Université de Strasbourg
UFR de Mathématiques et d'Informatique

20 novembre 2014
Rapport final

Projet de 150 heures

Filtrage d'images en bases d'ondelettes

Julien Vogel

Encadrant : Basile Sauvage **Master 2 ISI**

Table des matières

Chapitre 1

Contextualisation

1.1 Le cadre

Mon projet a été effectué au sein du laboratoire *ICube*. Né en 2013 et situé à Illkirch, il est issu de la fusion entre quatre grands laboratoires de recherche affiliés au **CNRS** et à l'**Université de Strasbourg**. Regroupant près de 500 membres, ses domaines de prédilection sont nombreux et variés (informatique, robotique, biophysique, microélectronique, photonique, mécanique, traitement d'images...).

Ce projet a été effectué sous la tutelle de Monsieur *Basile SAUVAGE*, enseignant-chercheur dans l'équipe *IGG* (Informatique Géométrique et Graphique). Ce travail est essentiellement un projet de développement. Comme son nom l'indique, il doit être effectué pendant une durée effective de 150 heures et a lieu pendant le troisième semestre du Master Informatique et Sciences de l'Image. Il a pour but de mettre en pratique les nouvelles notions vues en cours durant le semestre tout en mettant à l'épreuve notre capacité à programmer dans un langage donné et avec des outils précis.

1.2 Le thème : les ondelettes

Les ondelettes sont des fonctions mathématiques permettant une décomposition du signal similaire à la transformée de Fourier. Leur première utilisation remonte à 1909, définies par Alfréd Haar, un mathématicien hongrois. Elles n'obtiendront le nom d'ondelettes qu'en 1984. Elles sont souvent appelées par leur dénomination anglaise, wavelets, qui veut dire «petites ondes». La base de Haar, qui est la base de ce projet, n'est qu'une des nombreuses bases pour la transformée en ondelettes discrète (la transformée continue existe également mais dépasse le cadre d'application de ce projet de 150h).

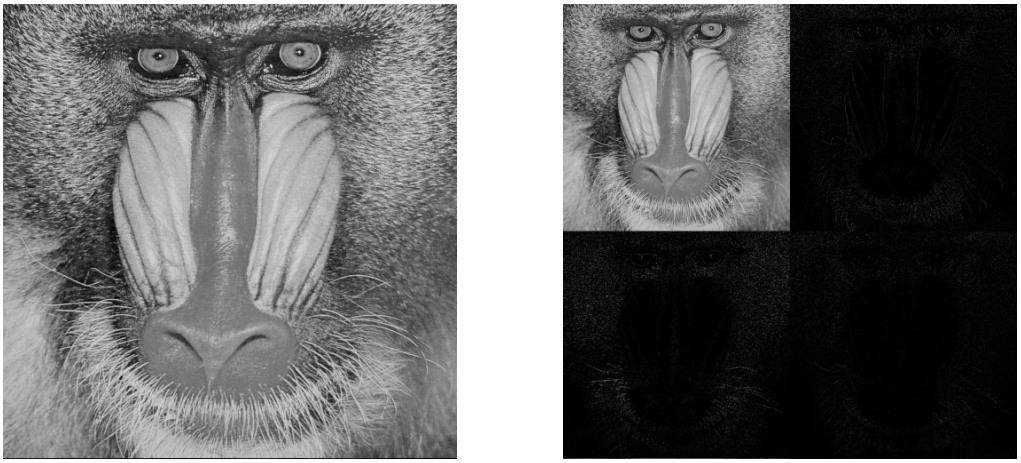
Les principales utilisations de la décomposition ou transformée en ondelettes discrète touchent à la compression de données numériques. On parle alors de compression par ondelettes. Celle qui nous intéressera par la suite est une transformée à deux dimensions utilisée notamment par le format d'image JPEG 2000 et va constituer en 4 étapes principales qui vont donner naissance à ce que l'on appelle un niveau d'analyse.

1. Pour chaque ligne de l'image, on va moyenner les coefficients des pixels P_i deux à deux (dans notre cas, il s'agira de niveaux de gris), ce qui correspond à un filtre passe-bas, les résultats seront appelés coefficients d'approximation :
$$A(x) = \frac{X_n + X_{n+1}}{2}$$
2. Les coefficients de détail correspondant à un filtre passe-haut et seront donnés par l'opération :
$$D(x) = \frac{X_n - X_{n+1}}{2}$$

Les coefficients d'approximation seront rangés dans la moitié gauche de la ligne, les coefficients de détail à droite.

3. Mêmes calculs que 1. sur les colonnes
4. Mêmes calculs que 2. sur les colonnes

Pour obtenir le niveau d'analyse suivant (voir figure 1.1), il suffira de réitérer ces opérations en divisant à chaque fois la longueur des lignes et colonnes à parcourir par deux, c'est pourquoi nous travaillerons essentiellement sur des images dont la hauteur et la largeur sont des puissances de 2. La succession des opérations inverses à celles citées ici permettra de retrouver l'image originale. Cette succession d'opérations inverses s'appelle la synthèse.



(a) Image d'origine (b) Premier niveau d'analyse

FIGURE 1.1 – Application de la transformée en ondelettes sur une image. L'image de droite représente une itération des quatre étapes décrites précédemment, les quatre blocs la composant sont appelés (de gauche à droite et de haut en bas) : image basse-résolution, détails verticaux, détails horizontaux et détails diagonaux

Chapitre 2

Analyse du besoin et méthodologie

2.1 Problématique et objectifs

À l'heure actuelle, l'équipe *IGG* travaille sur un projet de génération de textures infinies à la volée. Ses membres cherchent un moyen de transformer les images donnant naissance aux futures textures sans changer leur structure mais de sorte à créer des variations aléatoires de leur motifs, afin d'éviter le copier-coller bout à bout des mêmes textures comme on peut le voir dans les jeux vidéo de manière générale. L'équipe aimerait pouvoir expérimenter des transformations sur des versions décomposées en ondelettes de ces images et aurait besoin d'un outil efficace pour observer les résultats et les comparer à l'image d'origine. En effet, les ondelettes pourraient être une alternative intéressante à la transformée de Fourier discrète, utilisée pour l'instant (la transformée de Fourier discrète représentant le signal dans une base de sinusoïdes uniquement).

L'objectif premier sera de comprendre et implémenter la transformée en ondelettes. On s'intéressera principalement à la base de Haar qui est la plus accessible en terme d'algorithmes et de fondements mathématiques. La grosse partie du travail consistera ensuite en la conception et la programmation d'une interface graphique qui permettra de charger une image, la transformer en ondelettes, lui appliquer un ou plusieurs traitements (exemple : filtres) puis effectuer une transformée en ondelettes inverse. L'interface devra permettre de visualiser simultanément ces quatre étapes.

2.2 Organisation du projet

L'organisation pour ce projet peut s'apparenter à une méthode agile. À raison d'un rendez-vous par semaine avec mon encadrant, je lui montrais le travail effectué depuis la fois dernière et nous réfléchissions ensemble à l'ensemble des fonctionnalités et améliorations qui pourraient être apportées en l'espace d'une semaine. La stratégie consistait à implémenter certaines fonctionnalités importantes tout en pensant déjà aux suivantes afin d'obtenir au plus vite une application exploitable. C'est pour cette raison que j'ai consacré mon projet au travail sur les ondelettes plutôt qu'aux filtres qui seront appliqués pendant les tests de M. Sauvage, qui choisira lui-même lesquels implémenter le moment venu.

Pour connaître l'état de l'avancement, nous avons décidé ensemble de mettre à notre disposition un dépôt sur *Github* qui permettrait non seulement à mon encadrant de tester plus rapidement le programme et ses dernières fonctionnalités ou de trouver des erreurs qui m'auraient échappées, mais aussi de pouvoir générer presque automatiquement le calendrier de travail se trouvant en annexe de ce rapport. Chaque modification du code qui menait à une mise-à-jour du dépôt distant était accompagné d'un mail récapitulatif à M. Sauvage.

Chapitre 3

Choix d'implémentation

Le sujet indiquait que la programmation devait s'effectuer en langage *C++* à l'aide du framework *Qt*. J'ai donc opté pour un travail dans l'environnement de développement intégré *Qt Creator* avec lequel j'ai l'habitude de travailler depuis le début du Master. N'étant pas très à l'aise avec l'architecture Modèle-Vue-Contrôleur, j'ai opté pour une programmation orientée objet basée sur 2 classes, la première nommée WorkSpace va gérer la partie calculs et la seconde nommée MainWindow, l'affichage.

3.1 Structure de données

Afin de répondre à la problématique, il faut pouvoir effectuer un certain nombre d'opérations sur les images. Comme nous avons pu le voir dans la partie précédente, la transformée en ondelettes fait appel à des moyennes et des soustractions, il arrive alors que certains coefficients après calculs soient négatifs. Ceci nous pose un problème en terme de visualisation, des niveaux de gris négatifs n'ayant aucun sens pour *Qt*.

Mon idée a alors été de créer une classe qui ferait office de station de travail (workspace en anglais) sur des matrices de coefficients propres à chaque étape. Elles sont au nombre de 4 : l'image de départ, sa transformée en ondelettes, la transformée en ondelettes sur laquelle on a appliqué un ou plusieurs filtres et sa transformée inverse, la synthèse. Cette dissociation permet alors de séparer les matrices de coefficients de leur représentation visuelle pour laquelle j'ai pu choisir un moyen d'afficher ces valeurs négatives. Ramener ces coefficients à zéro lors de l'affichage va provoquer l'apparition de grosses taches noires sur l'image générée, c'est pourquoi j'ai choisi de leur appliquer une valeur absolue.

En résumé, il s'agit de l'image d'entrée et l'image de sortie et leur base de Haar respective. Pour éviter les arrondis lors des moyennes et soustractions, il était préférable que ces matrices contiennent des nombres flottants. Il s'agit au final de tableaux de flottants à 2 dimensions créés de manière dynamique afin de faciliter leur passage dans les fonctions et/ou l'interface graphique.

L'interface graphique aura besoin d'avoir accès à toutes les informations en temps réel, plusieurs options étaient envisageables :

- Passer une instance de la classe WorkSpace en paramètre de chaque fonction se trouvant dans la classe MainWindow
- Créer un objet de type WorkSpace dans les attributs de la classe MainWindow
- Créer un singleton dans la classe WorkSpace

J'ai finalement opté pour le singleton qui permet d'éviter plusieurs instanciations et évite le passage de la même instance dans chaque méthode de la seconde classe. Je n'ai pas gardé la seconde option car je voulais vraiment dissocier les éléments de la fenêtre des données sur lesquelles on travaille.

3.2 Interface graphique

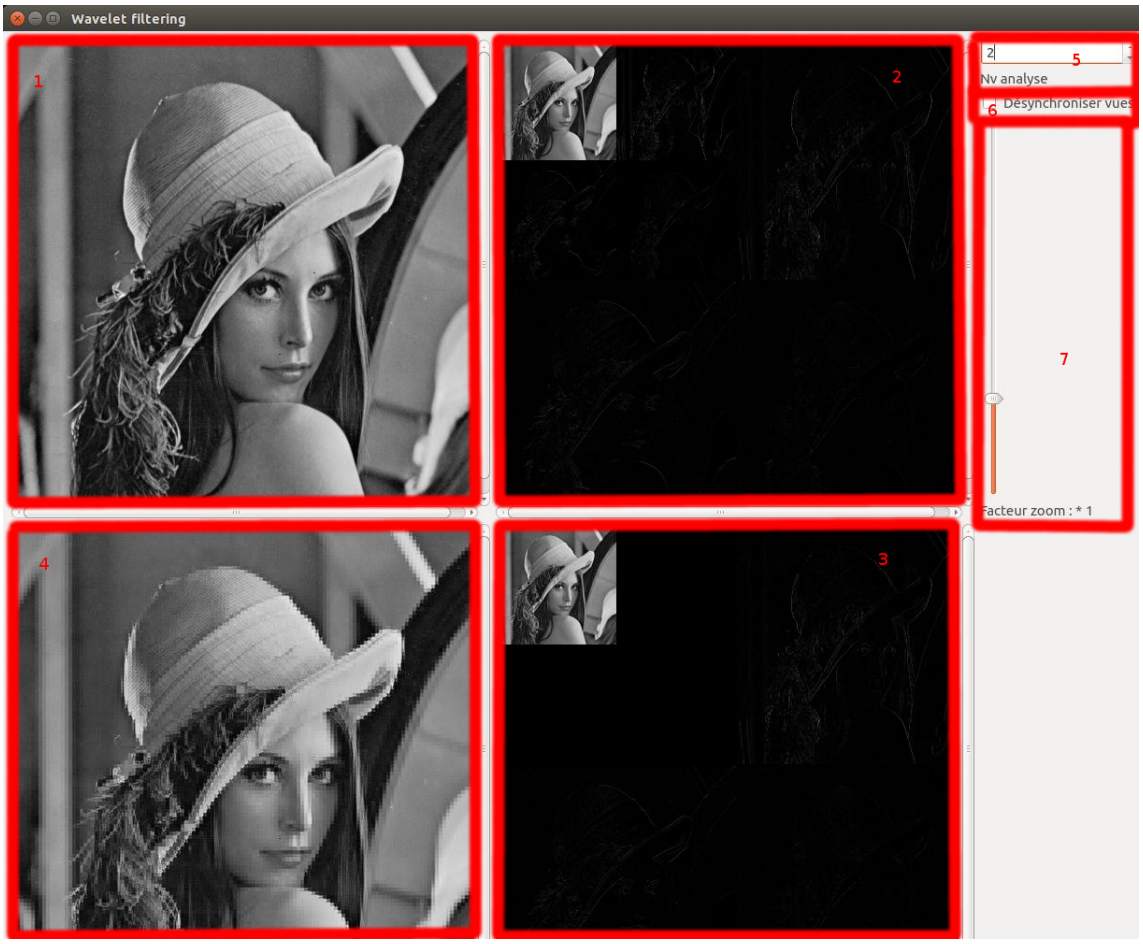


FIGURE 3.1 – Capture d'écran de l'interface.
(1) Image d'entrée, (2) Transformée en ondelettes, (3) Transformée en ondelettes + filtre, (4) Image de sortie, (5) Sélecteur du niveau d'analyse, (6) Case à cocher pour synchroniser ou désynchroniser les vues et (7) Zoom

L'interface graphique est relativement basique : il s'agit d'une seule fenêtre à laquelle vont se greffer plusieurs menus. Le premier menu regroupe les habituelles fonctions de chargement, sauvegarde et fermeture du programme. La nécessité d'appliquer une succession de filtres m'a amené à programmer une autre fonction pour ce menu, son rôle est de créer une nouvelle image d'entrée à partir de l'image de sortie courante afin d'enchaîner les transformations. Le second menu, intitulé "transformations", va contenir les différentes opérations de traitement d'images (filtres, seuillage. . .).

Pour ce qui est de l'interface elle-même, j'ai décidé de la diviser également en plusieurs parties à savoir une partie vue à gauche, et une partie contrôleur à droite. On retrouve dans la partie vue un rendu des 4 images provenant des 4 matrices dont je parlais dans la structure de données. En haut, l'entrée et en bas, la sortie. À droite, leur base d'ondelettes et à gauche leur version fine ou reconstituée (voir figure 3.1).

Dans la partie contrôleur, on va pouvoir choisir le niveau d'analyse de la transformée en ondelettes de l'entrée et de la sortie à l'aide d'une spinbox. Un slider représente le zoom qui permet de s'intéresser à une sous-partie de l'image de départ/d'arrivée ou encore de s'éloigner lorsque l'on a affaire à de grandes images. Ensuite, une case à cocher permet de désynchroniser les vues de gauche

et de droite afin de pouvoir étudier le comportement de chacune indépendamment de l'autre. Pour ce faire, j'ai implémenté deux types de zooms, le premier étant un zoom basique. Le second, en revanche est utilisé lorsque l'on désynchronise les vues. Il va permettre, lorsque l'on zoome sur une partie de l'image fine, d'observer cette même partie dans chaque sous-arbre de la base de Haar lui correspondant. Par conséquent, il a fallu implémenter un accès à chaque sous-arbre, ou bloc, ce qui correspond aux détails horizontaux, verticaux, diagonaux et à l'image basse-résolution décrits dans la partie contextualisation. La figure 3.2 illustre ce second type de zoom.

FIGURE 3.2 – Illustration du zoom lors de la désynchronisation des vues : dans l'exemple on observe le visage de "Lena" dans chaque bloc (ou sous-arbre)

Chapitre 4

Bilan du projet

4.1 Résultats obtenus

On peut dire que les tâches demandées ont été traitées en temps et en heure. Le programme est entièrement fonctionnel. Le code prend en compte les fichiers d'image .jpg et .bmp qui ont été utilisés pendant les premiers tests. Notre objectif étant de stocker des données sans pertes, j'ai également rendu possible la lecture des images .png qui répondent bien à cette problématique.

L'interface est assez basique et de ce fait, simple d'utilisation. Les menus vont permettre de charger, sauvegarder et transformer une image. Les widgets à droite des vues, quant à eux, vont permettre de changer le niveau d'analyse des transformées en ondelettes de manière instantanée, et ce au moins pour des images de tailles inférieures ou égales à 1024x1024 pixels. Les autres widgets permettront de zoomer de deux manières sur les quatre vues ou de synchroniser ou non les vues en question.

À noter également que grâce à la structure du code source, j'ai rendu possible et facile d'ajouter diverses opérations de traitement d'images au menu déjà en place. Une documentation *Doxygen* est disponible afin de passer le relais à d'éventuels successeurs.

4.2 Améliorations possibles

Une extension possible serait d'essayer d'implémenter une autre base d'ondelettes pour comparer les résultats obtenus. Cependant le temps de prise en main de la base de Haar, de son implémentation et des fonctions qui y sont liées donnent une idée du temps nécessaire à la compréhension et la programmation de bases plus complexes.

D'autres opérations de traitement d'images pourraient également être ajoutées afin de multiplier les possibilités du programme actuel. Leur code pourra aisément être greffé au reste du projet.

Finalement, il serait intéressant de pouvoir prendre en main d'autres formats d'images, notamment ceux qui permettent une compression sans perte (.tif par exemple en raison du grand nombre de bibliothèque d'images libres trouvables sur le net) mais qui demandent d'ajouter des plugins à *Qt* pour être reconnus.

Annexe A

Calendrier du travail effectué

Dates	**Tâches effectuées**
Début octobre 2014	Lecture du sujet et prise de contact avec mon encadrant
9 octobre 2014	Création du dépôt Git et recherches sur les ondelettes
11 octobre 2014	Début du développement
15 octobre 2014	Implémentation de la base de Haar
19 octobre 2014	Création de la structure de données (contenant au départ une QList de matrices)
24 octobre 2014	Structure de données plus légère, transformée en ondelettes inverse et premier filtre en place
27 octobre 2014	Mise en place de l'interface graphique
2 novembre 2014	Première version du zoom et rédaction d'une documentation *Doxygen*
9 novembre 2014	Possibilité de récupérer un bloc bien précis dans l'arborescence des niveaux d'analyse et de remonter la dernière image obtenue en vue d'enchaîner les opérations de filtrage
11 novembre 2014	Nouvel algorithme fonctionnel de zoom sur chaque bloc
15 novembre 2014	Refactoring du code source et suppression des données obsolètes
18 novembre 2014	Synchronisation entre les vues et zoom négatif pour les grandes images
21 novembre 2014	Début de la rédaction du rapport final et débug des problèmes de synchronisation
5 décembre 2014	Début de la rédaction des diapositives en vue de la soutenance finale
17 décembre 2014	Soutenance finale du projet de 150h

Annexe B

Bibliographie

— **Sujet de ce Projet 150h**
https://moodle2.unistra.fr/pluginfile.php/318343/mod_resource/content/1/14_filtrage_ondelettes_M2_dev.pdf

— **Cours sur les ondelettes par *V. PERRIER***
http://www-ljk.imag.fr/membres/Valerie.Perrier/PUBLI/Cours4-VP.pdf

— **Vidéo associée au sujet**
http://youtu.be/YOteWlIJ7gM

— **Le site du laboratoire *ICube***
https://icube.unistra.fr/