Transformations dans le domaine fréquentiel

Objectifs :
- manipulation des transformées de Fourier directe et inverse ;
- utilisation de la bibliothèque `fftw3` ;
- mise en oeuvre d'une transformation de Fourier discrète ;
- manipulation des spectres d'amplitude et de phase.

1 La bibliothèque `fftw3`

Interfaces :

```
#include <complex.h>
#include <fftw3.h>
```

Édition des liens : `-lfftw3 -lm`

Types :

- `fftw_complex` : type codant un nombre complexe. Correspond au type `complex` de `c99` :
  ```
  fftw_complex c  = a + I*b;
  double       re = creal(c);
  double       im = cimag(c);
  ```
- `fftw_plan` : structure de données utilisée pour paramétrer et mettre en œuvre une transformation directe ou inverse.

Fonctions :

- `fftw_plan *fftw_plan_dft_2d(height, width, input, output, dir, FFTW_ESTIMATE)` : création et initialisation d'un `plan` pour mette en œuvre une transformation complexe, directe ou inverse selon que le paramètre `dir` vaut `FFTW_FORWARD` ou `FFTW_BACKWARD`.
- `void fftw_execute(plan)` : mise en œuvre de la transformation associée à `plan`.
- `void fftw_destroy_plan(plan)` : libération d'un `plan` créé par `fftw2d_create_plan`.

2 Mise en œuvre des transformées discrètes de Fourier

2.1 Transformée directe

Rappel : pour une fonction en 1D f de taille N, la transformée de Fourier directe TF de f est en un point x est

$$TF(f(x))(u) = \sum_{x=0}^{N-1} f(x) \exp(-i2\pi ux/N) \tag{1}$$

avec $u \in [0, \ldots, N[$ les points dans le domaine fréquentiel.

Voici les différentes étapes nécessaires afin de mettre en œuvre une transformée directe :

1. Construction d'une image complexe à partir d'une image source monochrome (en niveaux de gris). L'image complexe est un vecteur de taille : `width*height*sizeof(fftw_complex)`.
   La partie réelle de chaque élément de l'image est initialisée avec le niveau de gris de l'image source, et la partie imagimaire avec 0.
2. Allocation d'une structure de donnée (ayant la même taille que l'image source complexe) afin de recevoir le résultat de la transformée directe.
3. Initialisation et calcul de la transformée directe :

```
fft_plan plan = fftw_plan_dft_2d(height, width,
                                 spatial_repr, frequency_repr,
                                 FFTW_FORWARD, FFTW_ESTIMATE);
fftw_execute(plan);
```

4. Libération mémoire des données intermédiaires (image complexe, plan).

2.2 Transformée inverse

Rappel : pour une fonction en 1D F de taille N, la transformée de Fourier inverse TF^{-1} de f est en un point u du domaine fréquentiel est

$$f(x) = TF^{-1}(F(u))(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) \exp(i2\pi ux/N) \tag{2}$$

avec $x \in [0, \ldots, N[$ les points dans le domaine spatial.

Voici les différentes étapes nécessaires afin de mettre en œuvre une transformée inverse :

1. Allocation d'une structure de donnée afin de recevoir le résultat de la transformée inverse.
2. Initialisation et calcul de la transformée inverse.
3. Extraction de la partie réelle I^R de la transformée pour obtenir la représentation spatiale.
4. Libération des données intermédiaires.

3 Travail à réaliser

3.1 Matériel

– Vérifier que la bibliothèque `fftw3` est bien installée sur votre machine. La documentation en ligne se trouve ici : `http://www.fftw.org/fftw3_doc/`.

– Récupérer l'archive `td-fft.tgz` et décompressez l'archive dans votre répertoire `project/src/`. À l'heure actuelle, votre arborescence doit ressembler à cela :

```
project +-- src  +-- bcl +-- ...
                 |-- td-fft +-- Makefile
                 |          |-- fft.h
                 |          |-- test_fft_module.c
                 |          |-- lena.ppm
                 |-- td-pnm +-- ...
```

L'archive contient : un `Makefile` qui devra être modifier (emplacement de la bibliothèque `fftw3`, édition des liens, ...) ; le fichier entête `fft.h` avec la définition des fonctions à programmer ; un squelette de programme (`test_fft_module.c`) qui vous permettra de tester votre module ; une image de test (`lena.ppm`).

3.2 Module FFT

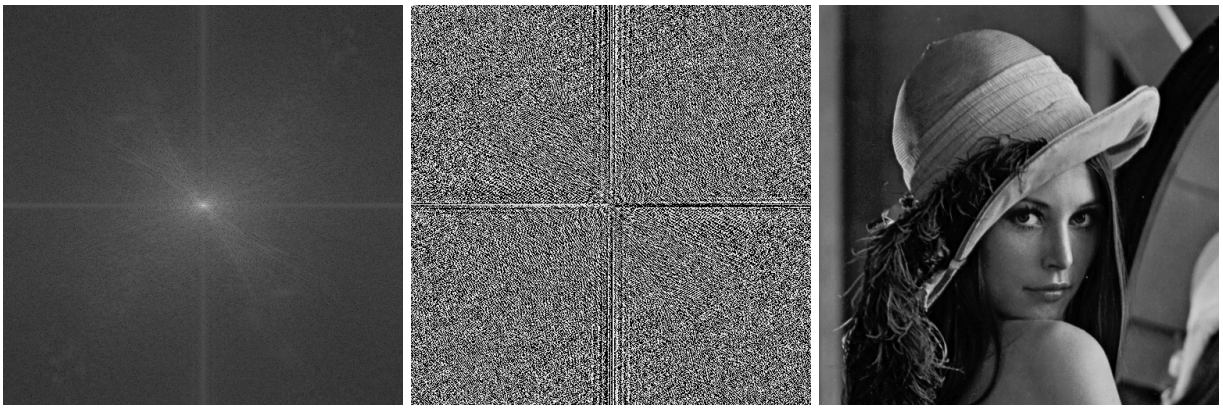
– Implanter dans un module en `C` (`fft.c`) les deux fonctions suivantes :
`fftw_complex *fft_forward(int width, int height, unsigned short* image);`
`unsigned short *fft_backward(int width, int height, fftw_complex* freq_repr);`
qui permettent de mettre en œuvre une transformée de Fourier directe et inverse sur une image monochrome.

3.3 Programme de tests

– Implanter dans le programme de test `test_fft_module.c` la fonction `void test_for_backward(pnm ims, char* name)` permettant de tester l'enchaînement et le bon fonctionnement d'une transformation directe puis inverse. `name` est le nom de l'image en entrée. Si vous utilisez `lena.ppm`, le résultat de votre fonction de test pourra par exemple produire une image nommée `FB_lena.ppm`.

3.4 Spectres d'amplitude (as) et de phase (ps)

- Écrire la fonction :
  `void fft_fr_to_spectra(int w, int h, fftw_complex* freq_repr, float* as, float* ps)`
  qui permet, à partir d'une représentation fréquentielle (`freq_repr`), de calculer le spectre d'amplitude (`as`) et de phase (`ps`). Vous pouvez utiliser les fonctions standards (`math.h`) `hypot` et `atan2`
- Écrire la fonction :
  `void fft_spectra_to_fr(int w, int h, float* as, float* ps, fftw_complex* freq_repr)`
  qui permet, à partir des spectres d'amplitude (`as`) et de phase (`ps`), de calculer une représentation fréquentielle (`freq_repr`).
- Mettre à jour votre programme de test (`test_fft_module`) en écrivant la fonction `void test_spectrum(pnm ims, char* name)` qui permet de tester les deux dernières fonctions codées. Cette fonction de test produit une image d'amplitude, une image de phase et une image reconstruite à partir des spectres d'amplitude et de phase.
  Si vous utilisez l'image fournie (`lena.ppm`), vous devez obtenir des résultats semblables à la figure 1.



(a) `AS_lena.ppm` (b) `PS_lena.ppm` (c) `FB_ASPS_lena.ppm`

FIGURE 1 – Images à obtenir à partir de `lena.ppm`. (a) : spectre d'amplitude ; (b) : spectre de phase ; (c) : image reconstruite à partir de (a) et (b).

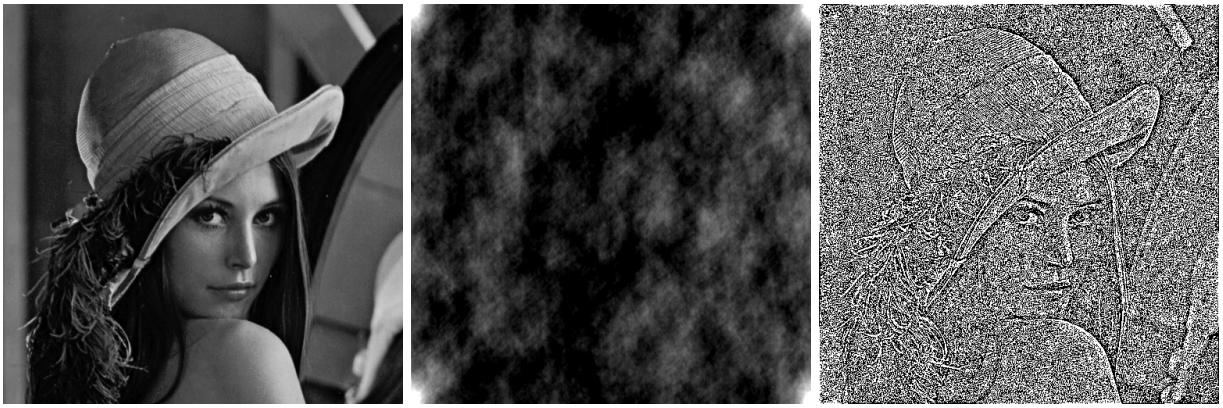
3.5 Modification des spectres et reconstruction

Mettre à jour votre programme de test (`test_fft_module`) en écrivant la fonction
`void test_modification(pnm ims, char* name)`
qui :

- Reconstruit une image à partir du spectre d'amplitude initial `as` mais avec des valeurs de phase aléatoires comprises dans $[0, \max\{\texttt{ps}\}]$.
- Reconstruit une image à partir du spectre de phase initial (`ps`) mais avec un spectre d'amplitude constant ($\max\{\texttt{as}\}$).
- La figure 2 montre les images reconstruites à obtenir si vous utilisez l'image fournie.
- Quelles conclusions pouvez tirer vis à vis des informations contenues dans les spectres d'amplitude et de phase ?

4 Bonus

L'ensemble des programmes que vous avez réalisé ne traite que des images en niveaux de gris. Proposer une modification de l'ensemble des programmes que vous avez réalisé pour le traitement des images couleurs.



(a) `lena.ppm` (b) `RPS_lena.ppm` (c) `CAS_lena.ppm`

FIGURE 2 – Images de reconstruction à obtenir à partir de `lena.ppm`. (a) : image initiale ; (b) : image reconstruite avec spectre d'amplitude initial et spectre de phase aléatoire ; (c) : image reconstruite avec spectre de phase initial et spectre d'amplitude aléatoire.