Despret Nancy
Gaudebert Adrian

Licence 3 Informatique, semestre 1

Projet de Mathématiques

Transformée de Fourier

Outils mathématiques pour l'informatique

Année 2008/2009

Sommaire

Introduction

Ce document s'inscrit dans le cadre du projet d'outils mathématiques pour l'informatique du premier semestre de la troisième année de licence informatique. L'objectif de ce projet est de programmer la transformée de Fourier pour une image de différentes manières.

Nous avons choisi d'utiliser le logiciel Scilab pour programmer et tester nos algorithmes, dans un soucis de simplicité, mais aussi parce que Scilab fourni une fonction de calcul de la transformée de Fourier, ce qui nous a permis de tester nos propres fonctions.

1. Transformée de Fourier directe discrète 2D

Le premier algorithme est celui de la transformée de Fourier directe discrète pour une image en deux dimensions. On utilisera la formule dans l'ensemble des nombres complexes pour calculer cette transformée :

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

Code:

```
// fonction tfdd
// Calcule la transformée de Fourier directe discrète, en 2 dimensions.
// @param Img Image à partir de laquelle la transformée de Fourier est
calculée. L'image doit être en niveau de gris.
// @return ImgTrans Résultat de la transformée de Fourier appliquée à
l'image Img.
function [ImgTrans] = tfdd(Img)

    // On récupère la taille de l'image passée en paramètre
    [Height, Width] = size(Img)

    // On créé l'image de retour, de même taille que l'image passée
en paramètre, et on l'initialise à 0
    ImgTrans = zeros(Height, Width)

    // On boucle pour chaque pixel de l'image résultat
    for u = 1 : Height

        for v = 1 : Width

            // On boucle pour chaque pixel de l'image passée en
paramètre
            for x = 1 : Height

                for y = 1 : Width

                    // On applique la formule de la transformée de
Fourier directe discrète sur le pixel traité
                    ImgTrans(u, v) = ImgTrans(u, v) + Img(x, y) *
exp( ( ( -2 * %i * %pi )  ) * ( ( ( u - 1 ) * ( x - 1 ) / Height ) +
( ( v - 1 ) * ( y - 1 ) / Width ) ) )

                end

            end

        end

    end

endfunction
```

2. Transformée de Fourier rapide discrète 1D (et inverse)

Le deuxième algorithme est celui de la transformée de Fourier rapide discrète pour une image à une dimension. La formule mathématique est la suivante :

$$\hat{I}(u) = \sum_{k=0}^{N-1} I(k) e^{-\frac{2i\pi ku}{N}}$$

La transformée de Fourier rapide nécessite que les données soient préparées : il est nécessaire d'inverser les pixels de l'image avant de calculer la transformée, et il faut ensuite ré-inverser les pixels de l'image résultant de la transformée. Afin de faire ces traitements sur les données, nous avons choisi de créer deux fonctions, la première préparant les données pour les envoyer à la deuxième, laquelle effectue le calcul de la transformée de Fourier rapide grâce à la méthode de Cooley-Tukey.

Le calcul de la transformée de Fourier est un calcul récursif : on va séparer les pixels pairs et impairs de chaque image jusqu'à ce qu'on ait plus que des points. On va ensuite reconstruire l'image transformée en assemblant les résultats de la formule de la transformée de Fourier sur chaque couple de pixels. Le schéma suivant montre le déroulement de l'algorithme :

```
                c = coefficient de calcul de la transformée

                              [ Image de base ]
                               1 2 3 4 5 6 7 8

                          1 3 5 7          2 4 6 8

                     1 5          3 7          2 6          4 8

  1          5          3          7          2          6          4          8

A=1+c*5 B=1-c*5      C=3+c*7 D=3-c*7      E=2+c*6 F=2-c*6      G=4+c*8 H=4-c*8

    I=A+c*C J=B+c*D K=A-c*C L=B-c*D      M=E+c*G N=F+c*H O=E-c*G P=F-c*H

    O=I+c*M P=J+c*N Q=K+c*H R=L+c*P S= I-c*M T= J-c*N U= K-c*H V= L-c*P

                               O P Q R S T U V
                          [Image résultat de la FFT1D ]
```

Déroulement de l'algorithme de la transformée de Fourier rapide 1D

Code :

```
// fonction fft1d
// Prépare les données pour le calcul de la transformée de Fourier en
une dimension, puis lance le calcul
// @param Img Image à partir de laquelle la transformée de Fourier est
calculée. L'image doit être en niveau de gris, et de taille (1, 2^n)
// @param SensFFT Détermine le sens de la transformation. Directe : -1;
Inverse : 1.
// @return ImgTrans Résultat de la transformée de Fourier appliquée à
l'image Img.
function [ImgTrans] = fft1d(Img, SensFFT)

    // Calcul de la taille de l'image
    Width = size(Img, 2)

    // Création d'une image temporaire pour l'inversion de l'image de
base
    ImgTemp1 = zeros(1, Width)

    // Inversion des pixels de l'image passée en paramètre
    for i = 1 : Width

        ImgTemp1(i) = Img(Width + 1 - i)

    end

    // Calcul de la FFT1D sur l'image retournée
    ImgTemp2 = _fft1d(ImgTemp1, SensFFT)

    // Initialisation de l'image de retour
    ImgTrans = zeros(1, Width)

    // Inversion du résultat de la transformée de Fourier
    for i = 1 : Width

        ImgTrans(i) = ImgTemp2(Width + 1 - i)

    end

    // Si on est en sens inverse, il faut diviser chaque pixel du
tableau par la taille du tableau
    if SensFFT == 1 then

        for i = 1 : Width

            ImgTrans(i) = ImgTrans(i) / Width

        end

    end

endfunction
```

Code :

```
// fonction _fft1d
// Calcule la transformée de Fourier rapide pour 1 dimension
// @param Img Image à partir de laquelle la transformée de Fourier est
calculée. L'image doit être en niveau de gris.
// @param SensFFT Détermine le sens de la transformation. Directe : -1;
Inverse : 1.
// @return ImgTrans Résultat de la transformée de Fourier appliquée à
l'image Img.
function [ImgTrans] = _fft1d(Img, SensFFT)

    // On calcule la taille de l'image passée en paramètre
    Width = size(Img, 2);

    // On créé la matrice qui sera retournée
    ImgTrans = Img

    // Si la Width est différent de 1, alors on traite l'image en
paramètre
    if Width <> 1 then

        // On créé les matrices
        MPair = zeros(1, Width / 2)    // Pixels de position paire
        MImpair = zeros(1, Width / 2)  // Pixels de position impaire

        // On coupe en deux l'image reçue, pour traiter indépendamment
chaque moitié
         for i = 1 : Width
              if modulo(i, 2) == 0 then
                    // Pair
                    MPair(i / 2) = Img(i)
              else
                    // Impair
                     MImpair( ( i + 1 ) / 2 ) = Img(i)
              end
         end

         // On appelle la fonction pour les deux nouvelles images
         ImgPair = _fft1d(MPair, SensFFT)
         ImgImpair = _fft1d(MImpair, SensFFT)

         // On applique la formule de la transformée de fourier pour
les images obtenues
         for u = 1 : (Width / 2)

                coef = exp( ( SensFFT * 2 * %i * %pi * u ) / Width )
                ImgTrans(u) = ImgPair(u) + ( coef * ImgImpair(u) )
                ImgTrans( u + ( Width / 2 ) ) = ImgPair(u) - ( coef *
ImgImpair(u) )

         end

    end

endfunction
```

Le calcul de la transformée de Fourier rapide inverse est intégré à la fonction. Il y a donc un paramètre *SensFFT* qui indique si la transformée est directe (*SensFFT = -1*) ou inverse (*SensFFT = 1*). L'algorithme de la transformée de Fourier rapide inverse est identique à celui de la transformée directe, à deux exceptions près : le signe du coefficient de calcul est opposé (d'où le paramètre *SensFFT* qui vaut soit -1 soit 1), et dans le cas de la transformée inverse, tous les pixels doivent être divisés par la taille de l'image après le calcul de la transformée.

3. Transformée de Fourier rapide discrète 2D (et inverse)

La transformée de Fourier rapide appliquée sur une image en deux dimensions de fait simplement en appliquant la transformée pour une dimension sur chaque ligne de l'image, puis sur chaque colonne de l'image.

La transformée de Fourier rapide inverse 2D utilise simplement la transformée rapide 1D inverse, il suffit de donc de passer un paramètre *SensFFT* à la fonction.

Code :

```
// fonction fft2d
// Calcule la transformée de Fourier rapide pour 2 dimensions
// @param Img Image à partir de laquelle la transformée de Fourier est
calculée. L'image doit être en niveau de gris, et de taille (2^n, 2^n)
// @param SensFFT Détermine le sens de la transformation. Directe : -1;
Inverse : 1.
// @return ImgTrans Résultat de la transformée de Fourier appliquée à
l'image Img.
function [ImgTrans] = fft2d(Img,SensFFT)

      // Dimensions de l'image de la taille d'une ligne de l'image
      [Height, Width] = size(Img)

    // On initialise l'image résultat
      ImgTrans = zeros(Height, Width)

      // Matrice à une ligne de la taille d'une ligne de l'image
      ImgWork = zeros(1, Width)

      // On traite chaque ligne de l'image
      for iRow = 1 : Height

          // On recopie la ligne courante dans la matrice de travail
          for iColumn = 1 : Width

              ImgWork(iColumn) = Img(iRow, iColumn)

          end

          // On applique la transformée de Fourier 1D sur la ligne
courante
          ImgWork = fft1d(ImgWork, SensFFT)

          // On recopie la ligne de travail, résultat de la
transformée, dans l'image résultat
          for iColumn = 1 : Width

              ImgTrans(iRow, iColumn) = ImgWork(iColumn)

          end

      end

      // Matrice à une ligne de la taille d'une colonne de l'image
      ImgWork = zeros(1, Height)

      // On traite chaque colonne de l'image
      for iColumn = 1 : Width

          // On recopie la colonne courante dans la matrice de travail
          for iRow = 1 : Height

              ImgWork(iRow) = ImgTrans(iRow, iColumn)

          end
```

```
        // On applique la transformée de Fourier 1D sur la colonne
courante
        ImgWork = fft1d(ImgWork, SensFFT)

        // On recopie la colonne de travail, résultat de la
transformée, dans l'image résultat
        for iRow = 1 : Height

            ImgTrans(iRow, iColumn) = ImgWork(iRow)

        end
    end

endfunction
```

Bibliographie

[1] Cours sur l'algorithme de la Transformée de Fourier rapide
http://www.polytech.unice.fr/~leroux/courssignal/node70.html

[2] Codes sources pour Scilab d'un ancien étudiant
http://chatvache.info/jonathan/index.php?post/2007/01/11/Scilab-%3A-Transformee-de-Fourier-2D-et-FFT-2D

[3] Cours d'Outils mathématiques pour l'informatique
http://www.u-bourgogne.fr/LE2I/jl.baril/Licence3.html

[4] Autres sources trouvées sur le net
http://www.google.fr