université de BORDEAUX

Cycle préparatoire de l'université de bordeaux

Mémoire de projet

Tranformée de Fourier appliquée au traitement de l'image

Jérémie Soetens
Launois Alexandre

Tuteur : M. Leclaire
2018-2019

Table des matières

1 Remerciements

Nous tenons à remercier Monsieur Popoff, Maître de conférences à l'Université de Bordeaux, qui fût notre enseignant de mathématiques lors du semestre 3 et qui nous a aidé à trouver notre tuteur pour ce projet.

Nous remercions tout particulièrement notre tuteur Monsieur Leclaire qui a sû nous aiguiller lors de ce projet, en nous faisant travailler sur des sujets à notre portée tout en nous introduisant de nouvelles connaissances en mathématiques.

Nous lui sommes très reconnaissant pour son investissement dans ce projet, et tout le temps qu'il nous a accordé. Son aide nous a également été précieuse pour la relecture de notre mémoire.

2 Introduction

Dans ce projet, nous verrons des techniques de traitement des images utilisant la transformée de Fourier. Cependant, vous me direz qui est Fourier ? Joseph Fourier est un mathématicien connu pour avoir déterminé par le calcul la diffusion de la chaleur, en décomposant une fonction en une série trigonométrique convergente. De telles fonctions sont appelées série de Fourier, et ainsi a découlé de ces séries la transformée de Fourier.

Dans ce mémoire, nous allons dans un premier temps introduire ce que sont les séries de Fourier et leurs propriétés et ensuite nous verrons plus en détail la transformée de Fourier. Mais avant introduisons certaines notions et notations que nous allons utiliser dans ce mémoire. Il est important de savoir qu'une fonction périodique est une fonction qui pour une variable donnée a une certaine valeur et reprend la même valeur si nous ajoutons à cette variable une quantitée fixe appelée période. Une notion à introduire est le bruit, lorsque nous parlons de bruit, nous parlons d'un grain, qui vient perturber l'image et donc nous faisant perdre des détails sur certains pixels.

L'intérêt de la transformée de Fourier pour le traitement des images vient du fait que nous pouvons faire une décomposition fréquentielle de l'image et ainsi modifier l'image bien plus rapidement. Puisque c'est plus efficace de modifier un signal que chaque éléments de l'image. Avec la transformée de Fourier nous pouvons d'autre part diagonaliser des opérateurs invariants par translation, qui nous sera très utile pour flouter une image. Dès le départ, le but de ce projet était de se rapprocher du mécanisme de super-résolution. Mais ce but était trop ambitieux puisque nous n'avions pas pensé au fait que lorsque nous prenons plusieurs photos en basse résolution pour en faire une en haute résolution, nous avons des translations entre chaques photo mais pour déterminer ces translations, cela est très compliqué puisque ceux sont des méthodes que nous utilisons en master et à notre niveau nous n'avons pas les connaissances nécessaires.

Vous pouvez retrouver nos simulations sur le traitement des images, si vous le souhaiter ré-exécuter les programmes pour voir le résultat en téléchargent l'archive GITHUB

3 Généralités sur les images en informatique

Un écran d'ordinateur n'est qu'un ensemble de pixels qui ont chacun une couleur. Chaque pixel est caractérisé par 3 diodes de couleurs : rouge, vert, bleu. Une image est en fait un tableau de couleur, elle peut être représentée par un tableau de 2 dimensions, dont chaque élément contient 3 nombres (pour le rouge, le vert et le bleu). Une couleur est représentée par 3 nombres entiers représentant respectivement (dans l'ordre) l'intensité de rouge, de vert et de bleu allant de 0 à 255 : (r, v, b). $(255, 0, 0)$ est la couleur d'un rouge pur, $(0, 0, 0)$ du noir et $(255, 255, 255)$ du blanc par exemple.

3.1 Les images en niveau de gris

Durant l'intégralité du projet nous allons travailler sur des images en niveau de gris. Un pixel gris est de la forme (n, n, n), l'intensité du rouge, du vert et du bleu est la même. Donc un pixel gris est décrit par un seul nombre entier allant de 0 à 255 et donc une image grise n'est en fait qu'un tableau de nombre entier allant de 0 à 255. Il existe différentes manières pour passer d'une image couleur en image grise, une manière simple et de faire la moyenne de r, v et b : $\frac{r+v+b}{3}$

FIGURE 1 – Exemple d'une image en niveau de gris

Voici un exemple d'une image en niveau de gris ainsi que sa représentation sous la forme d'un tableau de nombre entier allant de 0 à 255 comme expliqué.

3.2 Une image peut être assimilé à un signal périodique à deux dimensions

Nous allons voir comment associer à une image un signal périodique. Pour montrer cela nous allons procéder par analogie avec un signal 1D. Prenons par exemple le signal 1D du son d'une trompette

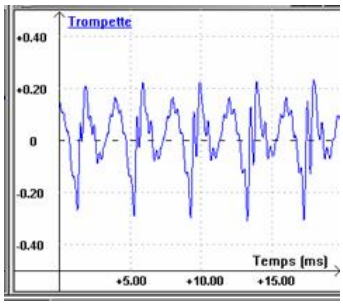


FIGURE 2 – Amplitude d'un signal sonore au cours du temps d'une trompette

L'amplitude varie par rapport à une seule variable : le temps et de plus le signal est périodique (cela se voit très bien sur FIGURE 2). Maintenant pour une image c'est plus difficile à voir. Prenons par exemple cet exemple très simple, et notons que chaque pixel a une valeur (un entier allant de 0 à 255) et que sa position est donnée par deux coordonnées (x, y). Ainsi une façon de voir l'image est comme un signal 2D : la couleur du pixel qui varie par rapport à deux variables : la coordonnée en x et la coordonnée en y.

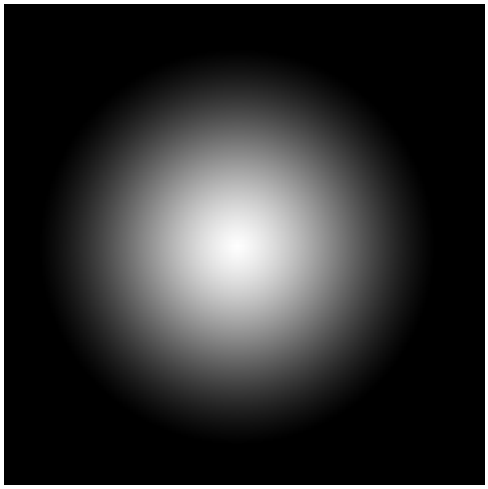


FIGURE 3 – Image vue sous sa forme "classique"

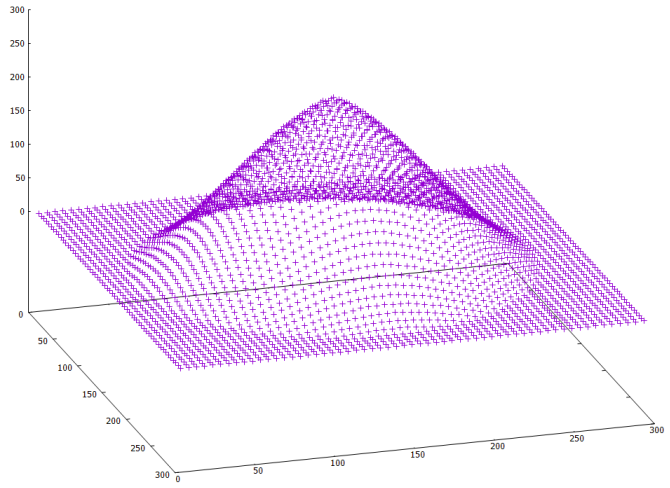


FIGURE 4 – Image vue comme un signal 2D

De plus pour que ce signal 2D soit périodique, l'image est répétée comme ceci et à l'infini donc si l'on sort de l'image par la droite on se retrouve à gauche de l'image.

FIGURE 5 – Lena répété 9 fois

Une image est un signal 2D périodique. Le fait de pouvoir représenter une image de cette manière est crucial puisqu'elle nous permet d'appliquer la théorie de Fourier qui a pour objet l'étude de signaux périodiques que nous allons détailler de suite.

4 Théorie de Fourier

4.1 Séries de Fourier et coefficients trigonométriques

Definition 4.1. *Coefficients trigonométriques Soit $f \in CM_{2\pi}$, on appelle coefficients trigonométriques de f, les coefficients :*

$$\forall n \in \mathbb{N}, a_n(f) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(nt)dt$$
$$\forall n \in \mathbb{N}, b_n(f) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(nt)dt$$
$$\forall n \in \mathbb{N}, c_n(f) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)e^{-int}dt$$

Definition 4.2. *On définit les suites de fonctions $(e_n)_{n\in\mathbb{Z}}$, $(r_n)_{n\in\mathbb{Z}}$ et $(s_n)_{n\in\mathbb{Z}}$ par :*

$$e_n : \mathbb{R} \to \mathbb{C}, e_n(t) = e^{ßnt}$$
$$r_n : \mathbb{R} \to \mathbb{R}, r_n(t) = \cos(nt)$$
$$s_n : \mathbb{R} \to \mathbb{R}, s_n(t) = \sin(nt)$$

.

Definition 4.3. *Soit $f \in CM_{2\pi}$, on appelle série de Fourier de f, la série de fonctions de $\mathbb{R}$ dans $\mathbb{C}$:*

$$\sum_{n\in\mathbb{Z}} c_n(f)e_n = \frac{a_0(f)}{2} + \sum_{n=1}^{+\infty} a_n(f)r_n + b_n(f)s_n.$$

Si on note $(S_p)_{p\in\mathbb{N}}$, la suite des sommes partielles de cette série de fonctions, on a :

$$\forall p \in \mathbb{N}, S_p(f)(x) = \sum_{n=-p}^{p} c_n(f)e^{ßnx} = \frac{a_0(f)}{2} + \sum_{n=-p}^{p}(a_n(f)\cos(nx) + b_n(f)\sin(nx))$$

Theorem 4.4. *Formule de Parseval Soit $f \in C_{2\pi}, ||f||_2^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi}|f(t)|^2dt = \sum_{n=-\infty}^{+\infty}|c_n(f)|^2$*

$$= |\frac{a_0(f)}{2}|^2 + \frac{1}{2}\sum_{n=1}^{+\infty}|a_n(f)|^2 + |b_n(f)|^2 \text{ ou } \sum_{n=-\infty}^{+\infty}|c_n(f)|^2 = \lim_{p\to+\infty}\sum_{n=-p}^{p}|c_n(f)|^2 \quad (1)$$

Theorem 4.5. *Inégalité de Bessel Soit $f \in CM_{2\pi}$, $\sum_{n=-p}^{p}|c_n(f)|^2 \leqslant ||f||_2^2$ ce qui implique que :*

$$|\frac{a_0(f)}{2}|^2 + \frac{1}{2}\sum_{n=1}^{+\infty}|a_n(f)|^2 + |b_n(f)|^2 \leqslant ||f||_2^2.$$

Theorem 4.6. *Convergence absolue et uniforme de la série de Fourier Si $f : \mathbb{R} \to \mathbb{C}$ est 2π-périodique C1 par morceaux et continue sur $\mathbb{R}$, alors la série de Fourier de f, $\lim_{n\to\infty} S_n(f)$, converge absolument et uniformément vers f.*

Theorem 4.7. *Convergence en moyenne quadratique Soit $f \in C_{2\pi}$. Alors la suite $(S_p(f))_p$, converge vers f dans l'espace vectoreil normé $(\mathbb{C}_{2\pi}, ||.||_2)$ et :*

$$\lim_{p\to\infty}||f - S_p(f)||_2 = 0.$$

4.2 Transformée de Fourier discrète (TFD)

4.2.1 Définition

Soit f une fonction 2π-périodique à valeur dans $\mathbb{C}$. On effectue une subdivision régulière de f sur un intervalle de longueur 2π, ce qui définit la famille de N points. $x \in \mathbb{C}^N$ par $\left(x(k) = f(\frac{2\pi k}{N})\right)_{k\in\{0,...,N-1\}}$

La transformée de fourier discrète de x, noté $\hat{x}$ est définie par :

$$\hat{x}(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-2ß\pi nk}{N}} \tag{2}$$

On peut montrer que l'on peut retrouver les $x(k)$ à partir des $\hat{x}(k)$ grâce à une transformée de Fourier inverse :

$$x(k) = \frac{1}{N} \sum_{n=0}^{N-1} \hat{x}(n) e^{\frac{2ß\pi nk}{N}} \tag{3}$$

À noter que le coefficient $1/N$ dans la transformée de Fourier inverse peut être mis dans la transformée Fourier directe, c'est au choix.

4.2.2 Quelques résultats essentiels

On munit l'espace vectoriel $\mathbb{C}^N$ du produit scalaire hermitien : $\langle x|y\rangle = \frac{1}{N}\sum_{k=0}^{N-1} x(k)\overline{y(k)}$, posons $e_k = \left(1, e^{\frac{2ß\pi k}{N}}, e^{\frac{2ß\pi 2k}{N}}, ..., e^{\frac{2ß\pi(N-1)k}{N}}\right) = \left(e^{\frac{2ß\pi lk}{N}}\right)_{l\in\{0,...,N-1\}}$

Lemme 4.8. *Les $(e_k)_{k\in\{0,...,N-1\}}$ forment une base orthonormée de $\mathbb{C}^N$*

Démonstration. $\langle e_m|e_j\rangle = \frac{1}{N}\sum_{k=0}^{N-1} e_m(k)\overline{e_j(k)} = \frac{1}{N}\sum_{k=0}^{N-1} e^{\frac{2ß\pi km}{N}} e^{\frac{-2ß\pi kj}{N}} = \frac{1}{N}\sum_{k=0}^{N-1} e^{\frac{2ß\pi k(m-j)}{N}} = \frac{1}{N}\sum_{k=0}^{N-1}\left(e^{\frac{2ß\pi(m-j)}{N}}\right)^k$

On obtient la somme des termes d'une suite géométrique, et puisque :

$e^{\frac{2ß\pi(m-j)}{N}} \neq 1 \Leftrightarrow \frac{2\pi(m-j)}{N} \not\equiv 0\ [2\pi] \Leftrightarrow \frac{(m-j)}{N} \notin \mathbb{N} \Leftrightarrow m \neq j$ (car $m, j \in \{0, ..., N-1\}$)

$$m \neq j \Rightarrow \langle e_m|e_j\rangle = \frac{1}{N}\sum_{k=0}^{N-1}\left(e^{\frac{2ß\pi(m-j)}{N}}\right)^k = \frac{1}{N}\frac{1-e^{\frac{2ß\pi Nk}{N}}}{1-e^{\frac{2ß\pi k}{N}}} = \frac{1}{N}\frac{1-e^{2ß\pi k}}{1-e^{\frac{2ß\pi k}{N}}} = \frac{1}{N}\frac{1-1}{1-e^{\frac{2ß\pi k}{N}}} = 0$$

$$i = j \Rightarrow \langle e_m|e_j\rangle = \frac{1}{N}\sum_{k=0}^{N-1}\left(e^{\frac{2ß\pi(m-j)}{N}}\right)^k = \frac{1}{N}\sum_{k=0}^{N-1} 1 = 1$$

Ainsi $\langle e_m|e_j\rangle = \delta_{mj} = \begin{cases} 1, & \text{si } m = j, \\ 0, & \text{si } m \neq j. \end{cases}$ c'est donc bien une base orthonormée.

□

Corollaire 4.8.1. *Ceci signifie que $\hat{x}(k) = \langle x|e_k\rangle$*

Démonstration. $\forall x \in \mathbb{C}^N$ x s'écrit $x = \sum_{k=0}^{N-1} \langle x|e_k\rangle e_k$ (résultat d'algèbre linéaire)

Autrement dit $x(l) = \sum_{k=0}^{N-1} \langle x|e_k\rangle e_k(l) = \sum_{k=0}^{N-1} \langle x|e_k\rangle e^{\frac{2ß\pi lk}{N}} = \frac{1}{N}\sum_{k=0}^{N-1} N\langle x|e_k\rangle e^{\frac{2ß\pi lk}{N}}$

On reconnaît la formule de la transformée de Fourier inverse et par identification $\hat{x}(k) = N\langle x|e_k\rangle$ □

En fait la transformée de Fourier discrète se résume à une simple projection orthogonale sur $(e_k)_{k\in\{0,\dots,N-1\}}$ (à un coefficient près)

Propriété 4.9 (Produit de convolution d'une TFD)**.** $\widehat{x * y} = \hat{x}\hat{y}$

Démonstration. $\widehat{x*y}(n) = \sum_{k=0}^{N-1}(x*y)(k)e^{\frac{-2ß\pi nk}{N}} = \sum_{k=0}^{N-1}\left(\sum_{j=0}^{N-1} x(k-j)y(j)\right)e^{\frac{-2ß\pi nk}{N}}$ Puis en utilisant le fait que $e^{\frac{-2ß\pi nk}{N}} = e^{\frac{-2ß\pi n(k-j+j)}{N}} = e^{\frac{-2ß\pi n(k-j)}{N}}e^{\frac{-2ß\pi nj}{N}}$. Et en permutant les deux sommes, on obtient :

$$\widehat{x*y}(n) = \sum_{j=0}^{N-1}\left(\sum_{k=0}^{N-1} x(k-j)y(j)e^{\frac{-2ß\pi n(k-j)}{N}}\right)e^{\frac{-2ß\pi nj}{N}} = \sum_{j=0}^{N-1} y(j)e^{\frac{-2ß\pi nj}{N}}\left(\sum_{k=0}^{N-1} x(k-j)e^{\frac{-2ß\pi n(k-j)}{N}}\right)$$

De plus, vu que $x(qN+p) = x(p)$ (du fait que y est un signal N périodique) et $e^{\frac{-2ß\pi n(qN+p)}{N}} = e^{-2ß\pi nq} * e^{\frac{-2ß\pi np}{N}} = e^{\frac{-2ß\pi np}{N}}$ on a $\forall j \in \{0,\dots,N-1\}$ $\sum_{k=0}^{N-1} x(k-j)e^{\frac{-2ß\pi n(k-j)}{N}} = \sum_{k=0}^{N-1} x(k)e^{\frac{-2ß\pi nk}{N}} = \hat{x}(k)$

Ainsi $\widehat{x*y}(n) = \hat{x}(k)\sum_{j=0}^{N-1} y(j)e^{\frac{-2ß\pi nj}{N}} = \hat{x}(k)\hat{y}(k)$

□

Cette propriété sur les convolutions va s'avérer très utile plus tard lors de l'implémentation d'un filtre local pour flouter une image par exemple.

4.2.3 Coefficients de Fourier et lien avec la TFD

Dans cette partie nous allons considérer une fonction f 2π-périodique à valeur dans $\mathbb{C}$ de classe C^k. De plus on effectue une subdivision régulière de f sur un intervalle de longueur 2π, ce qui définit la famille de N points : $x \in \mathbb{C}^N$ par $\left(x(k) = f(\frac{2\pi k}{N})\right)_{k\in\{0,\dots,N-1\}}$

Proposition 4.10. *$\hat{x}(k)$ est l'approximation du calcul de $c_k(f)$ par une somme de Reimann à un coefficient N près.*

Démonstration.

$$\frac{\hat{x}(p)}{N} = \frac{2\pi}{2\pi N}\sum_{k=0}^{N-1} x(k)e^{\frac{-2ß\pi pk}{n}} = \frac{1}{2\pi}\frac{2\pi}{N}\sum_{k=0}^{N-1} f(\frac{2\pi k}{N})e^{-ß(\frac{2\pi k}{N})p} \underset{N\to\infty}{\longrightarrow} \frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-itp}dt = c_p(f)$$

□

Nous voyons que c'est une somme de Riemann, ainsi calculer une transformée de Fourier d'une subdivision régulière de f revient à approximer les coefficients de Fourier de f, nous allons maintenant évaluer l'erreur commise par cette approximation.

Lemme 4.11. *Si f est de classe C^k alors $c_p(f) = o(\frac{1}{|p|^k})$*

Démonstration. Soit $n \in \{1, ..., k\}$ alors $f^{(n)}$ est continue donc $c_p(f^{(n)})$ existe. Ainsi :

$$c_p(f^{(n)}) = \frac{1}{2\pi}\int_0^{2\pi} f^{(n)}(t)e^{-ipt}dt \underset{\text{IPP}}{=} \frac{1}{2\pi}\left[f^{(n-1)}(t)e^{-ipt}\right]_0^{2\pi} + \frac{1}{2\pi}\int_0^{2\pi} f^{(n-1)}(t)ßpe^{-ipt}dt$$

Par 2π périodicité et continuité de $f^{(n-1)}$ et e^{-ipt}, le premier terme est nul, et le second terme on reconnaît la formule du coefficient de Fourier de $f^{(n-1)}$ à un facteur ip près. Ainsi on obtient :

$$c_p(f^{(n)}) = ßpc_p(f^{(n-1)})$$

Et par récurrence sur n on a finalement et en prenant $n = k$:

$$c_p(f^{(k)}) = ß^n p^k c_p(f)$$

Et donc $c_p(f) = \frac{1}{ß^k}\frac{c_p(f^{(k)})}{p^k} = o(\frac{1}{|p|^k})$ □

Proposition 4.12. $\frac{\hat{x}(p)}{N} - c_p(f) = \sum_{q\in\mathbb{Q}^*} c_{p+Nq(f)} = o(\frac{1}{N^{k-1}})$

Démonstration. Supposons que $\sum_{n=-\infty}^{+\infty} |c_n(f)| < +\infty$ en supposant par exemple que f est au moins C^2 par exemple ce qui nous assure que cette série converge. On peux écrire f d'après les résultats sur les séries de Fourier comme ceci : $f(t) = \sum_{m=-\infty}^{+\infty} c_m(f)e^{ßmt}$, de plus par division euclidienne m peux s'écrire $m = qN + n$ avec $0 < n < N$ et $q \in \mathbb{Z}$:

$$\begin{aligned} x(k) = f(\frac{2\pi k}{N}) &= \sum_{m=-\infty}^{+\infty} c_m(f)e^{ßm\frac{2\pi k}{N}} = \sum_{q=-\infty}^{+\infty}\sum_{n=0}^{N-1} c_{qN+n}(f)e^{ß(qN+n)\frac{2\pi k}{N}} = \sum_{n=0}^{N-1}\sum_{q\in\mathbb{Z}} c_{qN+n}(f)e^{ßn\frac{2\pi k}{N}} \\ &= \sum_{n=0}^{N-1}\left(\sum_{q\in\mathbb{Z}} c_{qN+n}(f)\right)e^{ß\frac{2\pi nk}{N}} \end{aligned}$$

Par identification avec la formule de transformée de Fourier inverse, on déduit que $\hat{x}(n) = N\sum_{q\in\mathbb{Z}} c_{qN+n}(f)$

Et donc que $\frac{\hat{x}(n)}{N} - c_n(f) = \sum_{q\in\mathbb{Z}^*} c_{qN+n}(f) \leq \sum_{|p|\geq N} c_p(f) = \sum_{|p|>N} o(\frac{1}{|p|^k}) = o(\frac{1}{|N|^{k-1}})$ □

Ainsi l'erreur faite entre la calcul de la transformé de Fourier et les coefficients de Fourier décroit en $\frac{1}{N^{k-1}}$. Donc plus la fonction f est régulière (classe C^k) et plus notre subdivision régulière est fine (N est grand), plus l'approximation devient faible.

4.3 Transformée de Fourier discrète 2D

4.3.1 Définition

De manière analogue, on a la transformée de Fourier dite "2D" (celle utilisée pour les images)
Soit un signal u discret de taille (N, M) on définit la transformée de Fourier discrète directe et inverse par les formules suivantes :

$$\hat{u}(\zeta,\eta) = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} u(x,y)e^{-2ß\pi\left(\frac{\zeta x}{N}+\frac{\eta y}{M}\right)} \tag{4}$$

$$u(x,y) = \frac{1}{NM}\sum_{\zeta=0}^{N-1}\sum_{\eta=0}^{M-1} \hat{u}(\zeta,\eta)e^{2ß\pi\left(\frac{x\zeta}{N}+\frac{y\eta}{M}\right)} \tag{5}$$

4.3.2 Propriétés

Les résultats importants pour une TFD 2D sont similaires à ceux d'une TFD 1D étudiée précédemment.

Lemme 4.13. *Les* $\left(e_{ij} = (e^{2ß\pi\frac{ik}{N}+\frac{jl}{M}})_{k\in\{0,..,N-1\},l\in\{0,..,M-1\}}\right)_{i\in\{0,..,N-1\},j\in\{0,..,M-1\}}$ *forment une base orthonormée de* $\mathbb{C}^{NM}$ *pour le produit scalaire* $\langle x|y\rangle = \frac{1}{NM}\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} x(k,l)\overline{y(k,l)}$

Propriété 4.14. *Soit* $u, v \in \mathbb{C}^{NM}$ *alors* $\widehat{u * v} = \hat{u}\hat{v}$
Avec $(u * v)(x,y) = \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} u(x-k, y-l)v(k,l)$

4.4 Algorithme de transformée de Fourier rapide (Cooley-Tuckey)

4.4.1 Motivation

Calculer la transformée de Fourier d'un signal u de taille N, est un processus coûteux en temps. En effet pour chaque terme de $\hat{u}$ il faut calculer N multiplication donc la compléxité du calcul d'une transformée de Fourier (directe et inverse) est a priori $O(N^2)$. Le but de cet algorithme est de considérablement réduire la compléxité et donc le temps de calcul.
Pour des raisons de simplification algorithmique on supposera le signal u de taille $N = 2^k$. L'algorithme dans le cas général fait appel à des résultats d'arithmétique avec entre autre le théorème des restes chinois. On se limite au cas simple d'un signal d'une puissance de 2.

4.4.2 Explication et démonstration de l'algorithme

Soit u un signal de taille $N = 2^k$ et U sa transformée de Fourier. $U(k) = \sum_{n=0}^{N-1} u(n)e^{\frac{-2i\pi nk}{N}}$ On note pour des raisons de lisibilité $e^{\frac{-2i\pi nk}{N}} = \omega_N^{-nk}$. U est de taille N donc divisible par 2 ($N = 2m$), ainsi :

$$U(k) = \sum_{n=0}^{N-1} u(n)\omega_N^{-nk} = \frac{1}{2}(P(k) + \omega_N^{-k}I(k))$$

avec

$$\begin{cases} P(k) = & \frac{1}{m}\sum_{n=0}^{m} u(2n)\omega_N^{-2nk} \\ I(k) = & \frac{1}{m}\sum_{n=0}^{m-1} u(2n+1)\omega_N^{-2nk} \end{cases}$$

Par N périodicité de u et de ω_N, $P(k+m) = P(k)$ et $I(k+m) = I(k)$. Et de plus par un rapide calcul on montre que $\omega_N^{-(k+m)} = -\omega_N^{-k}$ Ainsi :

$$U(k) = \frac{1}{2}(P(k) + \omega_N^{-k} I(k))$$

$$U(k+m) = \frac{1}{2}(P(k) - \omega_N^{-k} I(k))$$

Avec ces relations il faut maintenant remarquer P et I n'est en fait que la transformée de Fourier respectivement des éléments pair et impair de u. Autrement dit calculer la transformée de Fourier de u de taille $N = 2m$ revient à calculer 2 transformées de Fourier de taille m (et ensuite de faire N additions). Et on peut ensuite répéter ce processus pour le calcul de la transformée de Fourier : P et I. On a un algorithme par récurrence !
Ainsi en notant C(N) la complexité de calcul d'une transformée de Fourier d'un signal de taille N, $C(N) = 2C(N/2)+N$. On a supposé $N = 2^k$ donc $C(N) = 2C(N/2)+N = 2(2C(N/4)+N/2)+N = 2^2C(N/4)+2N = 2^3C(N/2^3)+3N = ... = 2^kC(1)+kN = N(k+1)$ Et $k = \log_2(N) = O(\log(N))$ donc finalement :

$$C(N) = O(N \log N)$$

Ainsi grâce à cet algorithme on est passé d'une compléxité en $O(N^2)$ avec l'application directe de la formule à une compléxité de $O(N \log N)$ avec l'algorithme de Cooley-Tuckey.

5 Application au traitement de l'image

5.1 Notion de filtre

5.1.1 Filtrage local

L'idée du filtrage local est que chaque pixel de la nouvelle image est calculé à partir des pixels voisins de l'image d'origine. Notons $u'(x,y)$ et $u(x,y)$ la valeur du pixel aux coordonnées (x,y) respectivement de la nouvelle image et de l'image d'origine. Le voisinage d'une distance 1 du pixel (x,y) de l'image d'origine est :
$V(1)(x,y) := \begin{pmatrix} u(x-1,y-1) & u(x,y-1) & u(x+1,y-1) \\ u(x-1,y) & u(x,y) & u(x+1,0) \\ u(x-1,y+1) & u(x,y+1) & u(x+1,y+1) \end{pmatrix}$

Ensuite définissons le filtre F $= \begin{pmatrix} F(-1,-1) & F(0,-1) & F(1,-1) \\ F(-1,0) & F(0,0) & F(1,0) \\ F(-1,1) & F(0,1) & F(1,1) \end{pmatrix}$ ou les $F(i,j)$ sont des scalaires.

On calcul alors le pixel de la nouvelle image en (x,y) : $u'(x,y)$ à partir du voisinage de $u(x,y)$ et du filtre comme etant la somme des éléments de cette matrice :

$$\begin{pmatrix} F(1,1)u(x-1,y-1) & F(0,1)u(x,y-1) & F(-1,1)u(x+1,y-1) \\ F(1,0)u(x-1,y) & F(0,0)u(x,y) & F(-1,0)u(x+1,0) \\ F(1,-1)u(x-1,y+1) & F(0,-1)u(x,y+1) & F(-1,-1)u(x+1,y+1) \end{pmatrix}$$

Ainsi appliquer le filtre local F revient à faire ce calcul pour tous les pixels de la nouvelle image.

Autrement dit, appliquer ce filtre a chaque pixel de l'image revient a faire une convolution de l'image avec le filtre.

$$(u * F)(x,y) = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} u(x-k, y-l)F(k,l)$$

5.1.2 Utilisation de la formule de convolution

On a la relation $\widehat{u * F} = \hat{u}\hat{F}$ et donc $u * F = \widetilde{\hat{u}\hat{F}}$ Autrement dit, au lieu de faire une convolution, il suffit de faire la transformée de Fourier de u et F de les multiplier puis de faire une transformée de Fourier inverse ce qui est beaucoup plus simple (au sens de la complexité).

En effet la compléxité d'une convolution de deux signaux de taille n est $O(n^2)$ alors que la complexité d'une transformée de Fourier rapide est $O(n\log(n))$ ce qui donne une complexité de $O(n\log(n))$ en passant de le domaine fréquentiel pour faire la convolution.

5.2 Le débruitage

5.2.1 Définition du problème

Maintenant, nous allons voir quels problèmes se posent lorsque nous voulons débruiter une image.
Lorsque nous essayons de débruiter une image, nous arrivons dans les problèmes inverses d'imagerie. Soit la formulation suivante : w = Au_0 + n, (où A est un opérateur et dans notre cas le débruitage A est l'identitée) qui mène à ce problème inverse de traitement du signal. Ici ce que nous essayons de faire est de retrouver un signal u_0 à partir de son observation w, c'est-à-dire que nous essayons de retrouver l'image sans bruit à partir de ce que nous avons sous la main (en l'occurence une image bruitée). Il existe des méthodes visant à retrouver notre u_0 en utilisant des méthodes dites variationnelles, ici nous essayons de retrouver notre u_0 (notre signal "originel") en minimisant la norme de deux fonctions nous obtenons donc l'équation suivante

qui résume ce problème :

$$\tilde{u} = argmin_u D(u, w) + \lambda R(u).$$

Dans notre équation D est une fonction de distance qui quantifie la distance d'une solution potentielle aux observations. Et R une fonction de régularisation qui quantifie l'éloignement par rapport au modèle de régularité u_0, c'est-à-dire que cette fonction nous donne un niveau de bruit du signal (indique si le signal est très ou peu bruité). Donc notre but est le suivant : pour un ensemble E qui permet de modéliser la régularité des inconnues u_0 dans E, nous devons minimiser le problème afin d'obtenir $\tilde{u} = u_0$. Généralement, la méthode appliquée revient à essayer de minimiser une distance de norme L^p pour D, nous obtenons donc le problème de minimisation suivant :

$$\tilde{u} = argmin_u ||Au - w||_p^p + \lambda R(u).$$

5.2.2 Le cas lorsque nous le traitons dans L^2, L^2

En particulier, lorsque nous traitons ce problème dans L^2, (c'est-à-dire où p = 2) nous reformulons le problème suivant sous la forme suivante :

$$\tilde{u} = argmin_u ||u - w||_2^2 + \lambda ||\nabla u||_2^2.$$

Nous obtenons cette forme suivante car nous avons dans le cas du débruitage énoncé que l'opérateur A est équivalent à l'identitée, donc Au = u. De plus, pour trouver notre fonction de régularisation, il s'agit de la même manière de minimiser une norme L^q dans le cas choisit ici q = 2. Après un développement détaillé dans l'annexe 1, utilisant l'egalité de parseval on obtient l'expression de $\tilde{u}$ suivante dans le domaine fréquentiel :

$$\hat{\tilde{u}}(\zeta, \eta) = \frac{\hat{w}(\zeta, \eta)}{1 + \lambda |\zeta + ß\eta|^2}.$$

Nous obtenons donc les résultats suivants :

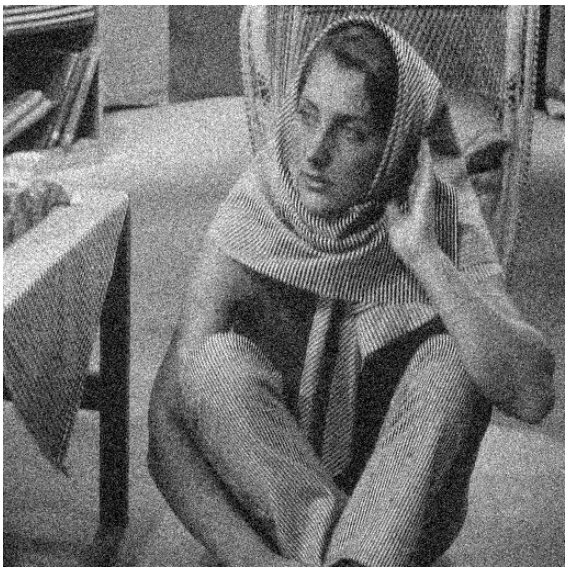


FIGURE 6 – Image bruitée

FIGURE 7 – Image débruitée

FIGURE 8 – Image bruitée

FIGURE 9 – Image débruitée

Bien que le bruit a presque disparu, il a été remplacé par un flou. Ce flou n'est pas voulu, il vient du choix de notre fonction de régularisation (la norme 2 du gradient). Ce choix arbitraire à l'avantage de rendre la minimisation assez facile à résoudre, mais n'est pas très adapté pour enlever le bruit, comme on peut le voir ci-dessus.

5.3 Problème : agrandir une image

Prenons par exemple une image 512x512 que nous voulons agrandir pour obtenir une image 1024x1024. Un algorithme assez naïf serait de prendre chaque pixel de la première image puis d'en faire un carré de 2x2 pixels. On pourrait aussi faire des carrés de 2x2 pixels pour chaque pixel et dont la valeur des 3 pixels rajouté sont une moyenne des pixels environnant, c'est-à-dire que nous faisons une interpolation linéaire des pixels. Et une dernière solution que nous allons détailler utilise une tranformée de Fourier.

Soit $u \in \mathbb{C}^{NM}$ on peut écrire $u(n,m) = \frac{1}{NM} \sum_{|k|<\frac{N}{2}} \sum_{|j|<\frac{M}{2}} \hat{u}(k,j) e^{2ß\pi\left(\frac{nk}{N}+\frac{mj}{M}\right)}$ par une tranformée de Fourier inverse où n et m sont des entiers.

En fait si on considère cette expression de u avec x et y des variables réelles à la place de n et m, on obtient un polynôme trigonométrique interpolateur de u. Nous allons utiliser ce polynôme interpolateur pour créer notre image deux fois plus grande. Définissons v tq $v(n,m) = u(\frac{n}{2}, \frac{m}{2})$.

$$\begin{aligned} v(n,m) &= \frac{1}{NM} \sum_{|k|<\frac{N}{2}} \sum_{|j|<\frac{M}{2}} \hat{u}(k,j) e^{2ß\pi\left(\frac{nk}{2N}+\frac{mj}{2M}\right)} \\ &= \frac{1}{NM} \sum_{|k|<N} \sum_{|j|<N} \hat{b}(k,j) e^{2ß\pi\left(\frac{nk}{2N}+\frac{mj}{2M}\right)} \\ &= \frac{1}{2N2M} \sum_{|k|<N} \sum_{|j|<N} 4\hat{b}(k,j) e^{2ß\pi\left(\frac{nk}{2N}+\frac{mj}{2M}\right)} \end{aligned}$$

Avec $\hat{b}(k,j) = \begin{cases} \hat{u}(k,j), & \text{si } |k| < N/2 \text{ et } |j| < M/2, \\ 0, & \text{sinon.} \end{cases}$

Donc par identification avec la formule de tranformée de Fourier inverse de v.

$$\hat{v}(k,j) = 4\hat{b}(k,j) = \begin{cases} 4\hat{u}(k,j), & \text{si } |k| < N/2 \text{ et } |j| < M/2, \\ 0, & \text{sinon.} \end{cases}$$

Il suffit alors de prendre une image u de taille (N,M), d'en faire la tranformée de Fourier directe pour obtenir $\hat{u}$ pour ensuite en déduire $\hat{v}$ par la formule ci dessus et par tranformée de Fourier inverse obtenir v : une image de taille (2N, 2M).

Ainsi nous obtenons pour chaque traitement cité ci-dessus les images suivantes :

FIGURE 10 – Image zoomée utilisant l'algorithme naïf

FIGURE 11 – Image zoomée par interpolation linéaire

FIGURE 12 – Image zoomée par un algorithme utilisant la transformée de Fourier

On peut remarquer que l'algorithme naïf n'est pas très performant, on voir apparaitre gros pixels notamment à la frontière entre le chapeau et les cheveux. Ensuite les deux dernières méthodes donnent un résultat assez similaire, cependant la méthode d'interpolation linéaire semble donner un résultat un peu moins net, notamment au niveau des cheveux.

5.4 L'anti-aliasing ou AA

5.4.1 Création de l'aliasing

Dans cette partie, nous allons voir comment créer de l'aliasing pour ensuite étudier une méthode pour éviter celui-ci. De nos jours, nombre de techniques existent pour éviter l'aliasing ou crénelage en français. Lorsque nous parlons d'aliasing aux générations actuelles, toute vont nous citer l'exemple des jeux-vidéos, car les principaux endroits ou nous pouvons traiter l'aliasing est bien sûr dans les jeux-vidéos. Cependant traiter l'aliasing et très compliqué et cela reste un problème ouvert car l'évolution des filtres est en lien avec l'évolution des graphismes dans les jeux. Ainsi dans cette partie nous aborderons une technique de filtrage qui est à notre portée car il existe beaucoup de filtres AA mais tous requiert des connaissances que nous n'avons pas.
Nous allons donc voir dans un premier temps une création artificielle d'effet de crénelage sur une image, pour ce faire nous aurons besoin de réduire l'image par deux, c'est-à-dire que nous voyons l'image comme un tableau ou chaque pixel représente une case, et nous supprimons un pixel sur deux pour chaque ligne et colonne.
Nous obtenons donc pour l'image de gauche, une image à droite qui est deux fois plus petite et qui a de l'aliasing (cependant pour que nous puissions mieux voir le résultat, nous avons mis les images aux mêmes dimensions) :

FIGURE 13 – Image originelle

FIGURE 14 – Image réduite avec aliasing

On peut voir que l'image réduite présente de l'aliasing au niveau des lignes du pantalon par exemple. Maintenant, nous allons voir une méthode pour réduire la taille d'une image sans créer cet effet d'aliasing. La technique que nous allons utiliser consiste à projeter l'image dans le domaine fréquentiel, supprimer les hautes fréquences et ensuite appliquer une transformée de Fourier inverse pour obtenir l'image réduite. De manière plus visuelle, on prend l'image que l'on veut réduire dans le domaine fréquentiel puis on place un rectangle centré ayant les dimensions de l'image que l'on veut obtenir à la fin (tracé en rouge), puis on garde uniquemenent les fréquences à l'intérieur de ce carré. On a alors la tranformée de Fourier de l'image réduite, il ne reste qu'a faire la tranformée de Fourier inverse.

FIGURE 15 – Image de départ

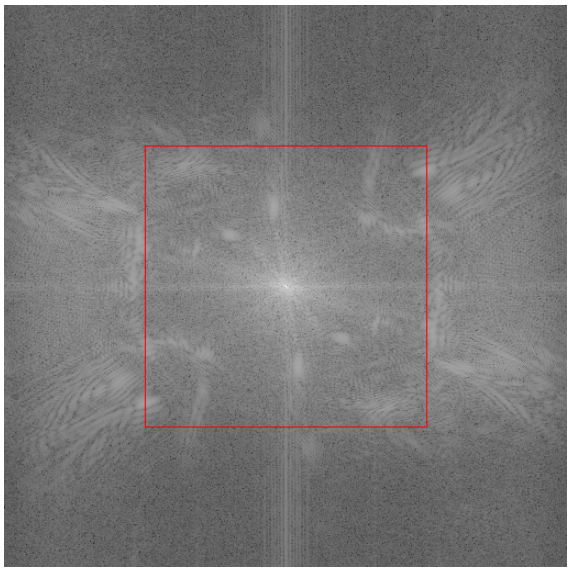


FIGURE 16 – Image de départ dans le domaine fréquentiel + carré rouge

FIGURE 17 – Image réduite sans aliasing

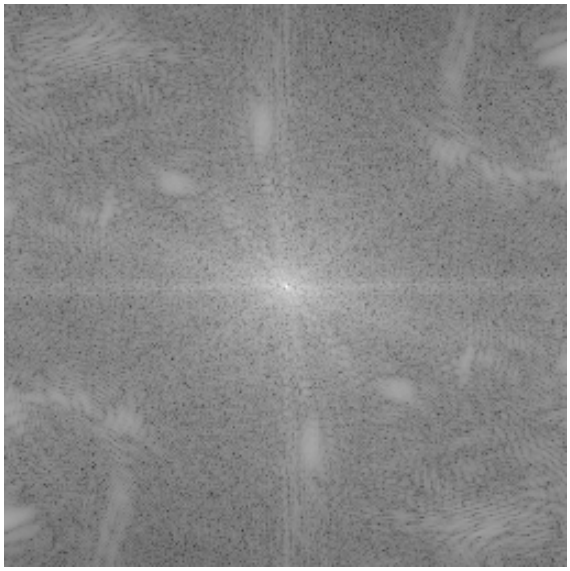


FIGURE 18 – Image réduite sans aliasing dans le domaine fréquentiel

5.5 Déconvolution

5.5.1 Définition du problème dans L^2, L^2

Beaucoup de filtres sont appliqués par une convolution. Nous avons une image de départ u_0 à laquelle nous appliquons un filtre A et nous obtenons l'image convolé v ; $v = A * u_0$. Le but dans cette partie est de déconvoler l'image. Une convolution est un calcul compliqué et pas directement réversible, on va donc utiliser la formule à minimiser suivante :

$$\tilde{u} = argmin_u ||A * u - v||_2^2 + \lambda ||\nabla u||_2^2.$$

C'est la même que celle utilisé pour le débruitage à l'exception du premier terme, où nous n'avons pas u tout seul mais $A * u$ puisqu'en effet on a appliqué le filtre A à l'image que l'on souhaite retrouver.

5.5.2 Résultion du problème et résultat

Par un développement très analogue à celui du débruitage détaillé en annexe, on obtient une expression explicite de $\tilde{u}$ dans le domaine fréquentiel :

$$\hat{\tilde{u}}(\zeta, \eta) = \frac{(\hat{A}^*.\hat{v})(\zeta, \eta)}{|\hat{A}|^2 + \lambda\,(\zeta^2 + \eta^2)}$$

Nous allons effectuer 2 tests de cette formule de déconvolution sur un filtre flou gaussien centré défini par la fonction

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp(-\frac{x^2 + y^2}{2\sigma^2})$$

On prendra $\sigma = 2.5$ puis $\sigma = 4$ ainsi que $\lambda \simeq 10^{-6}$.

FIGURE 19 – Image flouté $\sigma = 2.5$

FIGURE 20 – Image déflouté $\sigma = 2.5$

FIGURE 21 – Image flouté $\sigma = 4$

FIGURE 22 – Image déflouté $\sigma = 4$

Ici nous avons traité le défloutage d'une image dans un cas de perfection, cependant, ce qu'il se passe en réalité est très loin de ce que nous avons vu et nous obtenons des résultats étant assez loin de ce que nous avons obtenu ici. Puisque lorsque nous prenons une photo, le flou apparait avec l'appareil et un bruit s'ajoute après lorsque les rayon lumineux arrivent sur le capteur. Ainsi en réalité lorsque nous défloutons des images, nous défloutons des images un peu bruitée, car le bruit arrive après le flou. Et c'est ici que vient l'intérêt du lambda dans la formule ci-dessus. Ici le lambda est lié au niveau de bruit d'une image, c'est pour cela que nous avons pris $\lambda \simeq 10^{-6}$ car il n'y avait pas de bruit sur cette image, ainsi nous n'avons pas besoin de régularisation. Cependant lorsque l'image flou contient du bruit, nous avons besoin de ce lambda (qui est le coefficient devant le terme de régularisation), puisque nous avons besoin de régulariser l'image.

5.5.3 Modélisation et suppression d'un flou de bougé

Maintenant prenons un exemple qui se rapproche plus de la réalite. Le flou de bougé arrive fréquemment en photographie. Lorsque que le photographe appuie sur le déclencheur, il y a un laps de temps appelé le temps d'obturation pendant lequel la lumière peut aller jusqu'au capteur et est captée. Seulement si pendant cette courte durée le photographe bouge, il se produit un flou de bougé. De plus comme à chaque fois qu'on prend une photo, il y a un bruit parasite qui vient s'ajouter à l'image obtenu. On peut modéliser l'image obtenu par le photographe que l'on note v comme l'image idéal u_0 (sans flou de bougé ni rien) convolé avec un filtre (caractérisant le flou de bougé) noté A et auquel on a ajouté un bruit aléatoire ε où ε suit un loi normal centrée.

$$v = u_0 * A + \varepsilon$$

FIGURE 23 – Image original

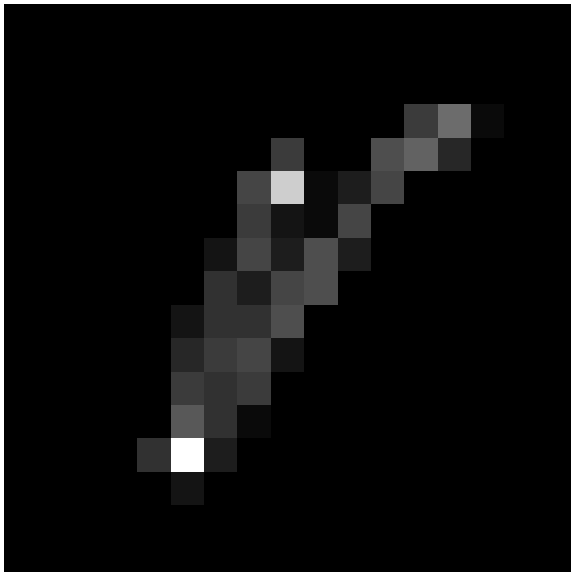


FIGURE 24 – Filtre A de flou de bougé

Si on calcule notre image ayant subie un flou de bougé modélisé par ce filtre l'image du dessus qui ensuite par application de la formule de déconvolution nous donne l'image ci-contre.

FIGURE 25 – Image flouté par le filtre A puis bruité

FIGURE 26 – Image obtenu après déconvolution

Ce résultat est plutôt satisfaisant seulement pour supprimer le flou de bougé de cette image il faut au préalable connaître le flou appliquée. Supprimer un flou "à l'aveugle" sans connaître le flou qui a été appliqué est un problème très compliqué qui ne sera donc pas abordé

6 Annexes

A Démonstration formule de débruitage L2 - L2

u est l'image bruitée de taille (N, M), on peut la modéliser comme l'image sans bruitage u_0 à laquelle nous avons ajouté du bruit ε, où $\varepsilon \sim U(0, \sigma^2)$, $u = u_0 + \varepsilon$. Le problème revient à chercher une image v qui se rapproche de u_0 donné par la formule.

$$\underset{v}{\operatorname{argmin}} \ \|u - v\|_2^2 + \lambda \|\nabla v\|_2^2$$

Vu que $\|\nabla v\|_2^2 = \|\partial_x v\|_2^2 + \|\partial_y v\|_2^2$ avec $\partial_x v = v(x+1, y) - v(x, y)$ et $\partial_y v = v(x, y+1) - v(x, y)$

$$\begin{aligned}
\hat{\partial_x} v(\zeta, \eta) &= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \left(v(x+1, y) - v(x, y)\right) e^{-2ß\pi\left(\frac{\zeta x}{N} + \frac{\eta y}{M}\right)} \\
&= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} v(x+1, y) e^{-2ß\pi\left(\frac{\zeta x}{N} + \frac{\eta y}{M}\right)} - \sum_{x=-1}^{N-2} \sum_{y=0}^{M-1} v(x+1, y) e^{-2ß\pi\left(\frac{\zeta(x+1)}{N} + \frac{\eta y}{M}\right)} \\
&= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} v(x+1, y) e^{-2ß\pi\left(\frac{\zeta x}{N} + \frac{\eta y}{M}\right)} \left(1 - e^{2ß\pi \frac{\zeta}{N}}\right) \left.\begin{array}{l} + \sum_{y=0}^{M-1} v(0, y) e^{-2ß\pi\left(\frac{\zeta 0}{N} + \frac{\eta y}{M}\right)} \\ - \sum_{y=0}^{M-1} v(N, y) e^{-2ß\pi\left(\frac{\zeta N}{N} + \frac{\eta y}{M}\right)} \end{array}\right\} = 0 \\
&= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} v(x+1, y) e^{-2ß\pi\left(\frac{\zeta(x+1)}{N} + \frac{\eta y}{M}\right)} \left(1 - e^{2ß\pi \frac{\zeta}{N}}\right) e^{2ß\pi \frac{\zeta}{N}} \\
&= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} v(x+1, y) e^{-2ß\pi\left(\frac{\zeta(x+1)}{N} + \frac{\eta y}{M}\right)} \left(i e^{-ß\pi \frac{\zeta}{N}} \sin(\pi \frac{\zeta}{N})\right) e^{2ß\pi \frac{\zeta}{N}} \\
&= \underbrace{\sum_{x=0}^{N-1} \sum_{y=0}^{M-1} v(x+1, y) e^{-2ß\pi\left(\frac{\zeta(x+1)}{N} + \frac{\eta y}{M}\right)}}_{= \hat{v}(\zeta, \eta)} i e^{ß\pi \frac{\zeta}{N}} \sin(\pi \frac{\zeta}{N})
\end{aligned}$$

Ainsi $|\hat{\partial_x} v(\zeta, \eta)|^2 = |\hat{v}(\zeta, \eta)|^2 \sin^2(\pi \frac{\zeta}{N})$ et de la même manière $|\hat{\partial_y} v(\zeta, \eta)|^2 = |\hat{v}(\zeta, \eta)|^2 \sin^2(\pi \frac{\eta}{M})$.
Et donc d'après la formule de Parseval

$$\begin{aligned}
\|\nabla v\|_2^2 = \|\partial_x v\|_2^2 + \|\partial_y v\|_2^2 &= \frac{1}{NM} \left(\left\|\hat{\partial_x} v\right\|_2^2 + \left\|\hat{\partial_y} v\right\|_2^2\right) \\
&= \frac{1}{NM} \sum_{\zeta=0}^{N-1} \sum_{\eta=0}^{M-1} \underbrace{\left(\sin^2(\pi \frac{\zeta}{N}) + \sin^2(\pi \frac{\eta}{M})\right)}_{=: C_{\zeta, \eta}} |\hat{v}(\zeta, \eta)|^2
\end{aligned}$$

et de même $\|u - v\|_2^2 = \frac{1}{NM} \|\hat{u} - \hat{v}\|_2^2$. On obtient alors une nouvelle formulation plus explicite de v

$$\underset{v}{\operatorname{argmin}} \sum_{\zeta=0}^{N-1} \sum_{\eta=0}^{M-1} |\hat{u}(\zeta, \eta) - \hat{v}(\zeta, \eta)|^2 + \lambda C_{\zeta, \eta} |\hat{v}(\zeta, \eta)|^2$$

Le problème revient alors à vouloir minimiser une somme, et il se trouve que les termes de cette somme sont indépendants. On cherche a minimiser pour tout (ζ,η), $|\hat{u}(\zeta,\eta)-\hat{v}(\zeta,\eta)|^2+\lambda C_{\zeta,\eta}|\hat{v}(\zeta,\eta)|^2$. $\hat{v}(\zeta,\eta)$ et $\hat{u}(\zeta,\eta)$ sont complexe, on note alors pour simplifier, $\hat{v}(\zeta,\eta)=a+ib$ et $\hat{u}(\zeta,\eta)=c+id$.

$$\begin{aligned}|\hat{u}(\zeta,\eta)-\hat{v}(\zeta,\eta)|^2+\lambda C_{\zeta,\eta}|\hat{v}(\zeta,\eta)|^2 &= |\hat{u}(\zeta,\eta)|^2+|\hat{v}(\zeta,\eta)|^2-2Re(\overline{\hat{u}(\zeta,\eta)}\hat{v}(\zeta,\eta))+\lambda C_{\zeta,\eta}|\hat{v}(\zeta,\eta)|^2\\ &= a^2+b^2+c^2+d^2-2(ac+bd)+\lambda C_{\zeta,\eta}(a^2+b^2)\\ &= (a^2+b^2)(1+\lambda C_{\zeta,\eta})-2ac-2bd+c^2+d^2\\ &= f(a,b)\end{aligned}$$

Ce qui se résume a vouloir minimiser une fonction dans $\mathbb{R}^2$. f est connexe donc le minimum local que l'on va trouver sera le minimum global.

$$\begin{cases}\dfrac{\partial f}{\partial a}=2a(1+\lambda C_{\zeta,\eta})-2c=0\\ \dfrac{\partial f}{\partial b}=2b(1+\lambda C_{\zeta,\eta})-2d=0\end{cases} \quad i.e. \quad \begin{cases}a=\dfrac{c}{1+\lambda C_{\zeta,\eta}}\\ b=\dfrac{d}{1+\lambda C_{\zeta,\eta}}\end{cases}$$

Ainsi $\hat{v}(\zeta,\eta)=\dfrac{\hat{u}(\zeta,\eta)}{1+\lambda C_{\zeta,\eta}}=\dfrac{\hat{u}(\zeta,\eta)}{1+\lambda\left(\sin^2(\pi\frac{\zeta}{N})+\sin^2(\pi\frac{\eta}{M})\right)}\simeq\dfrac{\hat{u}(\zeta,\eta)}{1+\lambda\left(\zeta 2+\eta^2\right)}$. On obtient ensuite v en calculant la transformée de Fourier inverse de $\hat{v}$.

B Convergence en norme infini du polynome trigonométrique interpolateur

Lemme .1. *Soient $f \in C^k$ une fonction de $[0;2\pi]$ à valeur réele 2π-périodique et N un entier naturel, on pose $x(k) = f(2\pi\frac{k}{N})$. On a les résultats suivants :*

1. $c_n(f) = c_{-n}(f)$
2. $c_n(f) = o(\frac{1}{|n|^k})$
3. $\frac{\hat{x}(p)}{N} - c_p(f) = \sum_{q\in\mathbb{Q}^*} c_{p+Nq(f)} = o(\frac{1}{N^{k-1}})$

On a par transformation de Fourier inverse $x(p) = \frac{1}{N}\sum_{n=0}^{N-1} \hat{x}(n)e^{2i\pi p\frac{n}{N}}$ On défini alors g le polynôme trigonométrique sur $[0, 2\pi[$ qui interpole les $x(p)$, $g = \frac{1}{N}\sum_{n=0}^{N-1}\hat{x}(n)e_n = \frac{1}{N}\sum_{|n|<N/2}\hat{x}(n)e_n$ avec $e_n : t \longmapsto e^{int}$. On note $S_p(f)$ la p-ième somme somme partiel de f.

Theoreme .2. $\|f-g\|_\infty = o(\frac{1}{N^{k-2}})$

Démonstration. Par l'inégalité triangulaire : $\|f-g\|_\infty \leq \|f - S_{N/2}(f)\|_\infty + \|g - S_{N/2}(f)\|_\infty$ On a $\|f - S_{N/2}(f)\|_\infty = \left\|\sum_{|n|>N/2} c_n(f)e_n\right\|_\infty \leq \sum_{|n|>N/2}\|c_n(f)\|_\infty = \sum_{|n|>N/2} o(\frac{1}{|n|^k}) = o(\frac{1}{N/2^{k-1}}) = o(\frac{1}{N^{k-1}})$.

En ensuite $\|g - S_{N/2}(f)\|_\infty = \left\|\sum_{|n|<N/2}(\frac{\hat{x}(n)}{N} - c_n(f))e_n\right\|_\infty \leq \sum_{|n|<N/2}\left\|\frac{\hat{x}(n)}{N} - c_n(f)\right\|_\infty = \sum_{|n|<N/2} o(\frac{1}{|N/2|^{k-1}}) = o(\frac{1}{N^{k-2}})$.

Ainsi $\|f-g\|_\infty = o(\frac{1}{N^{k-2}})$. □

C Projet artistique : dessiner avec les séries de Fourier

C.1 Introduction

L'idée de notre projet artistique est d'utiliser les séries de Fourier pour interpoler une courbe en 2D fermée sur elle même pour ensuite la redessiner. Dans ce projet artistique nous avons utilisé différentes librairies en python pour effectuer une segmentation, c'est-à-dire récuperer les contours d'une image. Nous n'allons pas parler de cette aspect là car trop compliqué et trop eloigné de notre projet initial. Nous allons nous intéresser au dessin de cette courbe paramétrique comme un exemple d'application des séries de Fourier.

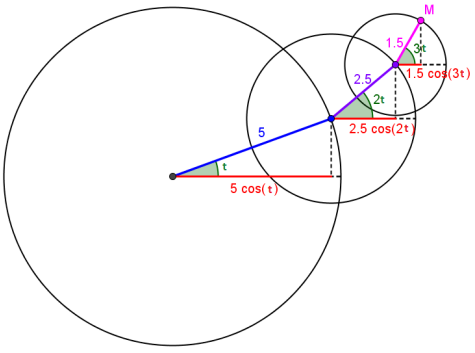
C.2 Décomposition d'une courbe paramétrique à l'aide de ses coefficients de Fourier

Nous pouvons considérer que le contour de l'image peut être décrite par une fontion f, avec f étant un arc paramétré continue 2π-périodique et C^1 par morceaux. c'est à dire $f(t) = (a(t), b(t))$ avec a et b deux fonctions numériques continues et 2π-périodique et C^1 par morceaux.

Ce que nous avons après traitement numérique de l'image n'est qu'une discrétisation de cette fonction (que l'on suppose à intervalle régulier), c'est-à-dire que nous avons N points sous la forme $M(k) = f(2\pi k/N) = (a(2\pi k/N), b(2\pi k/N))$. On note les coordonées de M, x et y et donc $M(k) = (x(k), y(k))$. Pour simplifier les calculs, nous allons passer dans le domaine complexe et considérer que f est complexe, et donc écrire : $f(t) = a(t) + ib(t)$ on fait de même pour les points $M(k)$ comme ceci $M(k) = x(k) + iy(k)$. On calcul ensuite la transformée de Fourier discrète de M, ce qui nous permet d'obtenir une approximation des coefficients de Fourier : $\forall p \in \{-N/2, ..., N/2\}, c_p(f) = \hat{M}(p \bmod N/2)$.

Ainsi nous avons une approximation des coefficients de Fourier de f : $c_p(f)$. On va alors les écrire sous la forme exponentielle : $c_p(f) = r_p e^{i\phi_p}$. Ensuite la somme partielle de la série de Fourier $S_n(t) = \sum_{|p|<n} c_p(f) e^{ipt}$ avec $0 < n < N/2$, s'ecrit : $S_n = \sum_{|p|<n} r_p e^{i\phi_p} e^{ipt} = \sum_{|p|<n} r_p e^{i(pt+\phi_p)} = \sum_{|p|<n} r_p(\cos(pt+\phi_p) + i\sin(pt+\phi_p))$

Repassons dans le plan $\mathbb{R}^2$, on a $S_n(t) = \left(\sum_{|p|<n} r_p \cos(pt+\phi_p), \sum_{|p|<n} r_p \sin(pt+\phi_p) \right)$ ce qui se réécrit : $S_n(t) = \sum_{|p|<n} r_p \left(\cos(pt+\phi_p), \sin(pt+\phi_p)\right)$.



Maintenant si on prend chaque terme de la somme indépendament, la fonction $P(t) = r_p\left(\cos(pt+\phi_p), \sin(pt+\phi_p)\right)$ est celle d'un cercle de rayon r_p et de phase ϕ_p. Quand t augmente on a le point $P(t)$ qui se déplace le long de ce cercle. Si maintenant on ajoute un second terme de la somme, le centre de ce second cercle sera au point $P(t)$ (qui se déplace) et ce second terme va lui décrire un autre cercle et ainsi de suite... Voici un exemple de cette construction avec les 3 premiers cercles.

De plus comme f est continue et C^1 par morceaux, d'après le théorème de convergence uniforme plus n est grand plus $S_n(f)$ va tendre vers f (en norme infini). Autrement dit, plus on ajoute de cercle à la suite plus notre construction va tendre à ressembler au dessin initial.

Références

[1] Maïtine Bergounioux. Quelques méthodes de filtrage en traitement d'image. 2011.

[2] Claude Gasquet and Patrick Witomski. *Analyse de Fourier et applications : filtrage, calcul numérique, ondelettes*. Dunod, 2000.

[3] Rafael C Gonzalez, Richard E Woods, et al. Digital image processing [m]. *Publishing house of electronics industry*, 141(7), 2002.

[4] Xavier Gourdon. *Les maths en tête : Analyse*. Ellipses, 2008.

[5] Elias M Stein and Rami Shakarchi. *Fourier analysis : an introduction*, volume 1. Princeton University Press, 2011.

[6] Yann Traonmilin. *Relations entre le modèle d'image et le nombre de mesures pour une super-résolution fidèle*. PhD thesis, Télécom ParisTech, 2014.