Mise en œuvre de traitements fréquentiels au moyen de la bibliothèque FFTW

Présentation du module

Description générale du travail attendu : mise en œuvre des transformations de Fourier directes et inverses, utilisation de la représentation complexe, extraction et visualisation des spectres d'amplitude et de phase, modification du spectre d'amplitude.

Activité notée : écriture de programmes en langage C utilisant la bibliothèque `FFTW3` et la bibliothèque `libimago` :

- `fft.c` : mise en œuvre des transformation directe et inverse, extraction et recomposition de spectres d'amplitude et de phase ;
- `fip.c` : sauvegarde de l'image d'un spectre d'amplitude ou de phase, suppression de fréquences dans le spectre d'amplitude ;
- `test-rgbfip-4.c` : traitement fréquentiel d'une image RGB.

Modalités : activité effectuée en binôme sur une durée de deux semaines.

Modalités d'évaluation de l'activité

Système de notation. Cette activité est évaluée par référence à cinq niveaux d'apprentissage :

- **échec** : aucun élément de réponse correct n'a été fourni ;
- **insuffisant** : la réponse contient des éléments attendus mais n'atteint pas le niveau minimum requis ;
- **suffisant** : un minimum requis d'éléments attendus est présent et correct ;
- **bon** : les éléments attendus sont globalement présents et corrects ;
- **excellent** : la réponse est complète et traduit une maîtrise des éléments évalués dans le contexte de cette activité.

Les notes correspondantes sont respectivement A, B, C, D et F. Les notes A, B et C des trois premiers niveaux peuvent dans le cas échéant être modulées en respectivement A+, A-, B+, B-, C+ et C-.

Critères d'évaluation. Capacité à utiliser la repésentation fréquentielle des images dans un programme en langage C (correspondance entre représentation spatiales et fréquentielles, utilisation des spectres).

D (mise en œuvre des transformations directes et inverses) : le fichier `fft.c` est compilable avec un environnement standard ISO C99/POSIX et les fonctions `fft_forward` et `fft_backward` produisent des résultats corrects aux tests de l'activité 1 ;

C (extraction et recomposition des spectres d'amplitude et de phase) : les fonctions `fft_extract_as`, `fft_extract_ps` et `fft_compose_aps` permettent d'obtenir des résultats corrects aux tests de l'activité 2.

B (manipulation des spectres d'amplitude et de phase) : le fichier `fip.c` est compilable avec un environnement standard ISO C99/POSIX et les fonctions `fip_save_image_spectrum` et `fip_as_cut` produisent des résultats corrects aux tests des activités 3 et 4.

A (mise en œuvre d'une transformée de Fourier sur une image RGB) : le fichier `test-rgbfip-4.c` est compilable avec un environnement standard ISO C99/POSIX et les tests de l'activité 5 produisent des résultats corrects.

Les jeux de test proposés sont similaires à ceux qui seront utilisés pour évaluer les réponses.

1 Mise en œuvre des transformations directes et inverses

Écriture des fonctions :
— `fft_forward`
— `fft_backward`

Activité **1**

Écrire un fichier `fft.c` contenant deux fonctions `fft_forward` et `fft_backward` dont l'interface est décrite dans le fichier `fft.h` fourni (le fichier `fft.c` devra inclure le fichier `fft.h`).

La fonction `fft_forward` reçoit en paramètre l'adresse d'un vecteur de caractères sans signe contenant le canal de l'image à transformer et retourne un vecteur de complexes alloué dynamiquement et contenant le résultat de la transformation directe (représentation fréquentielle).

Les allocations dynamiques et libérations de données complexe doivent être effectuées avec les fonctions `fftw_malloc` et `fftw_free`.

Ce fichier sera testé en le compilant avec le fichier `test-glfip-1.c`, également fourni, qui enchaîne transformation directe et inverse sur une image monochrome huit bits, et qui utilise la bibliothèque `imago2` (voir activité 1)

Conseils pour l'activité 1.

1. Il est conseillé d'écrire deux fonctions internes au module `fft` effectuant les conversions d'un canal du type `unsigned char` vers le type `fftw_complex` et réciproquement :

```
static fftw_complex *conv_uchar_to_complex(unsigned char *channel,
                                           int width, int height)
static unsigned char *conv_complex_to_uchar(fftw_complex *c_channel,
                                            int width, int height)
```

   C'est au sein de ces deux fonctions que peuvent s'effectuer les opérations permettant le recentrage des spectres par inversion du signe d'un point sur deux (voir rappels page 7). La seconde fonction doit effectuer une **troncature entre 0 et 255** avant la conversion en `unsigned char`.

2. Les transformations discrètes s'écrivent :

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

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

   La transformation directe nécessite une **normalisation** (division par $M{\times}N$ dans l'équation (1)). Cette normalisation n'est pas mise en œuvre par la bibliothèque `FFTW` et doit donc être effectuée au moment de la construction de la représentation spatiale complexe.

3. Deux tests peuvent être effectués au moyen de la commande `test-glfip-1` :

```
test-glfip-1 [-z] INPUT OUTPUT
```

   — sans option, le fichier `INPUT`[1] sur lequel sont enchaînées une transformation directe puis inverse, et le fichier `OUTPUT` contenant le résultat, doivent être visuellement identiques ;
   — avec l'option `-z`, le résultat doit être assombri, ce qui permet de vérifier que le recentrage a été correctement effectué.

1. Image monochrome 8 bit dans un format reconnu par la bibliothèque `imago2`.

2 Extraction et recomposition des spectres d'amplitude et de phase

Écriture des fonctions :
— `fft_extract_as`
— `fft_extract_ps`
— `fft_compose_aps`

Activité **2** *Ajouter au fichier `fft.c` les trois fonctions `fft_extract_as`, `fft_extract_ps` et `fft_compose_aps` effectuant respectivement l'extraction du spectre d'amplitude d'une représentation fréquentielle complexe, l'extraction du spectre de phase, et finalement la reconstruction de la représentation fréquentielle complexe à partir d'un spectre d'amplitude et d'un spectre de phase (voir rappels page 7).*

L'interface de ces fonctions est décrite dans le fichier `fft.h`.

Cette fonction sera testée en compilant `fft.c` avec le fichier `test-glfip-2.c` (fourni), qui enchaîne transformation directe, extraction des spectres, recomposition des spectres et transformation inverse sur une image monochrome 8 bits, et qui utilise la bibliothèque `imago2`.

Conseils pour l'activité 2.

1. Il est conseillé d'utiliser la fonction `atan2(b, a)` pour calculer $atan\left(\frac{b}{a}\right)$.
2. Deux tests peuvent être effectués au moyen de la commande `test-glfip-2` :

   ```
   test-glfip-2 [-c] INPUT OUTPUT
   ```

   — sans option, les fichier `INPUT` et `OUTPUT` doivent être visuellement identiques ;
   — avec l'option `-c`, la commande annule le spectre de phase avant reconstruction produisant une image dont toutes les fréquences sont de phase nulle (voir exemple fourni).

3 Visualisation des spectres d'amplitude et de phase

Écriture de la fonction :
— `fip_save_image_spectrum`

Activité **3** *Écrire un fichier `fip.c` (pour* Frequency Image Processing*) contenant la fonction `fip_save_image_spectrum` qui sauvegarde l'image d'un spectre d'amplitude ou de fréquence. L'interface de cette fonction, définie dans le fichier `fip.h` est la suivante :*

```
void fip_save_image_spectrum(double *spectrum,
                             int width, int height,
                             char *name,
                             bool phase);
```

- *`spectrum` : adresse d'un vecteur de double contenant le spectre à sauvegarder sous forme d'image ;*
- *`width`, `height` : largeur et hauteur du spectre ;*
- *`name` : nom du fichier image à construire ;*
- *`phase` : `true` s'il s'agit d'un spectre de phase et `false` sinon (voir explication plus bas).*

Ce fichier devra inclure le fichier `fip.h`. La fonction sera testée en compilant `fip.c` avec le fichier `test-glfip-3.c` (fourni), qui enchaîne transformation directe, extraction des spectres, et la sauvegarde des spectres d'amplitude et de phase.

Conseils pour l'activité 3.

1. La dynamique des spectres est trop importante pour être visualisée directement sur 256 niveaux de gris (voir cours). Pour cela on applique une **correction non linéaire** aux valeurs d'un spectre $S(u,v)$ avant de les ramener sur l'intervalle [0..255]. On utilisera ici la correction suivante :

$$S'(u,v) = \left(\frac{S(u,v)}{S_{max}(u,v)}\right)^k \times 255 \quad (\text{avec } k = 0.15)$$

2. Le spectre de phase ayant des valeurs négatives[2], le résultat de la normalisation $\frac{S(u,v)}{S_{max}(u,v)}$ est compris entre -1 et 1. Il est donc utile de ramener cette valeur entre 0 et 1 en lui appliquant la fonction $\frac{x+1}{2}$ avant d'effectuer la correction non linéaire. Cet ajustement sera effectué lorsque le paramètre `phase` vaut `true`.

2. Attention au calcul du maximum $S_{max}(u,v)$ dans ce cas.

4 Modification du spectre d'amplitude d'une image monochrome

Écriture de la fonction :
— `fip_as_cut`

Activité 4 *Ajouter au fichier `fip.c` la fonction `fip_as_cut` qui met à zéro des intervalles de fréquences et dont l'interface, définie dans le fichier `fip.h`, est la suivante :*

```
void fip_as_cut(double *spectrum,
                int width, int height,
                int min, int max,
                int mode);
```

— *`spectrum` : adresse d'un vecteur de `double` contenant le spectre à modifier ;*
— *`width`, `height` : largeur et hauteur du spectre ;*
— *`min`, `max` : intervalle de fréquences ;*
— *`mode` : paramétrage de la modification (voir ci-dessous).*

La fonction `fip_as_cut` doit mettre en œuvre deux modifications, selon la valeur du paramètre `mode` :

0 : annule l'amplitude des fréquences inférieures à `min` ou supérieures à `max` ;
1 : annule l'amplitude des fréquences comprises dans l'intervalle `[min, max]`.

Cette fonction sera testée en compilant le fichier `fip.c` avec le fichier `test-glfip-4.c` (fourni), qui enchaîne transformation directe, extraction des spectres, appel à la fonction `fip_as_cut`, reconstruction de la représentation spatiale et sauvegarde de l'image résultante.

5 Traitement fréquentiel d'image couleur

Écriture du fichier :
— `test-rgbfip-4.c`

Les fonctions développées dans les activités 1 à 4 permettent de mettre en œuvre un traitement fréquentiel d'image monochrome. Cette dernière activité a pour but d'utiliser ces fonctions sur une image couleur.

Activité 5 *En modifiant le fichier `test-glfip-4.c`, écrire le fichier `test-rgbfip-4.c`, effectuant le même traitement, mais pour image couleur.*

Conseils pour l'activité 5.

1. Le passage au traitement fréquentiel couleur s'effectuant canal par canal, il ne demande pas de modification des modules `fft` et `fip` déjà écrits.
2. La réécriture de `test-glfip-4.c` en `test-rgbfip-4.c` peut se faire de manière simple en utilisant les deux fonctions :

```
unsigned char *channel_extract(unsigned char *pixels, int size,
                               int channel_number);
unsigned char *channel_compose(unsigned char *r_chan, unsigned char *g_chan,
                               unsigned char *b_chan, size_t size);
```

réalisant l'extraction d'un des trois canaux d'une image RGB et combinant trois canaux pour construire une image RGB (voir activité AA01).

Compléments : principes d'utilisation de la bibliothèque FFTW.

— Voir également la documentation de la bibliothèque —

◦ API de la bibliothèque

— Interface :

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

— Compilation : `CFLAGS= -g -Wall -std=c99`

— Édition de liens : `-lfftw3 -lm`

◦ Types

— `fftw_complex` : type codant un nombre complexe. Correspond au type `complex` de `c99` :

```
complex c = a + I*b;
double a = creal(c);
double b = cimag(c);
```

— `fftw_plan` : structure de données utilisée pour paramétrer et mettre en œuvre une transformation directe ou inverse.

◦ Mise en œuvre des transformations

1. Construction d'un « plan », de type `fftw_plan`, décrivant les caractéristiques de la transformation à mettre en œuvre : taille de l'image, pointeurs vers les buffers stockant les représentations spatiales et fréquentielles de l'image, type de transformation (directe ou inverse), et mode de calcul :

   ```
   fftw_plan fftw_plan_dft_2d(int height, int width,
                              fftw_complex *in, fftw_complex *out,
                              int sense, unsigned flags);
   ```

   Les entiers `height` et `width` définissent la taille de l'image.

   Les pointeurs `in` et `out` doivent pointer vers des zones mémoires valides et correspondant à la taille des représentations complexes des images.

   Le paramètre `sense` permet de choisir entre une transformation directe (constante `FFTW_FORWARD`) ou inverse (constante `FFTW_BACKWARD`).

   Le paramètre `flag` sera fixé à `FFTW_ESTIMATE`.

2. Calcul de la transformation en activant l'exécution du plan :

   ```
   void fftw_execute(fftw_plan plan)
   ```

   Le résultat est placé à l'adresse `out` de l'appel à `fftw_plan_dft_2d`.

3. Libération du plan une fois la transformation effectuée :

   ```
   void fftw_destroy_plan(fftw_plan plan)
   ```

Rappels : étapes de mise en œuvre d'un traitement fréquentiel

1. Recentrage

$$I'(x,y) \ \rightarrow \ I'(x,y) \ = \ (-1)^{(x+y)} I(x,y)$$

2. Normalisation par $\frac{1}{MN}$ et conversion en complexe

$$\frac{1}{M \times N} \ I'(x,y) \ \rightarrow \ I'_C(x,y)$$

3. FFT appliquée à l'image complexe : construction de la représentation fréquentielle

$$I'_C \rightarrow F$$

4. Extraction des spectres d'amplitude A et de phase H à partir de la représentation fréquentielle :

$$F(u,v) \ = \ a + bi \ \rightarrow \ \begin{cases} A(u,v) = \sqrt{a^2 + b^2} \\ H(u,v) = atan(\frac{b}{a}) \end{cases}$$

5. Traitement fréquentiel par modification du spectre d'amplitude : $A \rightarrow A'$ (respect de la contrainte de symétrie)
6. Reconstruction de la représentation fréquentielle F' à partir des spectres (spectre d'amplitude modifié) :

$$F'(u,v) \ = \ A'(u,v) cos\left(H(u,v)\right) + iA'(u,v) sin\left(H(u,v)\right)$$

7. FFT inverse : reconstruction de la représentation spatiale de l'image modifiée :

$$F' \ \rightarrow \ I''_C$$

8. Inversion du recentrage et retour en représentation entière (ici, `unsigned char`) après avoir effectué une troncature (entre 0 et 255) :

$$I''_C \ \rightarrow \ I'' \ \rightarrow \ I'''(x,y) \ = \ (-1)^{(x+y)} I''(x,y)$$