UNIVERSITÉ DE BOURGOGNE

Outils mathématiques

RAPPORT : TRANSFORMÉE DE FOURIER

Mohamed BOUCHENGUOUR
Mehdi ASNI

TP6

ANNÉE UNIVERISTAIRE 2022/2023

Sommaire

Introduction

La transformée de Fourier discrète est un outil permettant de traiter un signal numérique. Cet outil a de nombreuses applications dans la reconnaissance vocale, l'amélioration de la qualité des images, la transmission numérique, la biomédecine ou encore l'astronomie. Cependant, cet outil a un défaut : sa complexité. Néanmoins, il existe un algorithme que nous allons étudier permettant de la diminuer drastiquement.

Transformée de Fourier discrète 1D

Formule

Tout d'abord, nous allons implémenter la transformation de Fourier discrète directe à une dimension à l'aide de la formule suivante :

$$G(u) = F(g(x)) = \sum_{x=0}^{N-1} g(x) \exp\left(-\frac{2i\pi ux}{N}\right) \; avec \; u = 0..N-1$$

Implémentation

```
function [J] = TFD1D(I)
  [x,y]=size(I);
  J=zeros(1,y);
  for i=1:y
      for j=1:y
        J(1,i)=J(1,i)+I(1,j)*exp(-(2*1i*pi*(i-1)*(j-1))/y);
      endfor
  endfor
endfunction
```

Complexité

Pour chaque élément de notre tableau, nous devons faire une somme de toutes les valeurs de notre tableau. Pour un tableau de taille N, il y aura donc N*N opérations soit une complexité de N^2. Pour traiter un tableau avec un million de valeurs, il faudrait environ 2 heures et 48 minutes pour l'exécuter.

Transformée de Fourier discrète 2D

Formule

Ensuite, nous allons implémenter la transformation de Fourier discrète directe à deux dimensions à l'aide de la formule suivante :

$$F\big(g(x,y)\big) = G(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)e^{-2i\pi(\frac{ux}{M}+\frac{vy}{N})}$$

Implémentation

```
function [J] = TFD2D(I)
  [x,y]=size(I);
  J=zeros(x,y);
  for i=1:x
      for j=1:y
          for k=1:x
              for l=1:y
                  J(i,j)=J(i,j)+I(k,l)*exp(-2*pi*1i*(((i-1)*(k-1)/x)+((j-1)*(l-1)/y)));
              endfor
          endfor
      endfor
  endfor
endfunction
```

Complexité

Cette fonction permet d'avoir la transformée de Fourier discrète d'une image. Cependant, cette fonction à un problème : sa complexité. En effet, pour appliquer la transformée de Fourier sur un pixel, la fonction doit parcourir tous les pixels de l'image. C'est à dire que pour une image ayant N pixels en hauteur et N pixels en largeur, il faudra faire N^4 opérations soit une complexité égale à N^4. Même pour une simple image en 1080p, le temps d'exécution est d'environ 12 heures. Si nous voulons appliquer la transformée de Fourier sur des vidéos (24 images par secondes) ou sur des images de très haute résolutions (4k, image de satellite, images d'astronomie) appliquer cet algorithme serait impossible. Pour pallier ce problème, l'algorithme de la transformée de Fourier rapide existe et nous allons maintenant l'étudier.

La transformée de Fourier discrète 1D rapide

Formule

La transformée de Fourier discrète 1D a pour formule :

$$S(k) = \sum_{n=0}^{N-1} s(n) e^{\frac{-2i\pi nk}{N}} \; avec \; k = 0..N-1$$

$$On \; pose \; W_N^k = e^{\frac{-2i\pi k}{N}} \; avec \; k = 0..N-1$$

$$On \; se \; retrouve \; avec \; une \; forme \; plus \; simplifiée :$$

$$S(k) = \sum_{n=0}^{N-1} s(n) W_N^{nk} \; avec \; k = 0..N-1$$

Ici, on rappelle que la complexité est de N². Pour améliorer ce temps d'exécution, nous allons utiliser l'algorithme de Cooley Tukey.

Nous allons d'abord couper notre somme de N points en 2 sommes contenant chacun N/2 points. La première somme contiendra les valeurs d'indices paires et la deuxième les valeurs d'indices impairs.

Le tableau d'indice pair aura pour formule :

$$s_0(m) = s(2m) \; avec \; m = 0..\frac{N}{2} - 1$$

Le tableau d'indice impair aura pour formule :

$$s_1(m) = s(2m+1) \; avec \; m = 0..\frac{N}{2} - 1$$

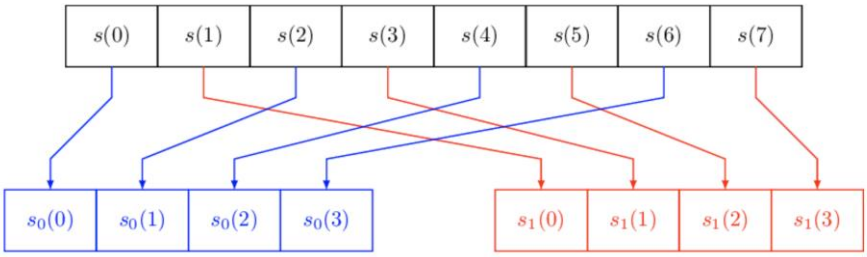

Figure 1- Exemple pour N=8

La transformée de Fourier du tableau s aura pour formule :

$$S(k) = \sum_{m=0}^{\frac{N}{2}-1} s(2m) \, W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} s(2m+1) \, W_N^{(2m+1)k} \; avec \; k = 0..N-1$$

$$W_N^{(2m+1)k} = W_N^{2mk+k} = e^{(\frac{-2i\pi(2mk+k)}{N})} = e^{(\frac{-2i\pi 2mk}{N} + \frac{-2i\pi k}{N})} = e^{(\frac{-2i\pi 2mk}{N})} e^{(\frac{-2i\pi k}{N})} = W_N^{2mk} W_N^k$$

$$donc\ S(k) = \sum_{m=0}^{\frac{N}{2}-1} s(2m)\, W_N^{2mk} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} s(2m+1)\, W_N^{2mk}\ avec\ k = 0..N-1$$

On obtient ainsi la somme de deux transformé de Fourier.
Le calcul de la transformée de Fourier pour N points est donc équivalent au calcul de deux transformées de Fourier ayant chacun N/2 points.

Nous allons désormais analyser la première moitié de la transformée de Fourier S(k) pour les indices k = $0..\frac{N}{2}-1$.

Nous précisons que :

$$W_N^{2mk} = e^{\frac{-2i\pi 2mk}{N}} = e^{\frac{-2i\pi mk}{\frac{N}{2}}} = W_{\frac{N}{2}}^{mk}$$

Alors la transformée de Fourier est sous la forme :

$$S(k) = \sum_{m=0}^{\frac{N}{2}-1} s(2m)\, W_{\frac{N}{2}}^{mk} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} s(2m+1)\, W_{\frac{N}{2}}^{mk}\ avec\ k = 0..\frac{N}{2}-1$$

Or on rappelle que :

$$s(2m) = s_0(m)\ avec\ m = 0..\frac{N}{2}-1$$

$$s(2m+1) = s_1(m)\ avec\ m = 0..\frac{N}{2}-1$$

Et :

$$S_0(k) = \sum_{m=0}^{\frac{N}{2}-1} s_0(m)\, W_{\frac{N}{2}}^{mk}\ avec\ k = 0..\frac{N}{2}-1$$

$$S_1(k) = \sum_{m=0}^{\frac{N}{2}-1} s_1(m)\, W_{\frac{N}{2}}^{mk}\ avec\ k = 0..\frac{N}{2}-1$$

Alors :

$$S(k) = S_0(k) + W_N^k S_1(k)\ avec\ k = 0..\frac{N}{2}-1$$

La première moitié de la transformée de Fourier S(k) est donc égale à la somme de la transformée de Fourier $S_0(k)$ et de la transformée de Fourier $S_1(k)$ multiplié par W_N^k, c'est-à-dire $e^{\frac{-2i\pi k}{N}}$.

Pour la seconde moitié de la transformée de Fourier, c'est-à-dire S(k+ $\frac{N}{2}$) avec k = 0..$\frac{N}{2}-1$, nous avons :

$$S\left(k+\frac{N}{2}\right)=\sum_{m=0}^{\frac{N}{2}-1}s(2m)W_N^{2m(k+\frac{N}{2})}+\sum_{m=0}^{\frac{N}{2}-1}s(2m+1)W_N^{(2m+1)(k+\frac{N}{2})}$$

Développons $W_N^{2m(k+\frac{N}{2})}$:

$$W_N^{2m(k+\frac{N}{2})}=W_N^{2mk}W_N^{2m\frac{N}{2}}$$

Nous avons vu précédemment que : $W_N^{2mk}=W_{\frac{N}{2}}^{mk}$

$$W_N^{2m\frac{N}{2}}=e^{\frac{-2i\pi 2m\frac{N}{2}}{N}}=e^{\frac{-2i\pi mN}{N}}=e^{-2i\pi m}=1$$

$$W_N^{2m(k+\frac{N}{2})}=W_{\frac{N}{2}}^{mk}$$

Développons $W_N^{(2m+1)(k+\frac{N}{2})}$:

$$W_N^{(2m+1)(k+\frac{N}{2})}=W_N^{2mk}\,W_N^{2m\frac{N}{2}}\,W_N^{k}\,W_N^{\frac{N}{2}}$$

Nous avons vu précédemment que : $W_N^{2mk}=W_{\frac{N}{2}}^{mk}$ et $W_N^{2m\frac{N}{2}}=1$

$$W_N^{\frac{N}{2}}=e^{\frac{-2i\pi\frac{N}{2}}{N}}=e^{\frac{-i\pi N}{N}}=e^{-i\pi}=-1$$

$$W_N^{(2m+1)(k+\frac{N}{2})}=-W_N^{k}W_{\frac{N}{2}}^{mk}$$

Nous avons donc :

$$S\left(k+\frac{N}{2}\right)=\sum_{m=0}^{\frac{N}{2}-1}s(2m)W_N^{2m(k+\frac{N}{2})}+\sum_{m=0}^{\frac{N}{2}-1}s(2m+1)W_N^{(2m+1)(k+\frac{N}{2})}\ avec\ k=0..\frac{N}{2}-1$$

$$=S\left(k+\frac{N}{2}\right)=\sum_{m=0}^{\frac{N}{2}-1}s(2m)W_{\frac{N}{2}}^{mk}-W_N^{k}\sum_{m=0}^{\frac{N}{2}-1}s(2m+1)W_{\frac{N}{2}}^{mk}\ \ avec\ k=0..\frac{N}{2}-1$$

On rappelle que :

$$S_0(k)=\sum_{m=0}^{\frac{N}{2}-1}s_0(m)\,W_{\frac{N}{2}}^{mk}\ avec\ k=0..\frac{N}{2}-1$$

$$S_1(k) = \sum_{m=0}^{\frac{N}{2}-1} s_1(m)\, W_{\frac{N}{2}}^{mk} \; avec \; k = 0..\frac{N}{2} - 1$$

Donc :

$$S\left(k + \frac{N}{2}\right) = S_0(k) - W_N^k S_1(k) \; avec \; k = 0..\frac{N}{2} - 1$$

La deuxième moitié de la transformée de Fourier S(k) est donc égale à la différence de la transformée de Fourier $S_0(k)$ et de la transformée de Fourier $S_1(k)$ multiplié par W_N^k, c'est-à-dire $e^{\frac{-2i\pi k}{N}}$.

Nous pouvons finalement écrire :

$$S(k) = S_0(k) + W_N^k S_1(k) \; avec \; k = 0..\frac{N}{2} - 1$$

$$S\left(k + \frac{N}{2}\right) = S_0(k) - W_N^k S_1(k) \; avec \; k = 0..\frac{N}{2} - 1$$

Nous allons appliquons cette même procédure récursivement pour chaque tableau à $\frac{N}{2}$ points. C'est-à-dire que le tableau pair et le tableau impair de $\frac{N}{2}$ points seront coupés en 2, nous allons donc avoir 4 tableaux avec $\frac{N}{4}$ points et ainsi de suite. Le tableau s_{00} contiendra les éléments d'indice pair du tableau s_0, le tableau s_{01} contiendra les éléments d'indice impair du tableau s_0, le tableau s_{10} contiendra les éléments d'indice pair du tableau s_1 et le tableau s_{11} contiendra les éléments d'indice impair du tableau s_1.

En effet, nous venons de voir que :

$$S(k) = \sum_{n=0}^{N-1} s(n) e^{\frac{-2i\pi nk}{N}} \; avec \; k = 0..N - 1$$

$$S(k) = S_0(k) + W_N^k S_1(k) \; avec \; k = 0..\frac{N}{2} - 1$$

$$S\left(k + \frac{N}{2}\right) = S_0(k) - W_N^k S_1(k) \; avec \; k = 0..\frac{N}{2} - 1$$

S_0 :

$$S_0(k) = \sum_{m=0}^{\frac{N}{2}-1} s_0(m)\, W_{\frac{N}{2}}^{mk} \; avec \; k = 0..\frac{N}{2} - 1$$

$$S_0(k) = S_{00}(k) + W_{\frac{N}{2}}^k S_{01}(k) \; avec \; k = 0..\frac{N}{4} - 1$$

$$S_0\left(k + \frac{N}{2}\right) = S_{00}(k) - W_{\frac{N}{2}}^k S_{01}(k) \; avec \; k = 0..\frac{N}{4} - 1$$

$$s_{00}(m) = s_0(2m) \; avec \; m = 0..\frac{N}{4} - 1$$

$$s_{01}(m) = s_0(2m+1) \; avec \; m = 0..\frac{N}{4} - 1$$

S_1 :

$$S_1(k) = \sum_{m=0}^{\frac{N}{2}-1} s_1(m)\, W_{\frac{N}{2}}^{mk} \; avec \; k = 0..\frac{N}{2} - 1$$

$$S_1(k) = S_{10}(k) + W_{\frac{N}{2}}^{k} S_{11}(k) \; avec \; k = 0..\frac{N}{4} - 1$$

$$S_1\left(k + \frac{N}{2}\right) = S_{10}(k) - W_{\frac{N}{2}}^{k} S_{11}(k) \; avec \; k = 0..\frac{N}{4} - 1$$

$$s_{10}(m) = s_1(2m) \; avec \; m = 0..\frac{N}{4} - 1$$

$$s_{11}(m) = s_1(2m+1) \; avec \; m = 0..\frac{N}{4} - 1$$

Nous réalisons cette opération jusqu'à avoir N tableaux de taille 1. La transformée de Fourier d'un tableau de 1 point est égale à lui-même car :

$$S(k) = \sum_{n=0}^{N-1} s(n) e^{\frac{-2i\pi nk}{N}}$$

$$S(0) = s(0) e^{\frac{-2i\pi 0*0}{N}} = s(0) e^{0} = s(0)$$

Implémentation

```
function [J] = TFR1D(I)
  [x,y]=size(I);
  if(y!=1)
      for i=1:(y/2)
          pair(i)=I(2*i-1);
          impair(i)=I(2*i);
      endfor
      pair=TFR1D(pair);
      impair=TFR1D(impair);
      for i=1:(y/2)
          J(i)=pair(i)+exp((-2*1i*pi*(i-1))/y)*impair(i);
          J(i+(y/2))=pair(i)-exp((-2*1i*pi*(i-1))/y)*impair(i);
      endfor
  else
    J = I;
  endif
endfunction
```

On récupère d'abord la taille du tableau passé en paramètre :

```
function [J] = TFR1D(I)

  [x,y]=size(I);
```

Si la taille du tableau est plus grande que 1 :

- On affecte à un tableau les valeurs d'indices pairs et à un deuxième tableau les valeurs d'indices impairs du tableau passé en paramètre de la fonction.

```
for i=1:(y/2)
    pair(i)=I(2*i-1);
    impair(i)=I(2*i);
endfor
```

- Ensuite, on appelle récursivement la fonction sur nos deux tableaux avec en paramètre leur tableau respectif. Le tableau pair contiendra la transformée de Fourier du tableau d'indices pair et le tableau impair contiendra la transformée de Fourier du tableau d'indices impair.

```
pair=TFR1D(pair);
impair=TFR1D(impair);
```

- On applique ensuite la formule vue précédemment.

```
for i=1:(y/2)
    J(i)=pair(i)+exp((-2*1i*pi*(i-1))/y)*impair(i);
    J(i+(y/2))=pair(i)-exp((-2*1i*pi*(i-1))/y)*impair(i);
endfor
```

Sur Octave, les tableaux commencent à l'indice 1. Nous devons donc décrémenter la valeur de j pour avoir le bon résultat.

Sinon, si la taille du tableau est de 1 :

```
else
  J = I;
endif
```

On renvoie le tableau tel qu'il est car la transformée de Fourier d'un tableau à une valeur est égale à sa propre valeur.

Complexité

Dans cet algorithme, nous savons que notre tableau de N points sera découpé récursivement en 2 jusqu'à avoir N tableau de taille 1. Or, N est égale à 2^n. Il y aura donc n découpage soit $log_2(N)$. Après avoir obtenu N tableau de taille 1, nous devons appliquer la formule afin de retrouver notre tableau à N points. Sachant qu'il y a $log_2(N)$ découpage, il y aura $log_2(N)$ étape pour recombiner le tableau à N points. A chaque étape, nous devons donc calculer N valeurs. Sachant qu'il y a $log_2(N)$ étape pour recombiner le tableau et que dans chaque étape il y aura N opérations, la complexité de l'algorithme est de $N * log_2(N)$. Avec un tableau ayant environ un million de valeurs, on passe d'un temps d'exécution d'environ 2 heures et 48 minutes à environ 210 millisecondes soit environ 48000 fois plus rapide que l'algorithme de base.

La transformée de Fourier discrète inverse 1D rapide

Formule

La transformée de Fourier discrète inverse d'un tableau à une dimension à pour formule :

$$s(k) = \frac{1}{N}\sum_{n=0}^{N-1} S(n)e^{\frac{2i\pi nk}{N}} \; avec \; k = 0..N-1$$

La transformée de Fourier inverse est très similaire à la transformée de Fourier à deux détails près : la valeur de l'exponentielle est positive et la somme est divisé par N, le nombre de valeurs.

Nous pouvons donc écrire que :

$$s(k) = \frac{1}{N}\sum_{n=0}^{N-1} S(n)e^{\frac{2i\pi nk}{N}} \; avec \; k = 0..N-1$$

$$On \; pose \; W_N^k = e^{\frac{2i\pi k}{N}} \; avec \; k = 0..N-1$$

$$s(k) = \frac{1}{N}\sum_{n=0}^{N-1} S(n)W_N^k \; avec \; k = 0..N-1$$

$$s(k) = \frac{1}{N}(s_0(k) + W_N^k s_1(k)) \; avec \; k = 0..\frac{N}{2} - 1$$

$$s\left(k + \frac{N}{2}\right) = \frac{1}{N}(s_0(k) - W_N^k s_1(k)) \; avec \; k = 0..\frac{N}{2} - 1$$

Nous allons donc appliquer le même algorithme en changeant le signe de l'exponentielle en positif et en divisant chaque élément du tableau s final par le nombre de points.

Implémentation

```
function [J] = TFRI1D(I)
  [x,y]=size(I);
  if(y!=1)
      for i=1:(y/2)
          pair(i)=I(2*i-1);
          impair(i)=I(2*i);
      endfor
      pair=TFRI1D(pair);
      impair=TFRI1D(impair);
      for i=1:(y/2)
          J(i)=(pair(i)+exp((2*1i*pi*(i-1))/y)*impair(i))/2;
          J(i+(y/2))=(pair(i)-exp((2*1i*pi*(i-1))/y)*impair(i))/2;
      endfor
  else
    J = I;
    endif
endfunction
```

Ici, nous pouvons voir que nous avons bien changé la valeur de l'exponentielle en positif.

```
for i=1:(y/2)
    J(i)=(pair(i)+exp((2*1i*pi*(i-1))/y)*impair(i))/2;
    J(i+(y/2))=(pair(i)-exp((2*1i*pi*(i-1))/y)*impair(i))/2;
endfor
```

Afin d'optimiser l'algorithme pour ne pas diviser chaque valeur du tableau par N à la fin de l'appel de la fonction, nous divisons les valeurs par deux à chaque étape de recomposition. Comme il y a $log_2(N)$ étape pour recomposer le tableau, une valeur dépend de $log_2(N)$ étapes. Donc si nous divisons par deux $log_2(N)$ fois une valeur, la valeur sera divisée par $2^{log_2(N)}$ soit N.

La transformée de Fourier discrète 2D rapide

Explication de l'algorithme

La transformée de Fourier discrète 2D rapide est très simple à implémenter une fois que nous avons l'algorithme de la transformée de Fourier discrète 1D rapide. Effectivement, pour avoir la transformée de Fourier rapide discrète d'un tableau à deux dimensions, nous appliquons sur chaque ligne la transformée de Fourier discrète 1D rapide que nous stockons dans une nouvelle matrice puis nous appliquons la transformée de Fourier 1D rapide sur chaque colonne de cette matrice.

Implémentation

```
function [J] = TFR2D(I)

[x,y]=size(I);

for i=1:x
  J(i, :)=TFR1D(I(i, :));
endfor

for i=1:y
  J(:, i) = flip(TFR1D(J(:, i)'));
endfor

J = circshift(J, 1);
endfunction
```

```
function [J] = TFR2D(I)
```

On récupère la taille de la matrice passée en paramètre.

```
[x,y]=size(I);
```

```
for i=1:x
  J(i, :)=TFR1D(I(i, :));
endfor
```

On applique la transformée de Fourier rapide discrète 1D sur chaque ligne. On stocke ces nouvelles lignes dans une matrice J.

```
for i=1:y
  J(:, i) = flip(TFR1D(J(:, i)'));
endfor
```

On applique ensuite la transformée de Fourier sur chaque colonne de la nouvelle matrice J. Pour ceci, on applique la transposée sur une colonne, on passe la ligne en paramètre de la fonction TFR1D puis nous appliquons la transposée sur le résultat de la fonction. On inverse ensuite la colonne.

```
J = circshift(J, 1);
```

On décale ensuite notre matrice d'une ligne par le haut.

Complexité

Pour avoir la transformée de Fourier discrète rapide d'une matrice ayant N lignes et N colonnes, nous allons appliquer 2*N transformée de Fourier rapide à une dimension. Nous aurons donc $2 * N^2 * log_2(N)$ opérations à faire soit une complexité égale à $N^2 * log_2(N)$. Pour une image ayant 2048 pixels en largeur et 1024 pixels en hauteurs (environ 1080p), on passe d'un temps d'exécution de 12 heures à 440 millisecondes soit environ 97000 fois plus rapide.

La transformée de Fourier discrète inverse 2D rapide

Explication de l'algorithme

La transformée de Fourier inverse 2D rapide fonctionne comme la transformée de Fourier rapide 2D. Nous appliquons donc sur chaque ligne la transformée de Fourier rapide inverse 1D que nous stockons dans une nouvelle matrice puis nous appliquons la transformée de Fourier inverse 1D rapide sur chaque colonne de cette nouvelle matrice.

Implémentation

```
function [J] = TFRI2D(I)

[x,y]=size(I);

for i=1:x
  J(i, :)=TFRI1D(I(i, :));
endfor

for i=1:y
  J(:, i) = flip(TFRI1D(J(:, i)')');
endfor

J = circshift(J, 1);
endfunction
```

```
function [J] = TFRI2D(I)
```

On récupère la taille de la matrice passée en paramètre.

```
[x,y]=size(I);
```

```
for i=1:x
  J(i, :)=TFRI1D(I(i, :));
endfor
```

On applique la transformée de Fourier rapide inverse 1D sur chaque ligne. On stocke ces nouvelles lignes dans une matrice J.

```
for i=1:y
  J(:, i) = flip(TFRI1D(J(:, i)')');
endfor
```

On applique ensuite la transformée de Fourier sur chaque colonne de la nouvelle matrice J. Pour ceci, on applique la transposée sur une colonne, on passe la ligne en paramètre de la fonction TFR1D puis nous appliquons la transposée sur le résultat de la fonction. On inverse ensuite la colonne.

```
J = circshift(J, 1);
```

Puis on décale notre matrice d'une ligne par le haut.

Conclusion

L'algorithme de la transformée de Fourier rapide a permis de révolutionner le domaine du numérique en diminuant de manière significatif la complexité de l'algorithme de la transformée de Fourier. De plus, cet algorithme permet d'avoir des résultats plus précis car en ayant beaucoup moins d'opérations à faire, il y aura beaucoup moins d'arrondis des nombres à virgules. Cependant, l'algorithme de la transformée de Fourier n'est pas parfait. En effet, cet algorithme s'applique uniquement sur des images dont le nombre de pixel en largeur et en hauteur sont des puissances de deux.