DFT et en voici le résultat:



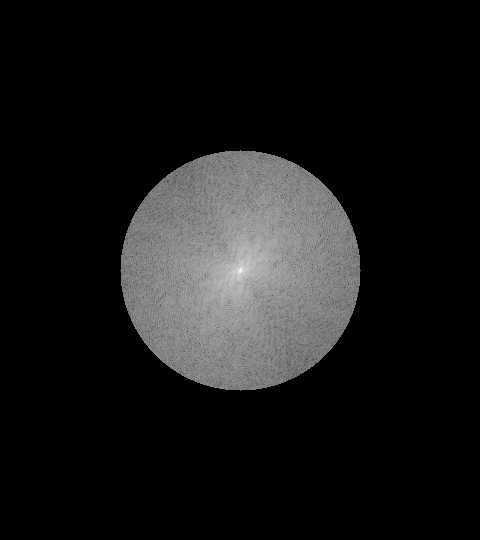
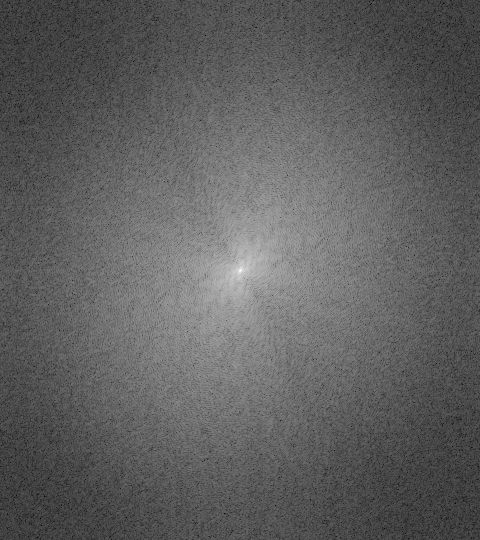
**3. Filtre passe Bas et Filtre passe haut**

1. **Filtre passe bas**

dans les lignes qui suivent nous présenterons le filtre passe bas appliqué à deux images, en effet nous n’allons pas appliquer le filtre directement à ces images mais plutôt à leurs Transformé de Fourier .

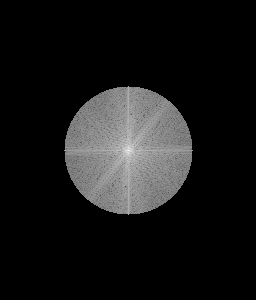
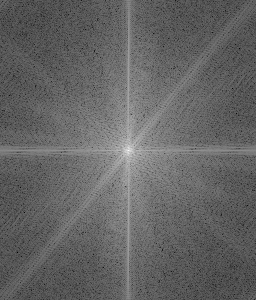
En premier lieu il va falloir définir un rayon de coupure(fréquence de coupure compris entre 0.0 et 1.0 ), c’est en effet ce cercle qui nous aidera à mettre toutes les valeurs de pixels qui ne se trouvent pas dans celui-ci à zéro.

voici notre première image en entré, la seconde qui est la transformée de Fourier et la dernière le filtre passe bas :



pour la troisième image nous voyons très clairement que tous les valeurs de pixels qui sont en dehors du cercle défini par la fréquence de coupure(ici nous avons utilisé une fréquence de 0.5) sont mit à zéro d'où la couleur noir.

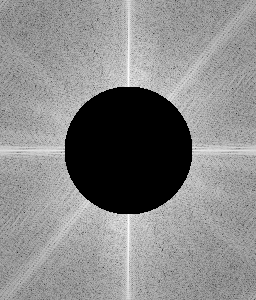
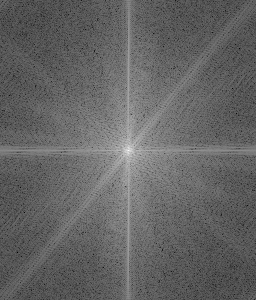
voici un deuxième résultat issue d’une seconde image:



avec les images ci-haut nous avons aussi utilisé une fréquence de coupure de 0.5

**B. Filtre passe Haut**

ce filtre est semblable à celui du filtre passe bas à la seule différence qu’il met à zéro toutes les valeurs de pixels qui sont compris dans le cercle.



**4. Conclusion**

en conclusion nous pouvons dire que nous avons bel et bien tester la transformée de Fourier et que nous l’avons afficher sous une échelle logarithmique et ainsi tester son inverse et ses filtre passe bas et passe haut en utilisant une fréquence de coupure de 0.5.