



École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5^e année
2014 - 2015

Rapport de projet de fin d'études

**Développement d'un outil de traitement
d'images par filtrage bilatéral**

Encadrants

Moncef HIDANE
moncef.hidane@insa-cvl.fr

Étudiante

Natacha MARLIO-MARETTE
natacha.marlio-marette@etu.univ-tours.fr

Université François-Rabelais, Tours

DI5 2014 - 2015

Version du 8 mai 2015

Table des matières

1	Introduction	7
2	Filtre bilatéral	8
2.1	Définition	8
2.1.1	Filtre bilatéral : définition	8
2.1.2	Notations et terminologies	8
2.1.3	Formulation du filtre Gaussien	8
2.1.4	Formulation du filtre bilatéral	9
2.2	Implémentation	10
2.2.1	Algorithme général	10
2.2.2	Image RGB et luminance	10
2.3	Applications	11
3	Utilisation du filtre bilatéral en débruitage	15
3.1	Mise en place	15
3.2	Tests et résultats	16
4	Pyramide multi-échelle à partir du filtre bilatéral	21
4.1	Décomposition et recomposition des images	21
4.1.1	Méthodes	22
4.1.2	Stratégies	22
4.1.3	Exemples de décompositions et recompositions	22
4.2	Manipulation des détails	25
4.2.1	Mise en place	25
4.2.2	Exemples de manipulation des détails : rehaussement	26
4.2.3	Exemples de manipulation des détails : atténuation	31
5	Interface graphique	35
5.1	Fonctionnalités et structure du système	35
5.1.1	Cas d'utilisation	35
5.1.2	Structure générale	35
5.2	Guide d'utilisation	37
6	Accélération	39
6.1	Utilisation du filtre bilatéral comme une convolution	39
6.1.1	Rappels	39
6.1.2	Approche	39
6.1.3	Intensité homogène	39
6.1.4	Convolution	40
6.2	Implémentation de Paris et Durand	41
6.3	Implémentation de Clmg	42
6.4	Comparaison entre les différents filtres	42



7 Conclusion	44
A Ressources complémentaires	45
Index	48
Bibliographie	48

Table des figures

2.1	Exemple de filtre gaussien	9
2.2	Comparaison de filtres en débruitage	12
2.3	Reconstruction d'une image à partir de photos avec des expositions différentes	13
2.4	Combinaison d'une image avec flash et d'une image sans flash.	13
2.5	Lissage d'un objet 3D à l'aide du filtre bilatéral	14
2.6	Effet "cartoon" d'une vidéo	14
3.1	Images utilisées pour les tests	16
3.2	Comparaison avec une image RGB des deux implémentations en faisant varier σ_r	17
3.3	Comparaison avec une image en niveau de gris(NdG) des deux implémentations en faisant varier σ_r	18
3.4	Comparaison avec une image RGB des deux implémentations en faisant varier σ_s	19
3.5	Comparaison avec une image en niveau de gris(NdG) des deux implémentations en faisant varier σ_s	20
4.1	Combinaison vectorielle de la décomposition d'une image	21
4.2	Décomposition entre la couche de base et la couche de détails	22
4.3	Décomposition pyramidale (méthode 1 et stratégie 2 - σ_r divisé par 2), paramètre de départ $\sigma_s=36$ et $\sigma_r=100$	23
4.4	Décomposition pyramidale (méthode 2 et stratégie 2 - σ_r divisé par 2), paramètre de départ $\sigma_s=36$ et $\sigma_r=100$	24
4.5	Décomposition pyramidale (méthode 1 et stratégie 1 - facteur 2), paramètre de départ $\sigma_s=4$ et $\sigma_r=10$	25
4.6	Réhaussement des détails avec $\alpha=0.8$ et $\beta=1$	26
4.7	Réhaussement des détails avec $\alpha=0.8$ et $\beta=3$	27
4.9	Réhaussement des détails avec $\alpha=0.8$ et $\beta=1$	27
4.8	Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$	28
4.10	Réhaussement des détails avec $\alpha=0.8$ et $\beta=3$	28
4.11	Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$	29
4.12	Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$	29
4.13	Réhaussement des détails avec $\alpha=0.8$ et $\beta=1$	30
4.14	Réhaussement des détails avec $\alpha=0.8$ et $\beta=3$	30
4.15	Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$	31
4.17	Atténuation des détails avec $\alpha=1$ et $\beta=0.5$	32
4.16	Atténuation des détails avec $\alpha=1$ et $\beta=0.8$	32
4.18	Atténuation des détails avec $\alpha=1$ et $\beta=0.8$	33
4.19	Atténuation des détails avec $\alpha=1$ et $\beta=0.5$	33
4.20	Atténuation des détails avec $\alpha=1$ et $\beta=0.8$	34
4.21	Atténuation des détails avec $\alpha=1$ et $\beta=0.5$	34
5.1	Diagramme des cas d'utilisations	35
5.2	Composition d'une fenêtre QMainWindow	36
5.3	Composition du widget central : WidgetImage	36



5.4	Visualisation de l'interface après chargement de l'image	37
5.5	Visualisation de l'interface après traitement de l'image	38
6.1	Processus appliqué à un signal 1D	41
6.2	Comparaison sur un jeu d'images des différents filtres	42
A.1	Graphe de collaboration de la classe FilterBilateral	45
A.2	Graphe de collaboration de la classe MainWindow	46

Introduction

Ce projet de fin d'études a été proposé par Mr. Moncef HIDANE, de l'INSA de Blois, en collaboration avec le laboratoire RFAI¹ de Polytech Tours. Il porte sur la manipulation des détails d'une image en utilisant un filtre bilatéral permettant de lisser une image tout en conservant les contours des objets de celle-ci. L'objectif est de réaliser une interface permettant à l'utilisateur de pouvoir modifier les détails d'une image soit en les atténuant soit en les réhaussant.

On peut décomposer ce projet en trois grandes parties. La première partie consiste en la mise en oeuvre d'une méthode de manipulation des détails d'une image par décomposition obtenue par filtrage bilatéral itératif. La deuxième partie porte sur la conception et implémentation d'une interface. Et enfin la dernière partie consistera à étudié une approximation rapide basé sur l'interprétation du filtre bilatéral comme une convolution et à l'intégrer dans la partie précédente.

Tout d'abord sera expliqué en quoi consiste le filtre bilatéral, ces différentes utilisations ainsi que l'implémentation qui en a été faîte. La partie suivante décrira comment cette implémentation du filtre a été validé. Par la suite, je décrirais comment j'ai décomposé puis recomposé une image afin de pouvoir en manipuler ces détails à différents niveaux. De cette partie va découler la description de l'interface, c'est-à-dire comment elle a été conçue et son utilisation. Puis pour finir, le dernier chapitre est consacré à une méthode permettant d'accélérer l'exécution du filtre en changeant d'espace et utilisant une convolution gaussienne.

Je tiens à remercier mon encadrant Mr. Moncef HIDANE pour sa disponibilité même lorsqu'il n'était pas présent à l'école.

1. RFAI : Reconnaissance des formes et Analyse d'images

Filtre bilatéral

2.1 Définition

2.1.1 Filtre bilatéral : définition

Le filtre bilatéral est un filtre non linéaire qui lisse les petites variations d'intensité et préserve les contours des objets. Il supprime les textures, bruits et petits détails tout en conservant les formes des objets mais sans flouter. Cette technique remonte à 1995 avec le travail de Aurich et Weule [1] sur les filtres gaussien non linéaires. Cela a été redécouvert quelques années plus tard, en 1997, par Smith et Brady [2], puis en 1998 par Tomasi et Manduchi [3] qui lui ont donné son nom actuel.

La formulation de ce filtre est relativement simple. Chaque pixel est remplacé par une moyenne pondérée de ses voisins. Il ne dépend que deux paramètres qui indiquent la taille et le contraste des caractéristiques à conserver.

2.1.2 Notations et terminologies

Soit I une image en niveau de gris mais chaque opération du filtre peut être dupliqué sur chaque composante dans le cas d'une image en couleur. On note I_p la valeur du pixel à la position p . $F[I]$ désigne la sortie du filtre F appliqué à l'image I . On considère S l'ensemble de toutes les emplacements possibles de l'image que l'on nomme le domaine spatial et R l'ensemble de toutes les valeurs des pixels possibles que l'on nomme le domaine d'intensité. La notation $\sum_{p \in S}$ correspond à la somme de tous les pixels de S . On utilise $|\cdot|$ pour la valeur absolue et $\|\cdot\|$ pour la norme c'est à dire que $\|p - q\|$ représente la distance euclidienne entre la position des pixels p et q .

2.1.3 Formulation du filtre Gaussien

La façon la plus simple de lisser une image est sûrement de la flouter (voir image 2.1). La valeur de chaque pixel est remplacée par une somme pondérée de ses voisins. Le filtre gaussien est principalement basé sur la convolution gaussienne et utilise une fonction gaussienne $G_\sigma(x)$ (eq.(2.2)). Une image filtrée par convolution gaussienne est donnée par la formule suivante :

$$GC[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|) I_q \quad (2.1)$$

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) \quad (2.2)$$

Avec ce filtre, on calcule la moyenne pondérée de l'intensité des pixels adjacents avec un poids qui décroît en fonction de la distance avec le pixel p . Le poids du pixel q est défini par la fonction gaussienne $G_\sigma(\|p - q\|)$ où σ est un paramètre qui définit la taille du voisinage. La valeur des pixels n'a pas d'influence dans la moyenne pondérée seulement leur position. La convolution gaussienne est indépendante du contenu de l'image.

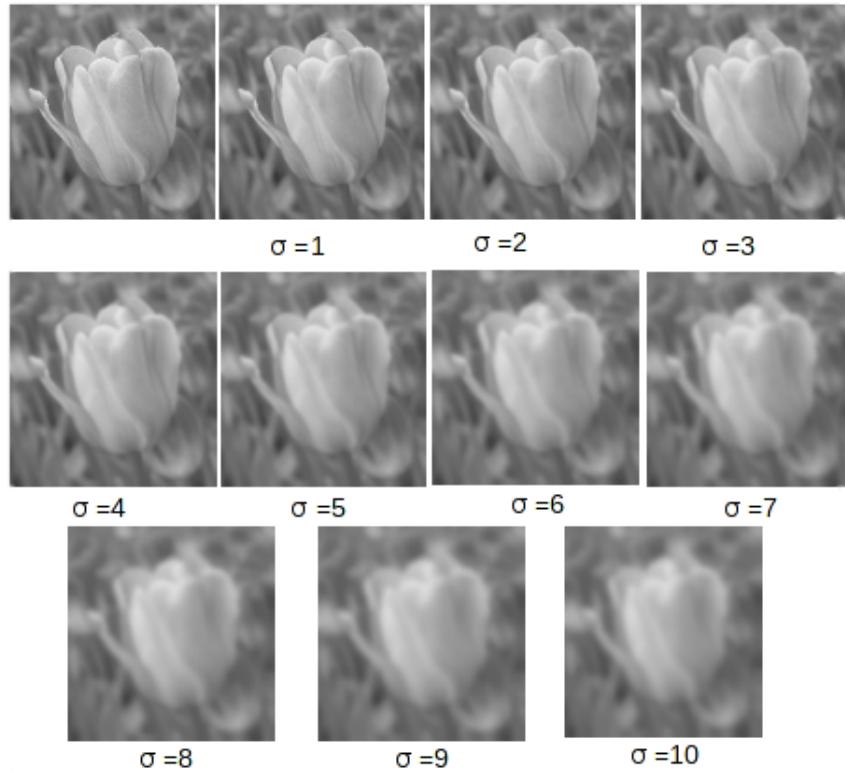


FIGURE 2.1 – Exemple de filtre gaussien avec différentes valeur de σ . Image originale en haut à gauche

2.1.4 Formulation du filtre bilatéral

Le filtre bilatéral est défini d'une manière similaire au filtre gaussien comme une moyenne pondérée des pixels environnants, à la différence que celui-ci tient compte des valeurs des pixels afin de préserver le contour des formes. Pour qu'un pixel influence un autre pixel, il ne doit pas seulement être proche spatialement mais aussi proche en intensité. Ce filtre, noté $BF[\cdot]$, est défini par :

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(|I_p - I_q|) I_q \quad (2.3)$$

où W_p est un facteur de normalisation :

$$W_p = \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(|I_p - I_q|) \quad (2.4)$$

Les paramètres σ_s et σ_r définissent le niveau de filtrage de l'image. L'équation (2.3) définissant le filtre est une moyenne pondérée normalisée où G_{σ_s} est une pondération gaussienne spatiale qui diminue l'influence des pixels distants et G_{σ_r} est une gausienne d'intensité qui diminue l'influence des pixels q si leur intensité diffère de celle de I_p .

Plus le paramètre σ_r est grand plus le filtre s'approche d'une convolution gaussienne c'est à dire du filtre gaussien.

2.2 Implémentation

2.2.1 Algorithme général

L'influence de pixel éloigné spatialement diminuant, on considère que l'ensemble S est un masque carré avec pour centre le pixel p . Ce masque est relativement petit, ici dans nos implémentations, il ne fait que 21x21 pixels de large. Il n'y a pas besoin de parcourir l'image entière dans le but d'obtenir la moyenne pondérée car plus les pixels sont éloignés spatialement moins ils ont d'influence. On peut donc considérer qu'ils sont négligeables. Si le masque doit déborder de l'image, les pixels étant en dehors de celle-ci seront considérés comme nul.

Entrées : Image I

Sorties : Image filtrée $BF[I]$

```

pour  $p \in I$  faire
    // Initialisation
     $BF[I]_p = 0$ 
     $W_p = 0$ 
    pour  $q \in S$  faire
         $w = G_{\sigma_s}(\| p - q \|)G_{\sigma_r}(| I_p - I_q |)$ 
         $BF[I]_p += wI_q$ 
         $W_p += w$ 
    fin
    // Normalisation
     $BF[I]_p = I_p/W_p$ 
fin
```

Algorithme 1 : Algorithme général du filtre bilatéral

2.2.2 Image RGB et luminance

L'algorithme (alg 1) précédent peut être appliqué à des images en couleur. Pour cela, il faut itérer le filtre pour chaque composante couleur de l'image. La gaussienne d'intensité G_{σ_r} ne prendra compte que de la composante couleur en cours dans son calcul pas des autres.

Dans le but d'optimiser l'exécution du filtre, on peut passer par la luminance, cela permet d'éviter de devoir parcourir chaque canal de couleur de l'image. La luminance correspond à la sensation visuelle de luminosité d'une image. C'est une grandeur photométrique qui est dépendante de l'œil humain. Il existe différents espaces colorimétriques prenant en compte la luminance tel que CIE XYZ, xyY ou YCbCr. Dans ces espaces, la luminance est représentée par la variable Y . Dans notre implémentation du filtre bilatéral pour des images RGB, l'espace colorimétrique YCbCr est utilisé.

L'espace YCbCr est principalement utilisé pour la vidéo que celle-ci soit en noir et blanc ou bien en couleur. Y représente le signal de luminance (noir et blanc). À ce signal, on ajoute deux composantes : Cb (le bleu moins Y) et Cr (le rouge moins Y). Il est très facile de passer d'un espace YCbCr à un espace RGB. La seule composante qu'il manque afin de changer d'espace est le vert. Cette composante peut être retrouvée mathématiquement avec $Y(\text{rouge} + \text{vert} + \text{bleu})$, Cb et Cr .

Afin de passer des composantes RGB d'une image à celle de YCbCr, on utilise les formules suivantes ¹ :

$$Y = \frac{66R + 129G + 25B + 128}{256} + 16 \quad (2.5)$$

$$Cb = \frac{-38R - 74G + 112B + 128}{256} + 128 \quad (2.6)$$

$$Cr = \frac{112R - 94G - 18B + 128}{256} + 128 \quad (2.7)$$

La conversion dans le sens inverse est faite avec les équations suivantes :

$$R = \frac{298(Y - 16) + 409(Cr - 128) + 128}{256} \quad (2.8)$$

$$G = \frac{298(Y - 16(Cb - 100) - 208(Cr - 128) + 128}{256} \quad (2.9)$$

$$B = \frac{298(Y - 16) + 516(Cb - 128) + 128}{256} \quad (2.10)$$

Dans le cas d'image RGB, l'algorithme ¹ est peut modifié. Avant de parcourir l'image, on la convertit vers l'espace de luminance YCbCr puis on applique le filtre bilatéral dessus normalement mais uniquement sur la nouvelle composante Y. Ensuite, il n'y a plus qu'à faire basculer l'image de nouveau vers l'espace RGB. On définit I_p^Y la valeur du pixel p pour la composante Y . Ce qui donne l'algorithme suivant :

Entrées : Image RGB I

Sorties : Image filtrée $BF[I]$

// Conversion RGB vers YCbCr

$I = \text{RGBtoYCbCr}(I)$

pour $p \in I$ **faire**

 // Initialisation

$BF[I]_p^Y = 0$

$W_p = 0$

pour $q \in S$ **faire**

$w = G_{\sigma_s}(\| p - q \|)G_{\sigma_r}(| I_p^Y - I_q^Y |)$

$BF[I]_p^Y += wI_q^Y$

$W_p += w$

fin

 // Normalisation

$BF[I]_p^Y = I_p^Y / W_p$

fin

// Conversion YCbCr vers RGB

$BF[I] = \text{YCbCrtoRGB}(BF[I])$

Algorithme 2 : Algorithme du filtre bilatéral pour des images RGB

2.3 Applications

Le filtre bilatéral est utilisé dans divers domaines et pour différents types d'utilisation. Ci-dessous une partie de ces applications sont présentées.

1. Les formules de conversion RGB/YCbCR sont celles utilisées dans la librairie CImg [4]

Débruitage

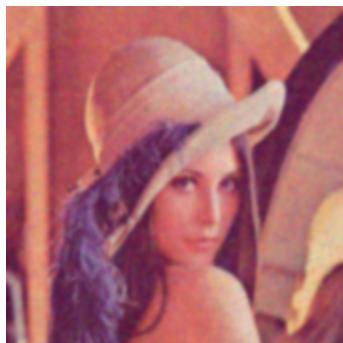
Le premier but du filtre bilatéral lorsqu'il a été créé était l'utilisation comme outil de débruitage pour les images. L'avantage de ce filtre par rapport au filtre gaussien est qu'il va en plus de lisser l'image conserver sa structure comme le montre l'image ci-dessous.



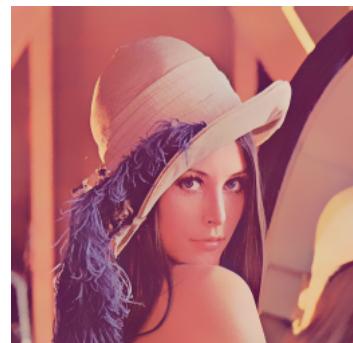
(a) Image originale



(b) Image bruitée (sel et poivre)



(c) Filtre gaussien



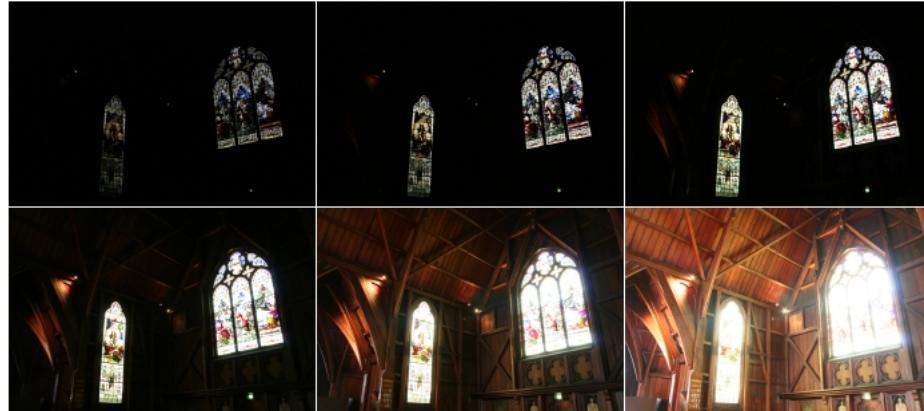
(d) Filtre bilatéral

FIGURE 2.2 – Comparaison de débruitage entre le filtre gaussien et le filtre bilatéral à partir de l'image (b). On remarque que les contours sont conservés avec le filtre bilatéral (d).

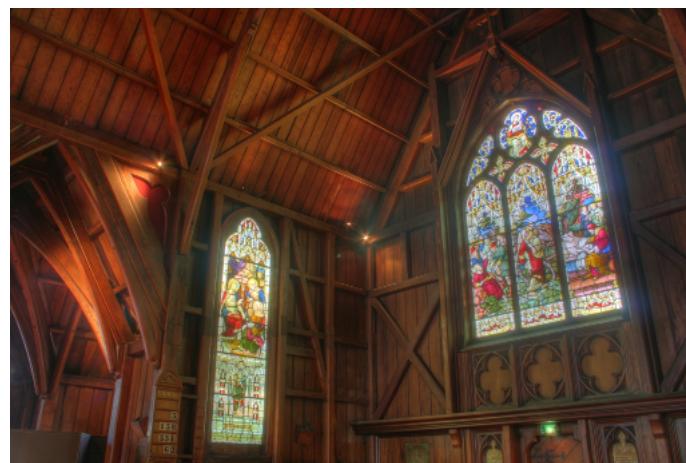
Outils de gestion de contrastes

Le filtre bilatéral peut servir d'outil à la manipulation de détails afin de les améliorer ou les atténuer (voir section 4.2 page 25). Une des applications les plus connue est certainement en photographie avec la technologie HDR². Cette technologie permet de construire une nouvelle image à partir d'une série de photos avec des expositions différentes (fig. 2.3). Cela permet d'obtenir tous les détails de la scène.

2. HDR : High Dynamic Range



(a) Six prises de vue avec des expositions différentes (1/40 1/10 1/2 1 6 25)



(b) Image créée avec la technologie HDR

FIGURE 2.3 – Reconstruction d'une image (b) à partir de photos avec des expositions différentes (a) [5]

Fusion d'images

Une autre technique, découverte par Eisemann et Durand[6], pouvant être utilisé avec le filtre bilatéral est la création d'image à partir d'une photo avec flash et une autre sans (fig. 2.4). Cela permet d'obtenir des photos de bonne qualité lorsque l'on se trouve dans un environnement avec peu de lumière.



(a) Photograph with flash

(b) Photograph without flash

(c) Combination

FIGURE 2.4 – Combinaison d'une image avec flash et d'une image sans flash.

Autres

Le filtre bilatéral peut servir dans bien d'autres domaines d'applications, par exemple pour lisser un objet 3D (fig. 2.5) ou bien pour donner un effet "cartoon" à une vidéo (fig. 2.6).

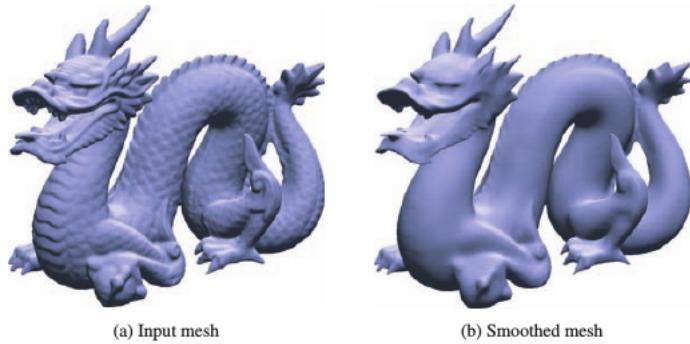


FIGURE 2.5 – Lissage d'un objet 3D à l'aide du filtre bilatéral

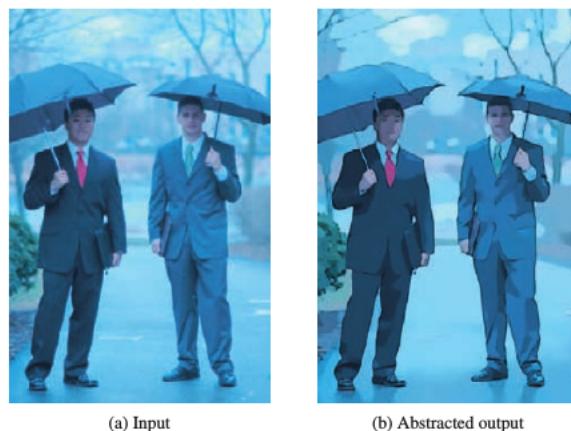


FIGURE 2.6 – Effet "cartoon" d'une vidéo

Utilisation du filtre bilatéral en débruitage

3.1 Mise en place

Afin de pouvoir utiliser le filtre bilatéral par la suite avec l'interface, il a fallu tout d'abord valider son implémentation. Pour cela, on va comparer les résultats obtenus entre l'implémentation du filtre dans Clmg et la nôtre en débruitage (voir 2.3 p. 12).

La validation du filtre bilatéral s'est faite en deux étapes. La première consistait à vérifier que le filtre bilatéral reproduisait correctement le comportement d'un filtre gaussien (voir 2.1.3). Une valeur grande σ_r permet d'approcher une convolution gaussienne car la gaussienne G_{σ_r} va tendre vers un et la formule du filtre bilatéral sera donc semblable à celle de la convolution gaussienne.

La seconde étape permet de vérifier que les résultats obtenus concordent avec ceux de l'implémentation de Clmg. Pour cela deux pourcentages sont calculés : un premier correspondant au maximum de différence entre les valeurs des deux images obtenues (Eq. 3.1) et un deuxième correspondant à la différence entre les deux valeurs (Eq. 3.2). Ces calculs sont réalisés avec les équations suivantes :

$$diffPixel = \max_{i,y} \frac{\sum_c \frac{I_{fb,c} - I_{Clmg,c}}{255} * 100}{c} \quad (3.1)$$

$$diffImage = \frac{\sum_x \sum_y \sum_c \frac{I_{fb,c} - I_{Clmg,c}}{255} * 100}{x * y * c} \quad (3.2)$$

La comparaison se fait avec des images initiales bruitées par un bruit suivant une loi gaussienne. On acceptera un pourcentage de différence entre l'image produite avec Clmg et celle produite avec notre implémentation d'environ 5% maximum et un pourcentage entre les valeurs des pixels d'environ 20% maximum.

3.2 Tests et résultats



(a) Image NdG

(b) Image RGB

FIGURE 3.1 – Images utilisées pour les tests

Les tests sur notre implémentation du filtre bilatéral en débruitage se sont fait à partir des deux images ci-dessus. Une image en niveau de gris et une image RGB ont été testé afin de comparer les différentes implémentations du filtre et vérifier que le nôtre fonctionne correctement.

Afin de comparer le filtre implémenté dans la librairie CImg et le nôtre, nous les avons testé en faisant varier les paramètres avec différentes valeurs

$$\sigma_s = \{4, 8, 16, 32, 64\}$$

$$\sigma_r = \{20, 30, 40, 50, 60\}$$

Toutes les combinaisons possibles ont été essayé afin d'obtenir les résultats présents ci-dessous. On remarque que les écarts se creusent si les valeurs des paramètres augmentent. Les écarts entre les images peuvent être diminué en travaillant avec un masque plus grand mais cela ralentirait l'exécution du filtre. Les résultats présentés ci-après sont une moyenne calculée à partir de dix tests.

Les tests ont été faits dans une machine virtuelle dont voici les caractéristiques :

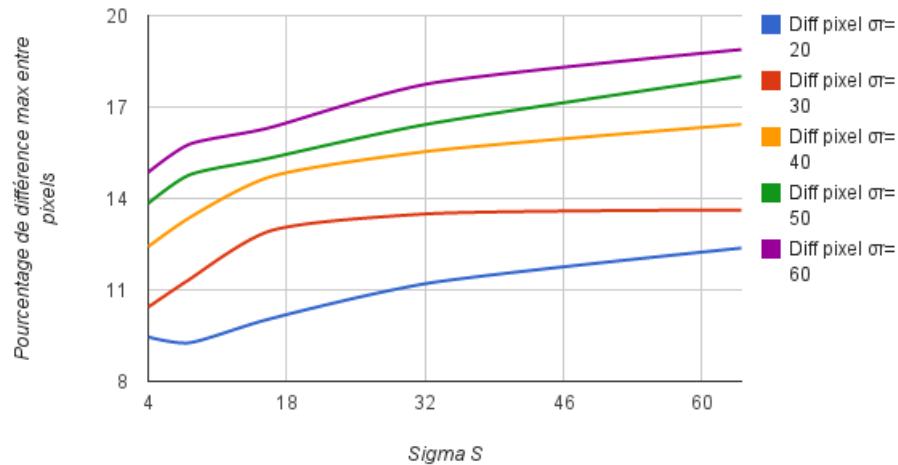
Machine hôte

- ◊ Windows 7 64bits
- ◊ 8Go RAM
- ◊ Intel(R)Xeon(R)CPU W3550 @ 3.06GHz

Machine virtuelle

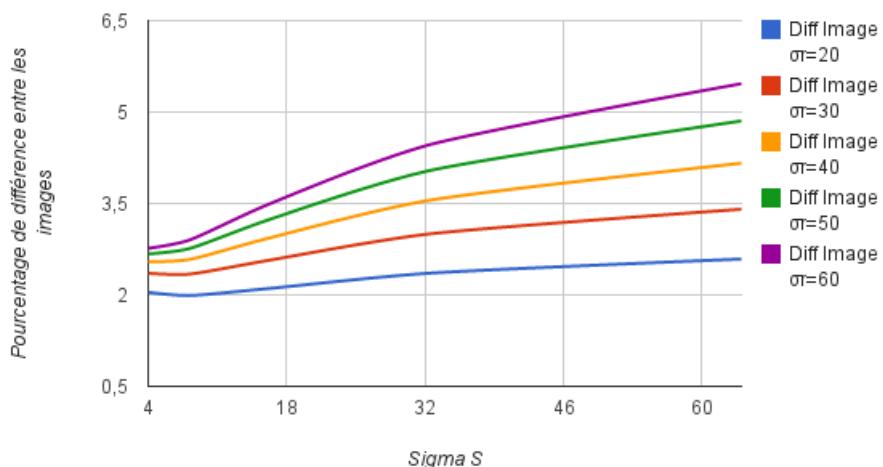
- ◊ Manjaro 64Bits
- ◊ 4Go RAM
- ◊ 4 Core

Comparaison en débruitage (variation de σ_r , image RGB)



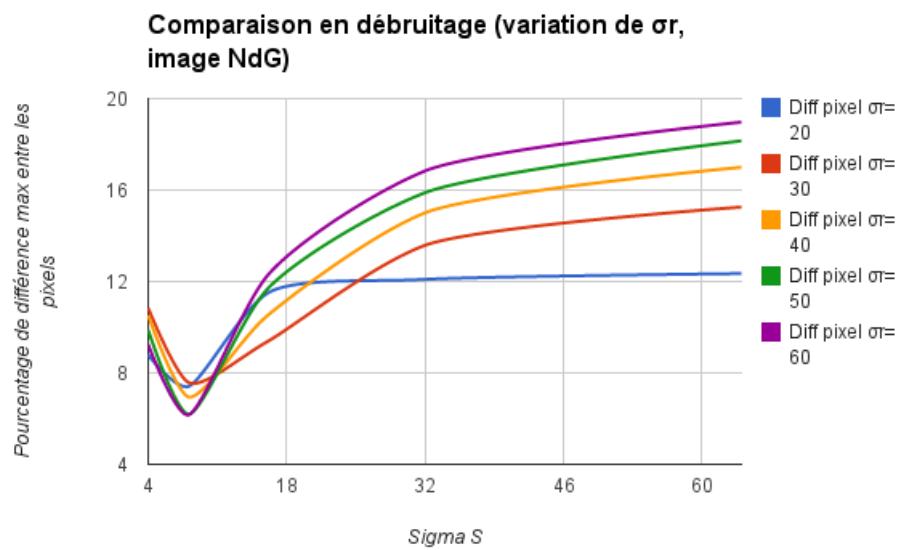
(a) Différence max entre les pixels

Comparaison en débruitage (variation de σ_r , image RGB)

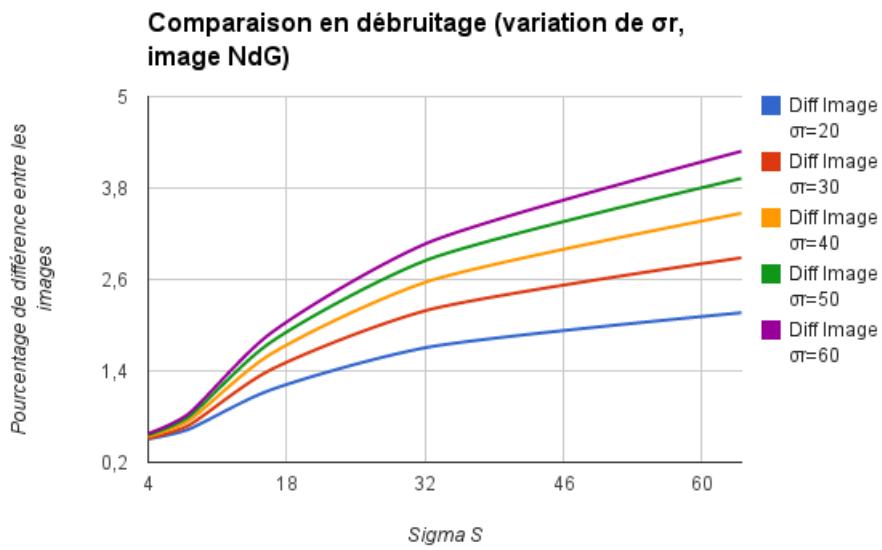


(b) Différence entre les images

FIGURE 3.2 – Comparaison avec une image RGB des deux implémentations en faisant varier σ_r

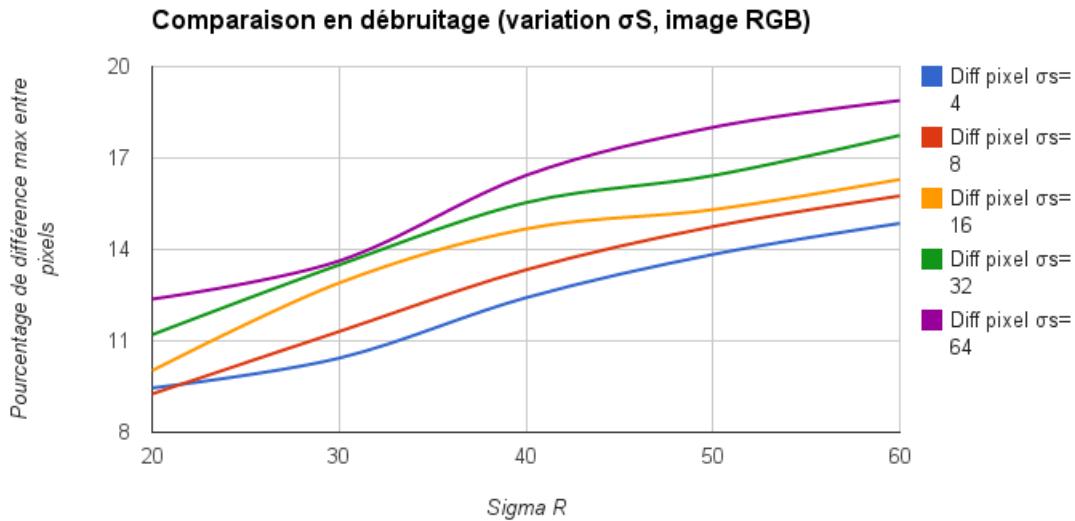


(a) Différence max entre les pixels

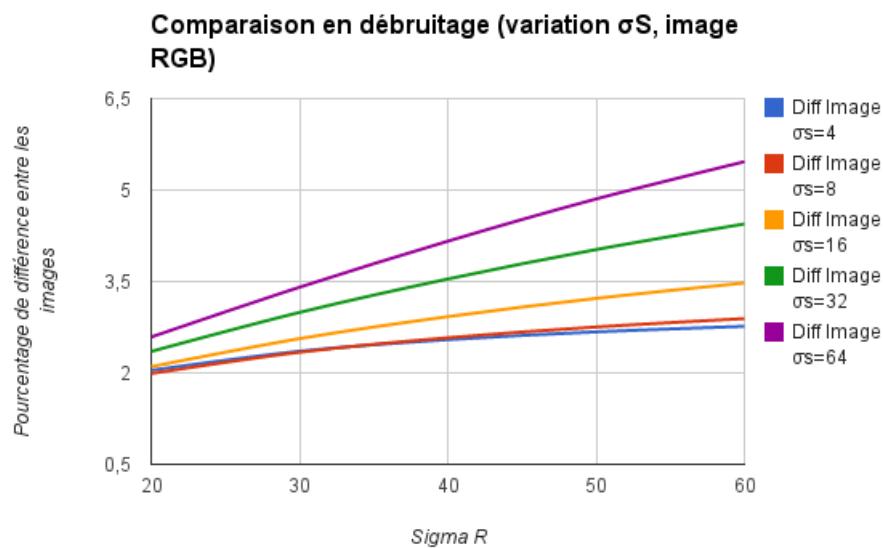


(b) Différence entre les images

FIGURE 3.3 – Comparaison avec une image en niveau de gris(NdG) des deux implémentations en faisant varier σ_r

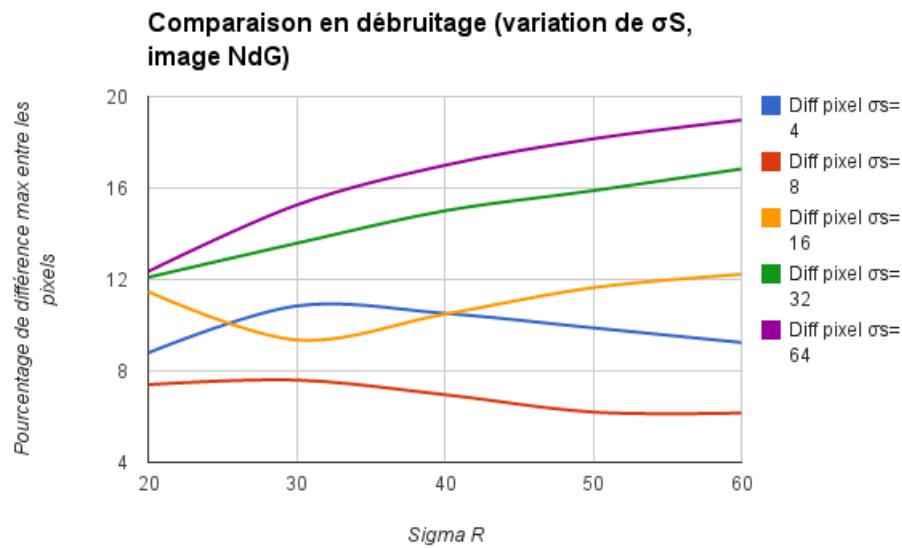


(a) Différence max entre les pixels

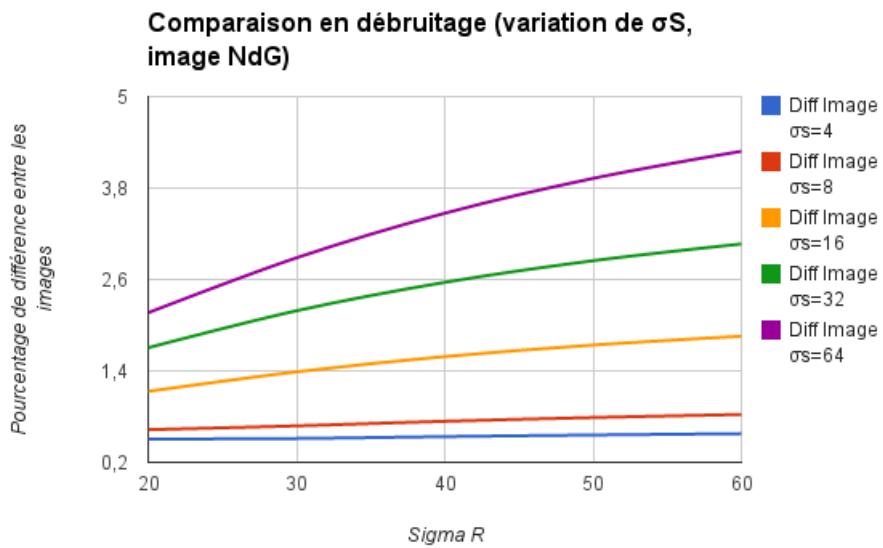


(b) Différence entre les images

FIGURE 3.4 – Comparaison avec une image RGB des deux implémentations en faisant varier σ_s



(a) Différence max entre les pixels



(b) Différence entre les images

FIGURE 3.5 – Comparaison avec une image en niveau de gris(NdG) des deux implémentations en faisant varier σ_s

Pyramide multi-échelle à partir du filtre bilatéral

4.1 Décomposition et recomposition des images

En photographie, il est possible de décomposer une image en :

- ◊ une couche de base qui contient une grande échelle de variations de l'intensité
- ◊ et une couche de détails à petite échelle qui contient tous les petits détails de l'image.

Afin de pouvoir manipuler les détails d'une image, il faut tout d'abord la décomposer. Pour cela, une pyramide multi-échelle est réalisée à partir du filtre bilatéral ce qui va permettre d'obtenir une succession d'images filtrées. Chaque image obtenue correspond à une combinaison de vecteurs (la couche de base et le couche de détails) comme sur l'image ci-dessous.

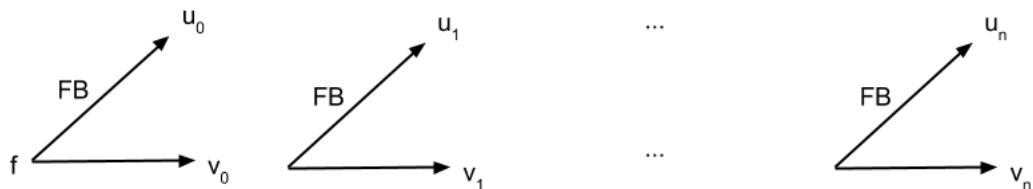


FIGURE 4.1 – Combinaison vectorielle de la décomposition d'une image en itérant le filtre bilatéral

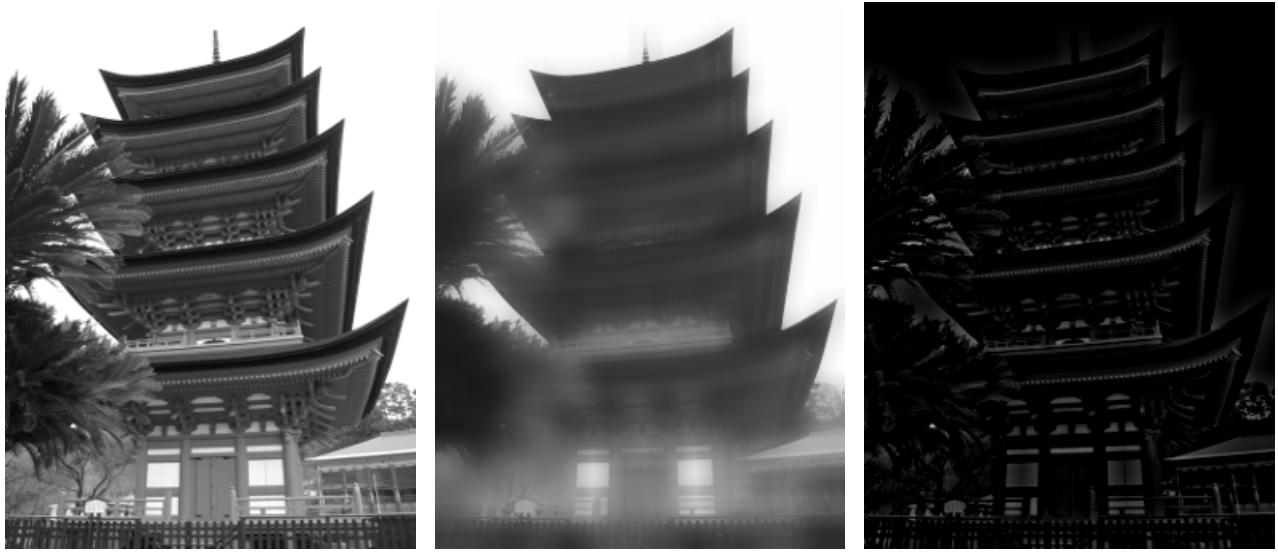
En appliquant le filtre bilatéral sur une image, on peut la décomposer ensuite en une couche de base, c'est à dire l'image résultant du filtre et une couche de détails qui correspond à la différence entre l'image originale et la couche de base (voir fig. 4.2). chem

On notera g l'image originale, u la couche de base et v correspondra à la couche de détails. $BF[\cdot]$ désigne le filtre bilatéral. Soit une décomposition à $(k+1)$ niveau, on aura u^1, \dots, u^k les versions filtrées obtenues de g . La dernière version u^k sera notée b et correspondra à la couche de base. La couche de détail est définie comme suit :

$$v^i = u^{i-1} - u^i, \text{ avec } i = 1 \dots k \text{ et } u^0 = g \quad (4.1)$$

L'image originale g est retrouvé à l'aide de l'équation suivante :

$$g = b + \sum_{i=1}^k v^i \quad (4.2)$$



(a) Image originale

(b) Couche de base

(c) Couche de détails

FIGURE 4.2 – Décomposition entre la couche de base et la couche de détails

Les différentes méthodes et stratégies misent en place ci-dessous se basent sur l'article [7].

4.1.1 Méthodes

La pyramide multi-échelle basée sur le filtre bilatéral peut être construite suivant deux méthodes. Les images sont ensuite reconstruite en utilisant l'équation (4.2).

Méthode n° 1

Cette méthode consiste à itérer le filtre bilatéral sur l'image originale en modifiant uniquement les paramètres σ_s et σ_r . Les séquences u^1, \dots, u^k sont obtenues en résolvant k fois le système suivant :

$$u^{i+1} = BF[g] \quad (4.3)$$

Méthode n° 2

La deuxième méthode que l'on peut mettre en place consiste à appliquer le filtre bilatéral sur la dernière séquence obtenue, ce qui va donner la formule suivante :

$$u^{i+1} = BF[u^i] \quad (4.4)$$

4.1.2 Stratégies

Deux stratégies peuvent être mise en place afin de réaliser la pyramide multi-échelle. La première consiste à augmenter σ_s et σ_r à chaque itération. La seconde stratégie consiste à utiliser la méthode n° 2 (Eq. 4.4) et faire ne faire varier que σ_r , à chaque itération.

4.1.3 Exemples de décompositions et recompositions

Les résultats présentés ci-dessous montrent les différentes séquences obtenues ainsi que leur couche de détails et l'image reconstruite.

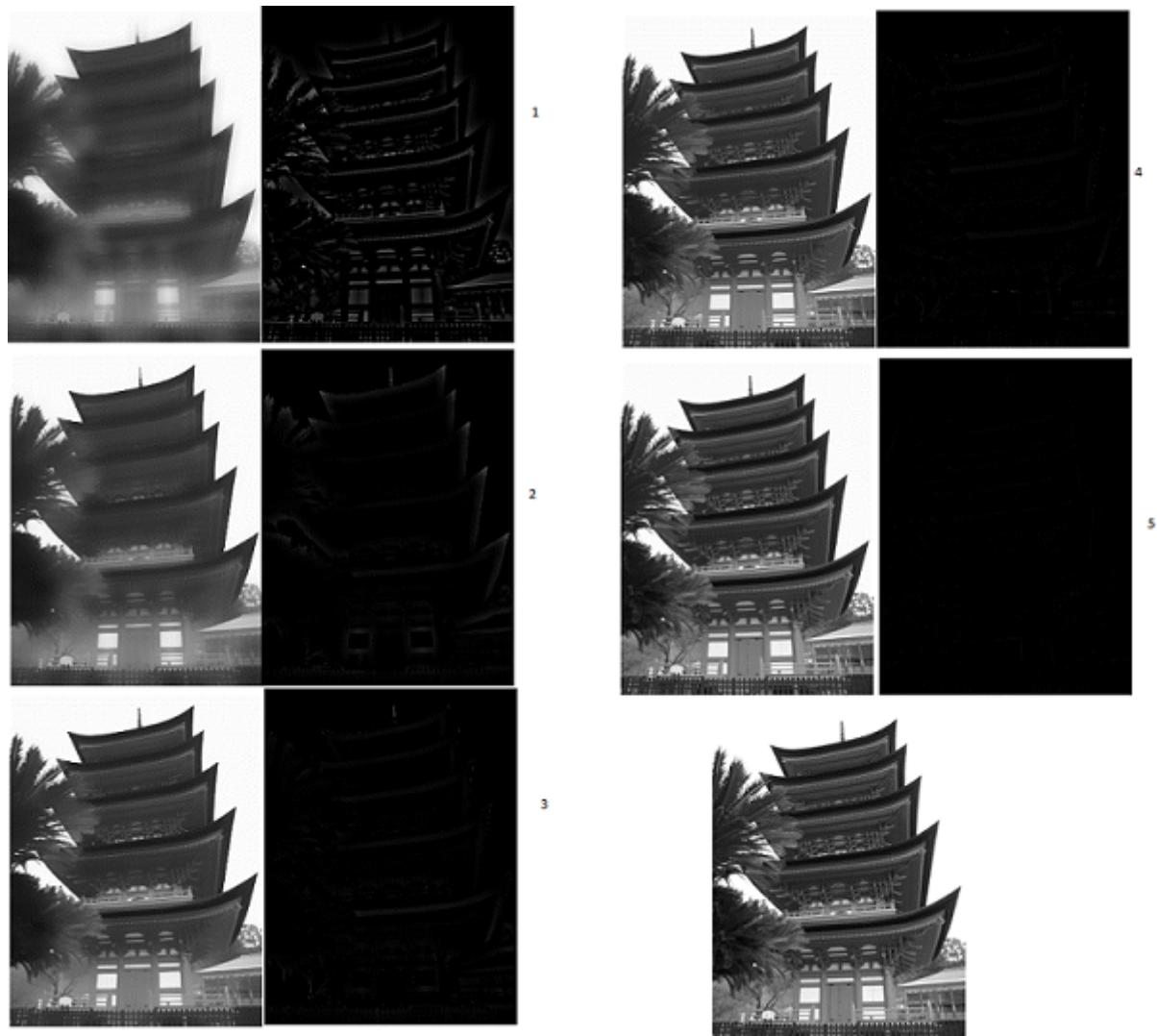


FIGURE 4.3 – Décomposition pyramidale (méthode 1 et stratégie 2 - σ_r divisé par 2), paramètre de départ $\sigma_s=36$ et $\sigma_r=100$

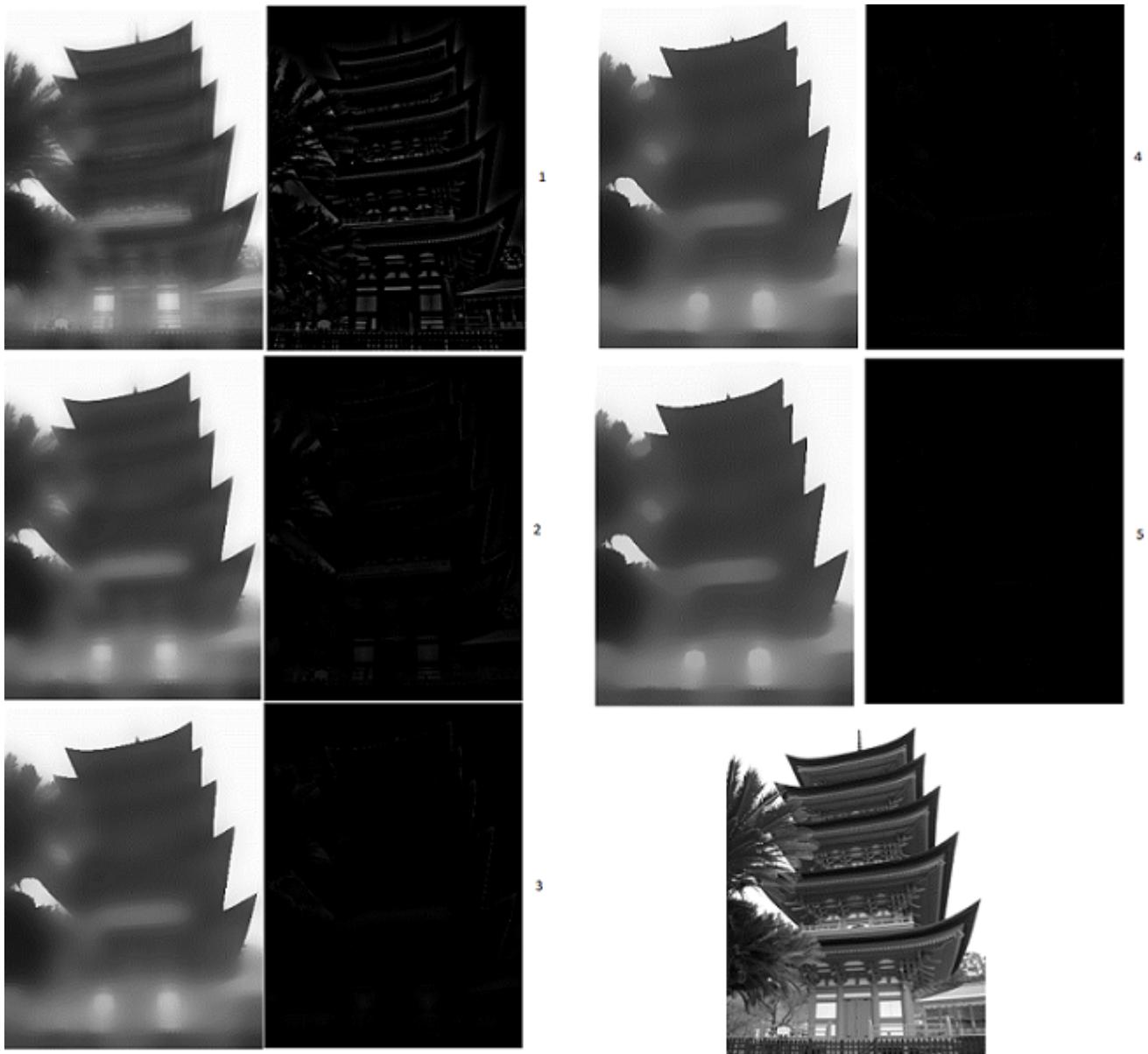


FIGURE 4.4 – Décomposition pyramidale (méthode 2 et stratégie 2 - σ_r divisé par 2), paramètre de départ $\sigma_s=36$ et $\sigma_r=100$

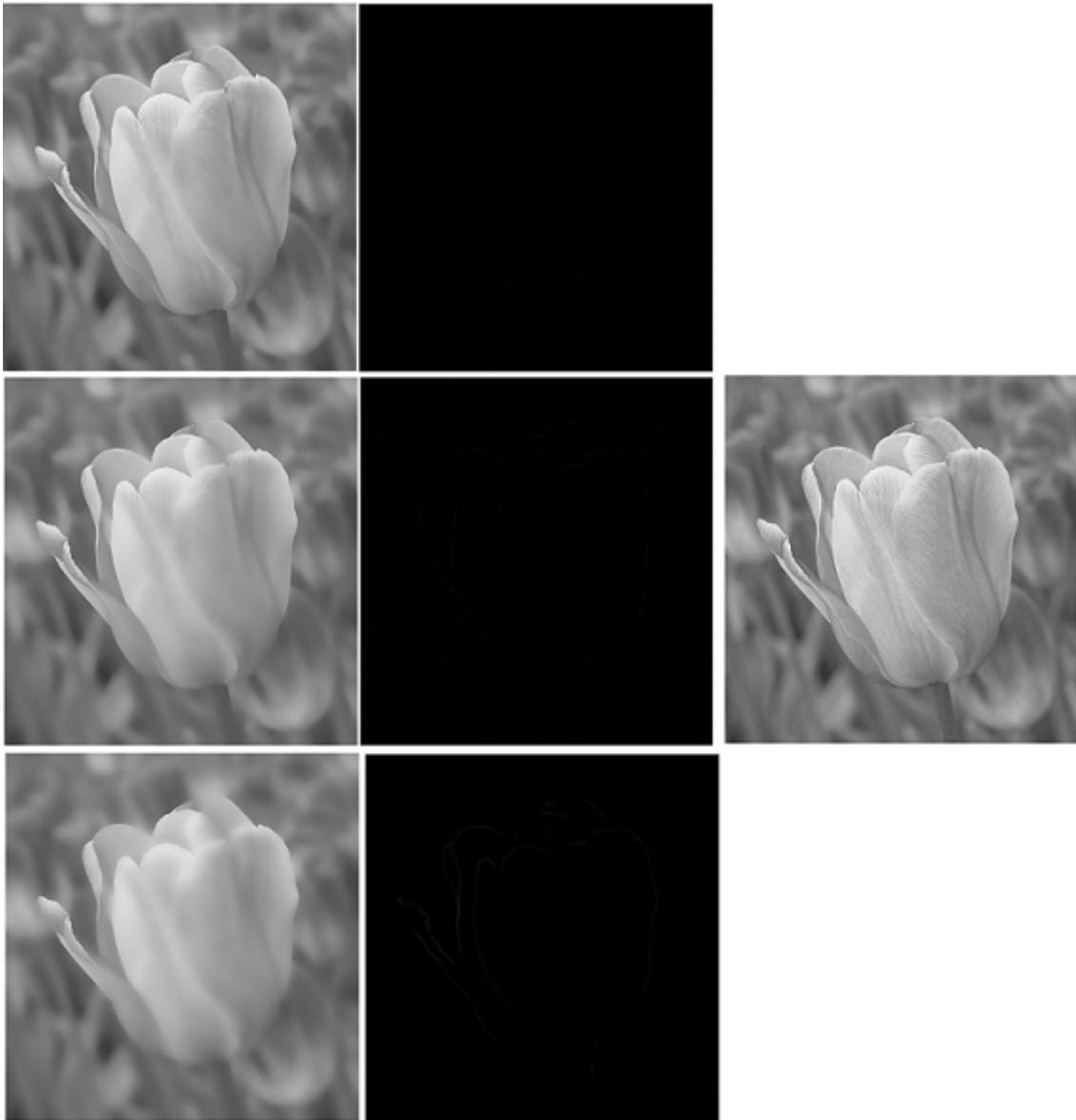


FIGURE 4.5 – Décomposition pyramidale (méthode 1 et stratégie 1 - facteur 2), paramètre de départ $\sigma_s=4$ et $\sigma_r=10$

4.2 Manipulation des détails

4.2.1 Mise en place

La décomposition pyramidale multi-échelle permet d'obtenir un ensemble d'images composé de couches de base (u) et de détails (v). L'image sera reformé avec cet ensemble en utilisant la formule suivante :

$$g = b + \sum_{i=1}^k v^i \quad (4.5)$$

Afin de manipuler le niveau de détails de l'image, l'équation 4.5 va être légèrement modifiée. Les différentes couches de détails d et la couche de base b seront multiplié par un coefficient β et α ce qui donnera

l'équation 4.6.

$$g = \alpha * b + \sum_{i=1}^k \beta * (i+1) * v^i \quad (4.6)$$

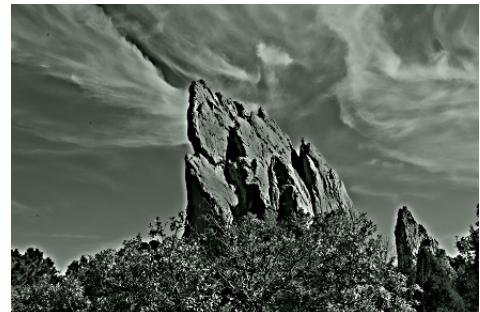
Les facteurs α et β permettent d'indiquer l'importance des couches et donc de déterminer le niveau de détails à manipuler. Une valeur inférieure d' α par rapport à β est utilisée pour rehausser et le contraire pour atténuer.

4.2.2 Exemples de manipulation des détails : rehaussement

De nombreux tests ont été réalisés afin de trouver des valeurs de base pour α et β afin de rehausser le niveau de détails de l'image. Ces valeurs de base seront celles utilisées par défaut dans l'application (voir sec. 5).



(a) Méthode 1



(b) Méthode 2



(c) Image originale

FIGURE 4.6 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=1$



(a) Méthode 1



(b) Méthode 2



(c) Image originale

FIGURE 4.7 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=3$ 

(a) Méthode 1



(b) Méthode 2



(c) Image originale

FIGURE 4.9 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=1$

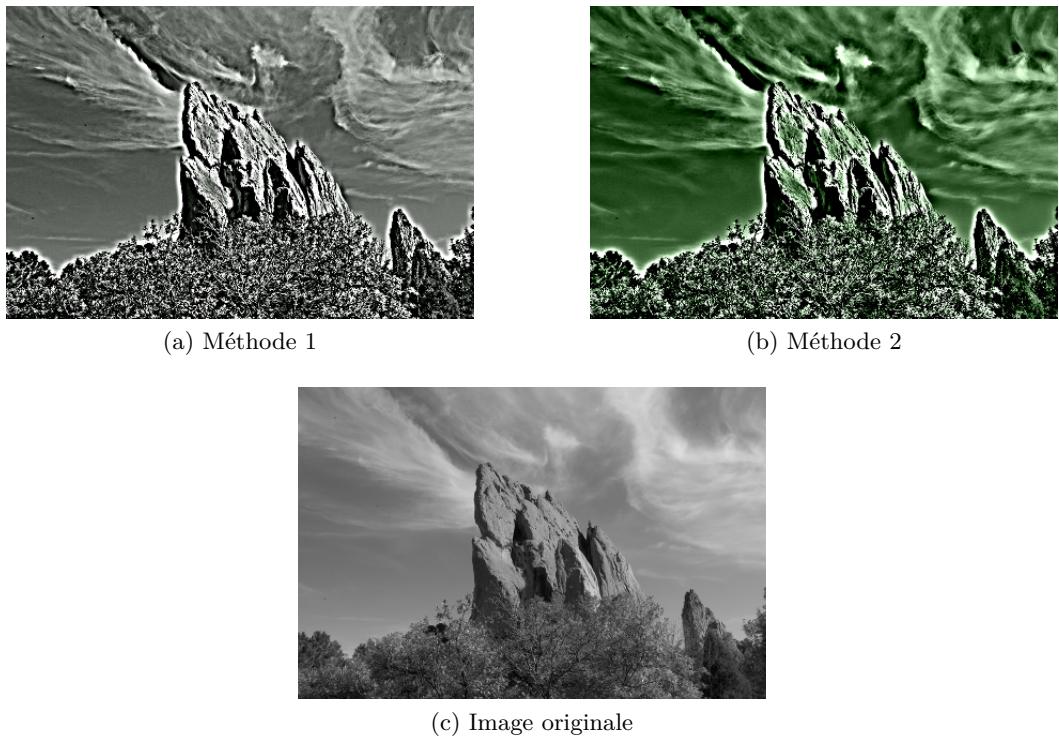


FIGURE 4.8 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$

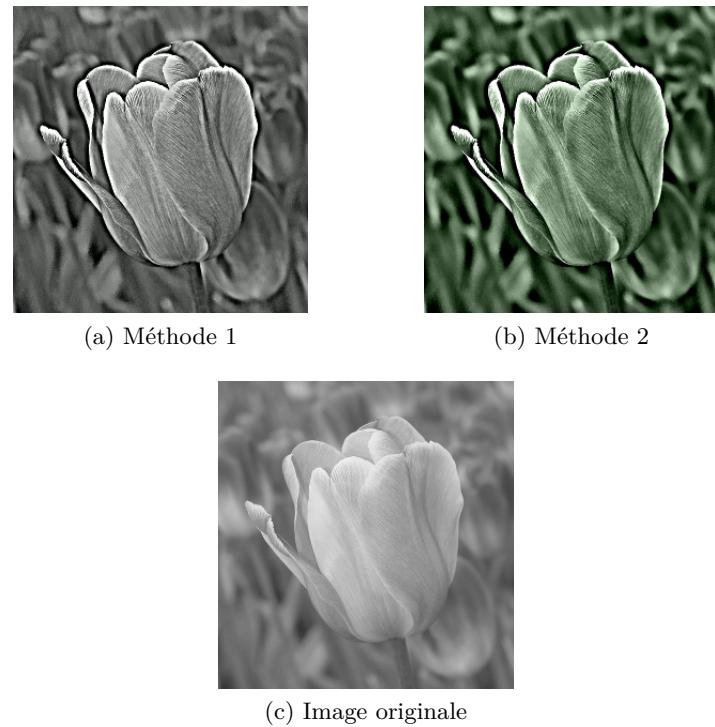
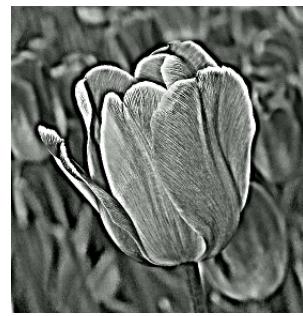


FIGURE 4.10 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=3$



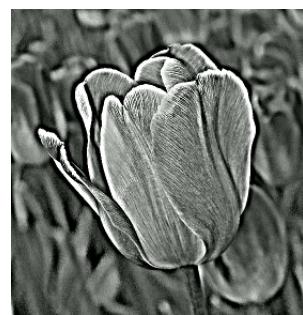
(a) Méthode 1



(b) Méthode 2



(c) Image originale

FIGURE 4.11 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$ 

(a) Méthode 1



(b) Méthode 2

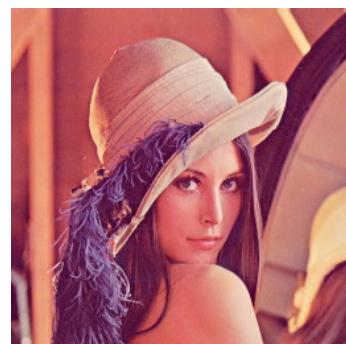


(c) Image originale

FIGURE 4.12 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$



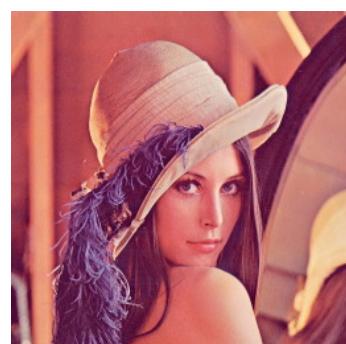
(a) Méthode 1 et 2



(b) Image originale

FIGURE 4.13 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=1$


(a) Méthode 1 et 2



(b) Image originale

FIGURE 4.14 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=3$



(a) Méthode 1 et 2



(b) Image originale

FIGURE 4.15 – Réhaussement des détails avec $\alpha=0.8$ et $\beta=5$

4.2.3 Exemples de manipulation des détails : atténuation

Les recherches précédemment effectuées ont été réitérées mais cette fois-ci pour avoir un α et un β permettant d'atténuer au mieux les particularités de l'image.



(a) Méthode 1



(b) Méthode 2



(c) Image originale

FIGURE 4.17 – Atténuation des détails avec $\alpha=1$ et $\beta=0.5$


(a) Méthode 1



(b) Méthode 2



(c) Image originale

FIGURE 4.16 – Atténuation des détails avec $\alpha=1$ et $\beta=0.8$



(a) Méthode 1



(b) Méthode 2



(c) image originale

FIGURE 4.18 – Atténuation des détails avec $\alpha=1$ et $\beta=0.8$ 

(a) Méthode 1



(b) Méthode 2



(c) Image originale

FIGURE 4.19 – Atténuation des détails avec $\alpha=1$ et $\beta=0.5$

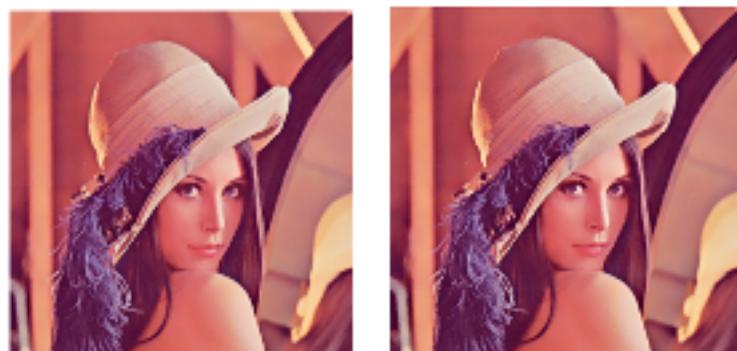


(a) Méthode 1 et 2

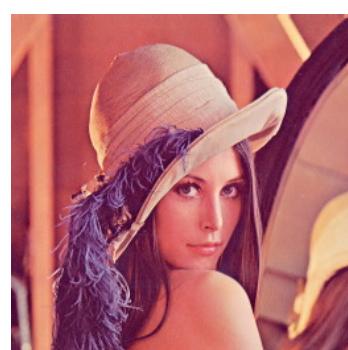


(b) Image originale

FIGURE 4.20 – Atténuation des détails avec $\alpha=1$ et $\beta=0.8$



(a) Méthode 1 et 2



(b) Image originale

FIGURE 4.21 – Atténuation des détails avec $\alpha=1$ et $\beta=0.5$

Interface graphique

5.1 Fonctionnalités et structure du système

5.1.1 Cas d'utilisation

Afin que l'utilisateur puisse utiliser modifier ses images en manipulant le niveau de détails de celle-ci, une interface a été réalisée. À partir de celle-ci, il peut effectuer différentes actions (fig. 5.1) :

- ◊ charger une image
- ◊ choisir quel niveau de détails il souhaite manipuler
- ◊ rentrer des informations complémentaires :
 - ▷ le nombre d'itérations
 - ▷ la valeur de σ_s
 - ▷ la valeur de σ_r
 - ▷ la valeur de α
 - ▷ la valeur de β
- ◊ lancer le traitement
- ◊ sauvegarder la nouvelle image

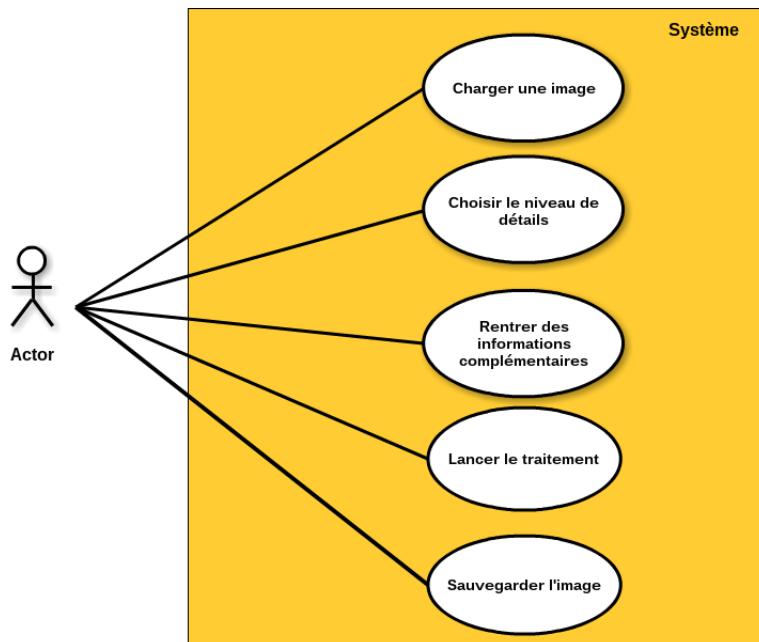


FIGURE 5.1 – Diagramme des cas d'utilisations

5.1.2 Structure générale

L'IHM¹ a été développé suivant un modèle MVC (Modèle-Vue-Contrôleur) à l'aide du framework Qt en C++. La vue est gérée par la classe MainWindow (voir diagramme de classe A.2). Cette classe hérite de

1. IHM : Interface Homme-Machine

QMainWindow. Cette classe permet de gérer la fenêtre principale d'une application en fournissant différents outils tels qu'une barre de menu, une barre d'outils, une barre d'états, un dock central (pour les différents outils pouvant être nécessaire au widget central) et enfin un widget central (fig. 5.2).

Dans notre application seulement le widget central sera utilisé. Mais à l'avenir une barre de menu ou d'outils pourraient être ajouté afin de fournir des utilisations supplémentaires. La barre d'état pourrait elle aussi être utilisé afin de notifier l'utilisateur de l'avancement du traitement ou de l'action en cours.

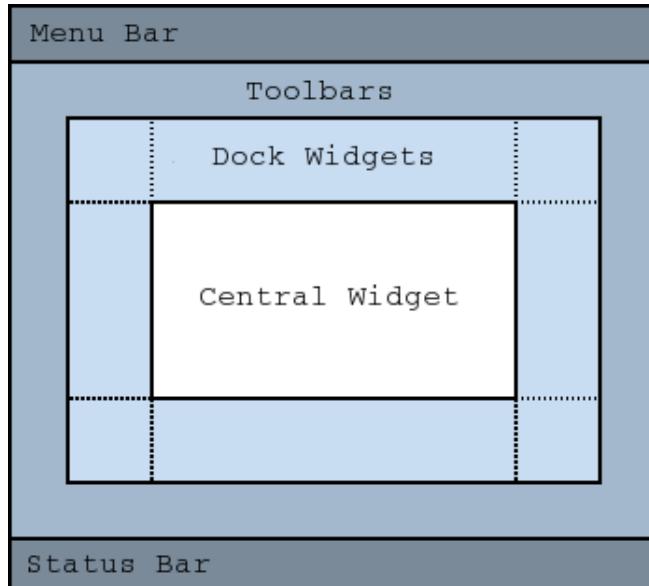


FIGURE 5.2 – Compostion d'une fenêtre QMainWindow

Le widget principal a été crée dans une classe à part héritant de QWidget appelée WidgetImage. Il est composé de quatres QGroupBox et d'un bouton (QPushButton).



FIGURE 5.3 – Composition du widget central : WidgetImage

La classe `WidgetImage` gère seulement l'affichage du composant ainsi que des images qui s'afficheront dans des `QLabel` inclus dans des `QGroupBox`. Ces `QGroupBox` apparaissent lorsque l'utilisateur charge une image ou à la fin du traitement de celle-ci.

Lorsque l'utilisateur lance le traitement à l'aide du bouton "Lancer" (fig. 5.3), celui-ci va être contrôlé par la classe `Controller` qui sera lancé dans un thread (`ControllerThread`) à part afin que le traitement ne bloque pas l'application.

La classe `FilterBilateral`(voir diagramme de classe fig. A.1) modélise la partie "Modèle" du MVC. Elle est uniquement appelée par le contrôleur. L'implémentation du filtre bilatéral est celle décrite dans la section 2.2.

5.2 Guide d'utilisation

Au lancement de l'application, l'utilisateur ne verra que la partie gauche de l'interface (fig. 5.4). Il va ensuite pouvoir choisir l'image qu'il souhaite modifier. Pour cela, le bouton "Navigation" va ouvrir une fenêtre de navigation dans les dossiers et affichera l'image à droite s'il y a une de sélectionnée. L'utilisateur ne pourra choisir que des images de types png ou jpeg.

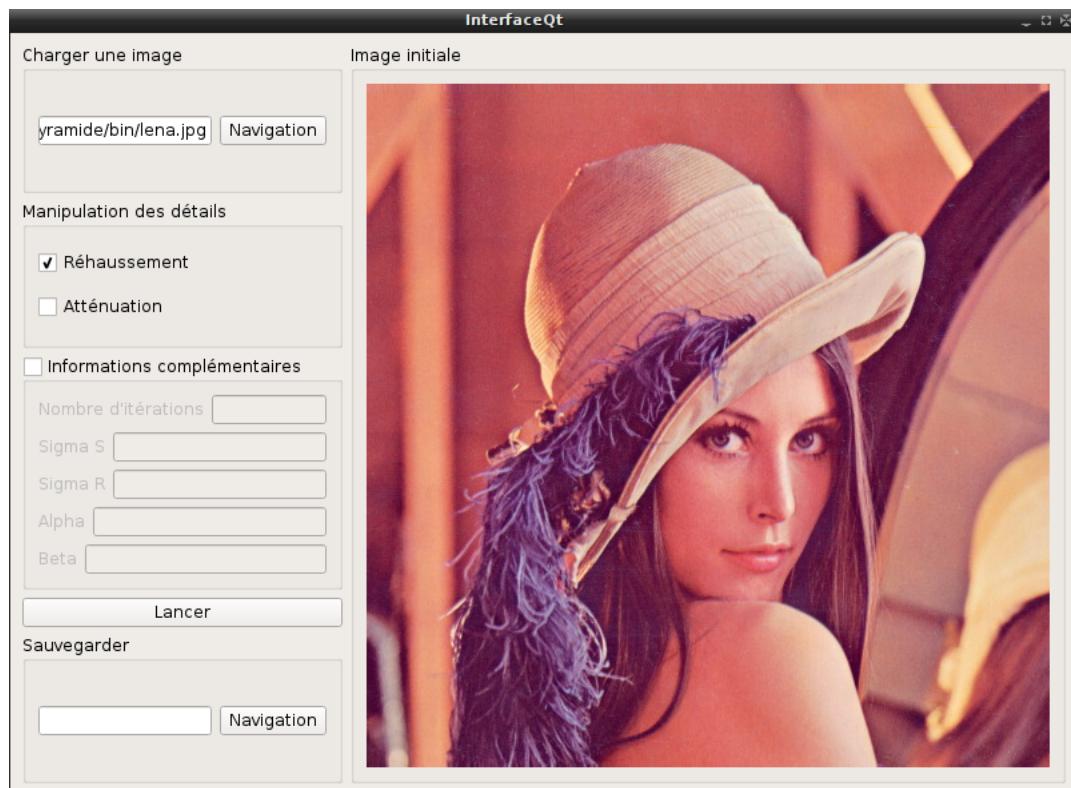


FIGURE 5.4 – Visualisation de l'interface après chargement de l'image

Avant de lancer le traitement, il faut choisir le niveau de détails à manipuler : soit rehausser soit atténuer. L'utilisateur a ensuite la possibilité de choisir ses propres paramètres. S'il préfère ne pas utiliser des paramètres personnalisés, ceux par défaut seront pris en compte. Dans le cas où la `QGroupBox` est coché, il n'est pas nécessaire de renseigner des informations pour tous les champs. Si un champ n'est pas rempli, sa valeur par défaut sera utilisée.

Une fois que l'utilisateur a fini de choisir comme le traitement doit être exécuté, il peut cliquer sur le bouton "Lancer". À la fin de la manipulation de l'image, la nouvelle image apparaît (fig. 5.5). Lorsque le traitement est fini, il est possible de sauvegarder l'image.

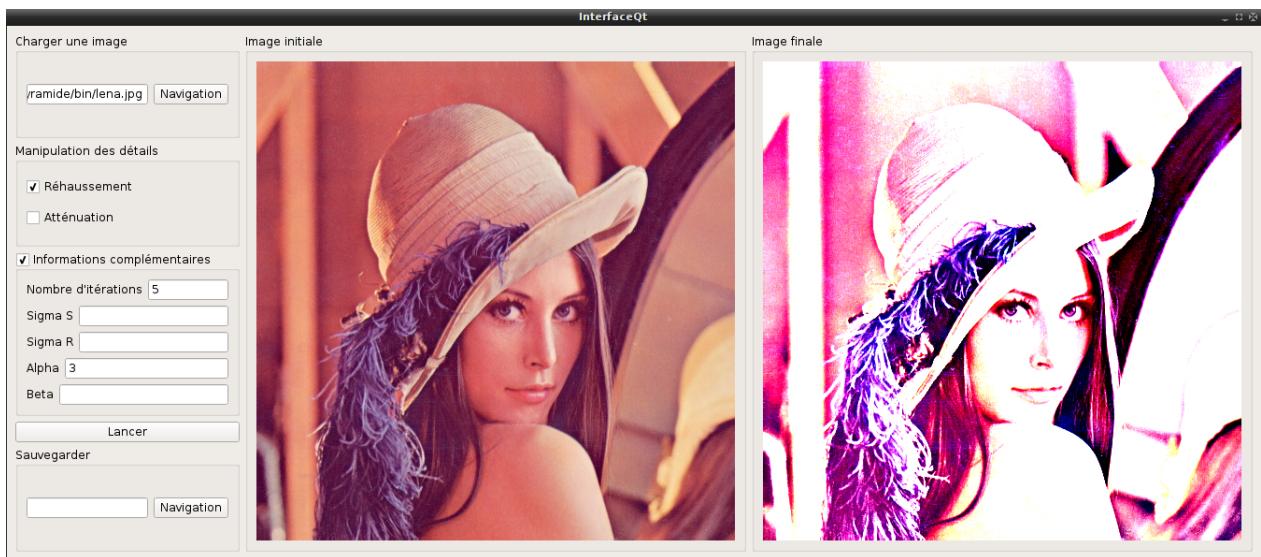


FIGURE 5.5 – Visualisation de l'interface après traitement de l'image

Accélération

Par la suite nous avons cherché à optimiser notre implémentation du filtre bilatéral mais nous nous sommes rendus compte que des versions optimisées existaient déjà. La version du filtre bilatéral présente dans la librairie CImg ainsi que celle de Paris et Durand [8] font partie de ces versions.

6.1 Utilisation du filtre bilatéral comme une convolution

Cette partie est basée sur l'article de Paris et Durand [9].

6.1.1 Rappels

Le filtre utilisé précédemment (voir 2.1.4) est :

- ◊ non-linéaire : chaque pixel est remplacé par un poids de ses voisins
- ◊ et non-itératif : le résultat est obtenu en un seul passage.

Il permet de remplacer chaque pixel par un poids moyen de ses voisins. Ce poids va décroître avec à la fois la distance spatiale dans l'image (le domaine spatial S) et la distance en intensité des pixels (le domaine d'intensité R). Le filtre bilatéral est défini par (G_σ est une fonction gaussienne) :

$$I_p^{bf} = \frac{1}{W_p^{bf}} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(| I_p - I_q |) I_q \quad (6.1)$$

$$W_p^{bf} = \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(| I_p - I_q |) \quad (6.2)$$

σ_s définit la taille du voisinage considéré et σ_r contrôle l'importance des pixels environnant en fonction de leur intensité. W_p normalise la somme des poids.

6.1.2 Approche

Paris et Durand ont cherché à décomposer le filtre bilatéral en une convolution suivie par deux non-linéarités. Pour transformer le filtre en une convolution, une intensité homogène est définie ce qui permet d'obtenir que le terme de la normalisation W_p^{bf} soit un composant homogène après la convolution. Dans l'équation (6.1), la non-linéarité vient de la division par W_p^{bf} ainsi que de la dépendance entre l'intensité des pixels par $G_{\sigma_r}(| I_p - I_q |)$. Chaque point va être étudié séparément et isolé pendant le calcul.

6.1.3 Intensité homogène

Une première solution pour palier à la division est de multiplier l'équation (6.1) des deux côtés par W_p^{bf} . Les équations (6.1) et (6.2) sont alors similaires. Ce qui donne l'équation à deux dimensions suivante :

$$\begin{pmatrix} W_p^{bf} I_p^{bf} \\ W_p^{bf} \end{pmatrix} = \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(| I_p - I_q |) \begin{pmatrix} I_q \\ 1 \end{pmatrix} \quad (6.3)$$

Afin de maintenir que le filtre est moyenne pondérée, une fonction W est introduite. Celle-ci vaut 1 en tout point, ce qui donne :

$$\begin{pmatrix} W_p^{bf} I_p^{bf} \\ W_p^{bf} \end{pmatrix} = \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(|I_p - I_q|) \begin{pmatrix} W_q I_q \\ W_q \end{pmatrix} \quad (6.4)$$

En assignant à chaque pixel q un couple $(W_q I_q, W_q)$, les pixels filtrés sont exprimés comme une combinaison linéaire de leurs voisins. La division n'a pas encore été supprimée car pour accéder à la valeur individuelle d'un pixel, le premier coordonnée (WI) doit encore être divisé par le second (W). Le couple (WI, W) est appelé l'intensité homogène.

Bien-que l'équation (6.4) soit une combinaison linéaire, cela ne définit pas un filtre linéaire puisque le poids dépend de la valeur actuelle des pixels.

6.1.4 Convolution

Si l'on ignore le terme $G_{\sigma_r}(|I_p - I_q|)$, l'équation (6.4) est une convolution gaussienne :

$(W_q^{bf} I_q^{bf}, W_q^{bf}) = G_{\sigma_s} \otimes (WI, W)$. Mais l'intensité des poids dépend de $I_p - I_q$ et il n'y a pas de somme de I . Pour surmonter ce point, une dimension additionnelle ζ est introduite. Avec le symbole de Kronecker $\delta(\zeta)$ (1 si $\zeta = 0$, 0 sinon) et R l'intervalle sur lequel l'intensité est définie, l'équation (6.4) est réécrite en utilisant $[\delta(\zeta - I_q) = 1] \leftrightarrow [\zeta = I_q]$:

$$\begin{pmatrix} W_p^{bf} I_p^{bf} \\ W_p^{bf} \end{pmatrix} = \sum_{q \in S} \sum_{\zeta \in R} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(|I_p - \zeta|) \delta(\zeta - I_q) \begin{pmatrix} W_q I_q \\ W_q \end{pmatrix} \quad (6.5)$$

L'équation (6.5) est une somme sur le produit dans l'espace SxR . Les fonctions définies dans SxR sont en minuscules. Le produit $G_{\sigma_s} G_{\sigma_r}$ défini une gaussienne g_{σ_s, σ_r} dans SxR :

$$g_{\sigma_s, \sigma_r} : (x \in S, \zeta \in R) \mapsto G_{\sigma_s}(\|x\|) G_{\sigma_r}(|\zeta|) \quad (6.6)$$

Deux fonctions i et w sont créées :

$$i : (x \in S, \zeta \in R) \mapsto I_x \quad (6.7)$$

$$w : (x \in S, \zeta \in R) \mapsto \delta(\zeta - I_x) W_x \quad (6.8)$$

Ce qui donne les relations suivantes :

$$I_x = i(x, I_x) \quad (6.9)$$

$$W_x = w(x, I_x) \quad (6.10)$$

$$\forall \zeta \neq I_x, w(x, \zeta) = 0 \quad (6.11)$$

L'équation (6.5) est réécrite de la manière suivante :

$$\begin{pmatrix} W_p^{bf} I_p^{bf} \\ W_p^{bf} \end{pmatrix} = \sum_{(q, \zeta) \in SxR} g_{\sigma_s, \sigma_r}(p - q, I_p - \zeta) \begin{pmatrix} w(q, \zeta) i(q, \zeta) \\ w(q, \zeta) \end{pmatrix} \quad (6.12)$$

La formule ci-dessous correspond à la valeur au point (p, I_p) de la convolution entre g_{σ_s, σ_r} et la fonction à deux dimensions (wi, w) :

$$\begin{pmatrix} W_p^{bf} I_p^{bf} \\ W_p^{bf} \end{pmatrix} = \left[g_{\sigma_s, \sigma_r} \otimes \begin{pmatrix} wi \\ w \end{pmatrix} \right] (p, I_p) \quad (6.13)$$

Les fonctions i^{bf} et w^{bf} sont introduites :

$$(w^{bf} i^{bf}, w^{bf}) = g_{\sigma_s, \sigma_r} \otimes (wi, w) \quad (6.14)$$

Le filtre est exprimé comme une convolution suivie d'opérations non-linéaires :

$$(w^{bf} i^{bf}, w^{bf}) = g_{\sigma_s, \sigma_r} \otimes (wi, w) \quad (\text{linéaire}) \quad (6.15)$$

$$I_p^{bf} = \frac{w^{bf}(p, I_p) i^{bf}(p, I_p)}{w^{bf}(p, I_p)} \quad (\text{non-linéaire}) \quad (6.16)$$

La partie non-linéaire est en réalité composée de deux opérations. Les fonctions $w^{bf} i^{bf}$ et w^{bf} sont calculées au point (p, I_p) . Cette opération est nommée *slicing*. La deuxième opération non-linéaire est la division. Dans ce cas, le *slicing* et la division commute, c'est-à-dire que le résultat est indépendant de l'ordre car g_{σ_s, σ_r} est positif et w vaut soit 0 soit 1, ce qui assure que w^{bf} est positif.

6.2 Implémentation de Paris et Durand

Le domaine spatial S est un plan classique dans l'image xy et le domaine d'intensité R est un simple axe ζ . La fonction w peut être interprétée comme le graph dans l'espace $xy\zeta$ de $\zeta = I(x, y)$, c'est-à-dire que w est null partout sauf sur les points $(x, y, I(x, y))$ où il vaut 1. La fonction wi est similaire à w . À la place d'utiliser des valeurs binaires (0 ou 1), 0 ou $I(x, y)$ sont utilisés.

En premier, on applique une convolution gaussienne définie sur $xy\zeta$ à wi et w . Ce qui permet d'obtenir les fonctions $w^{bf} i^{bf}$ et w^{bf} . Pour chaque point de l'espace $xy\zeta$, on calcule $i^{bf}(x, y, \zeta)$ en divisant $w^{bf}(x, y, \zeta) i^{bf}(x, y, \zeta)$ par $w^{bf}(x, y, \zeta)$. La dernière étape consiste à obtenir la valeur du pixel (x, y) de l'image filtrée I^{bf} qui correspond directement à la valeur de i^{bf} en $(x, y, I(x, y))$.

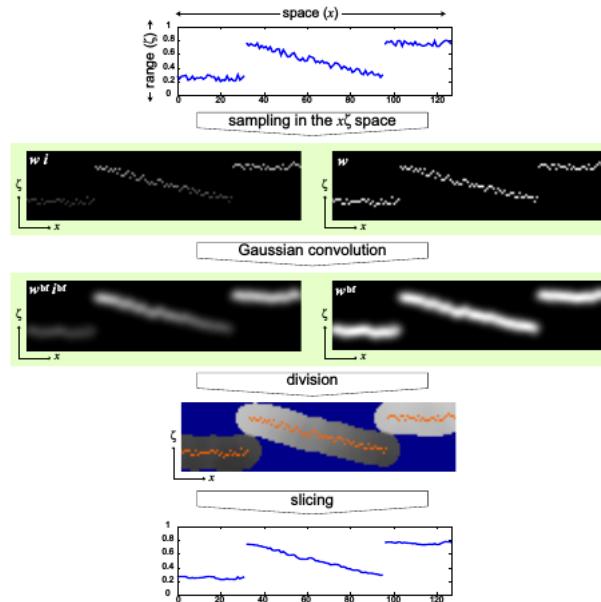


FIGURE 6.1 – Les données de bases (ligne du hau) sont représentées par une fonction à deux dimensions (wi, w) (seconde ligne). À cette fonction est appliquée une convolution gaussienne pour obtenir ($w^{bf} i^{bf}, w^{bf}$) (troisième ligne). La première composante est divisé par la seconde (quatrième ligne). Le résultat final est obtenu en extrayant par échantillonnage la valeur à la position des données de bases.

Sur leur site [8], Paris et Durand propose différentes versions du filtre bilatéral optimisé : pour les images RGB ou en niveau de gris, une version utilisant les transformées rapides de fourrier.

6.3 Implémentation de CImg

L'implémentation du filtre bilatéral dans la librairie CImg utilise les techniques d'optimisation proposées dans l'article de Paris et Durand [9]. La librairie permet aussi d'appliquer le filtre sur des objets 3D. Cette version est basée sur l'implémentation du filtre pour MATLAB réalisé par Jiawen Chen [10].

6.4 Comparaison entre les différents filtres

Les résultats présentés ci-après sont une moyenne calculée à partir de 5 tests.

Les tests ont été faits dans une machine virtuelle dont voici les caractéristiques :

Machine hôte

- ◊ Windows 7 64bits
- ◊ 8Go RAM
- ◊ Intel(R)Xeon(R)CPU W3550 @ 3.06GHz

Machine virtuelle

- ◊ Manjaro 64Bits
- ◊ 4Go RAM
- ◊ 4 Core

Afin de pouvoir intégrer à notre interface (voir chap. 5) la version de l'implémentation du filtre bilatéral la plus rapide, elles ont été comparées avec différentes images RGB de tailles diverses et avec différentes valeurs de σ_s et σ_r .

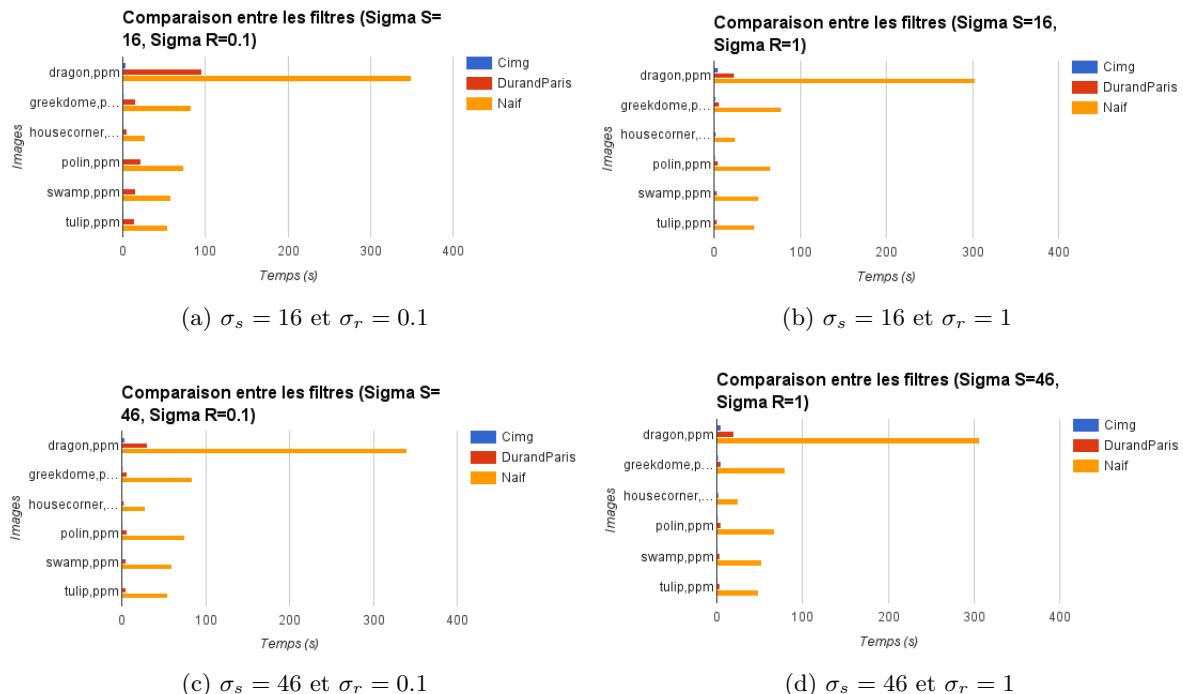


FIGURE 6.2 – Comparaison sur un jeu d'images des différents filtres

On remarque facilement que la courbe censée représenter le temps d'exécution du filtre de CImg (en bleu) n'est quasiment jamais visible grâce à sa rapidité par rapport aux autres. L'implémentation de Paris et

Durand est légèrement moins rapide que celle de Cimg sur des images de grandes tailles (ici dragon.ppm, la plus petite étant housecorner). La rapidité dépend principalement de la taille de l'image. Il en va de même pour la version du filtre naïve.

C'est donc le filtre bilatéral implémenté dans la librairie Clmg qui sera ajouté à l'interface graphique afin de rendre le rendu de l'image plus rapide.

Conclusion

À la fin du temps imparti pour réaliser ce projet, on peut dire que les différents objectifs fixés au début ont été atteints. En effet l'interface permettant de manipuler à différents niveaux les détails d'une image a été réalisé au début avec une implémentation naïve du filtre bilatéral puis par la suite celle-ci a été remplacée par une version optimisée (celle de la librairie CImg).

Cela a été un réel plaisir de pouvoir travailler pendant au temps de mois sur un sujet de traitement d'image. Cela m'a permis d'améliorer mes compétences dans ce domaine et de mieux comprendre certains cours que j'ai pu suivre cette année.

On peut déjà commencer à envisager diverses idées d'améliorations tels que l'ajout de manipulation d'objet 3D soit en utilisant l'outil présent dans la librairie CImg soit en créant le nôtre ou bien chercher à améliorer la manipulation des détails en rajoutant des niveaux et en cherchant des paramètres de base qui correspondront à plus d'image. La reconstruction des images pourraient être faite différemment, par exemple en passant par des gradients comme dans l'article [7].

Ressources complémentaires

Ci-dessous sont disponibles toutes les ressources mentionnées dans ce rapport.

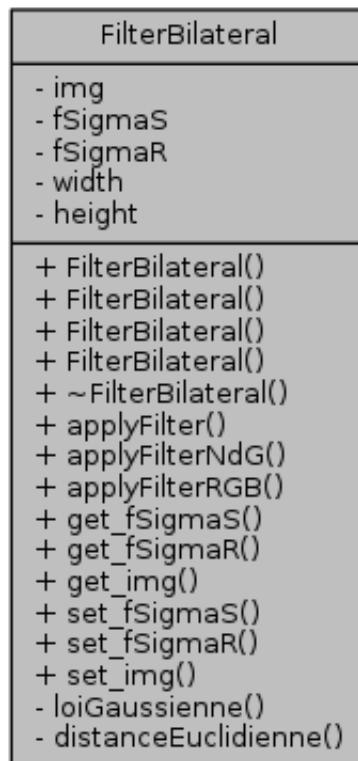


FIGURE A.1 – Graphe de collaboration de la classe `FilterBilateral`

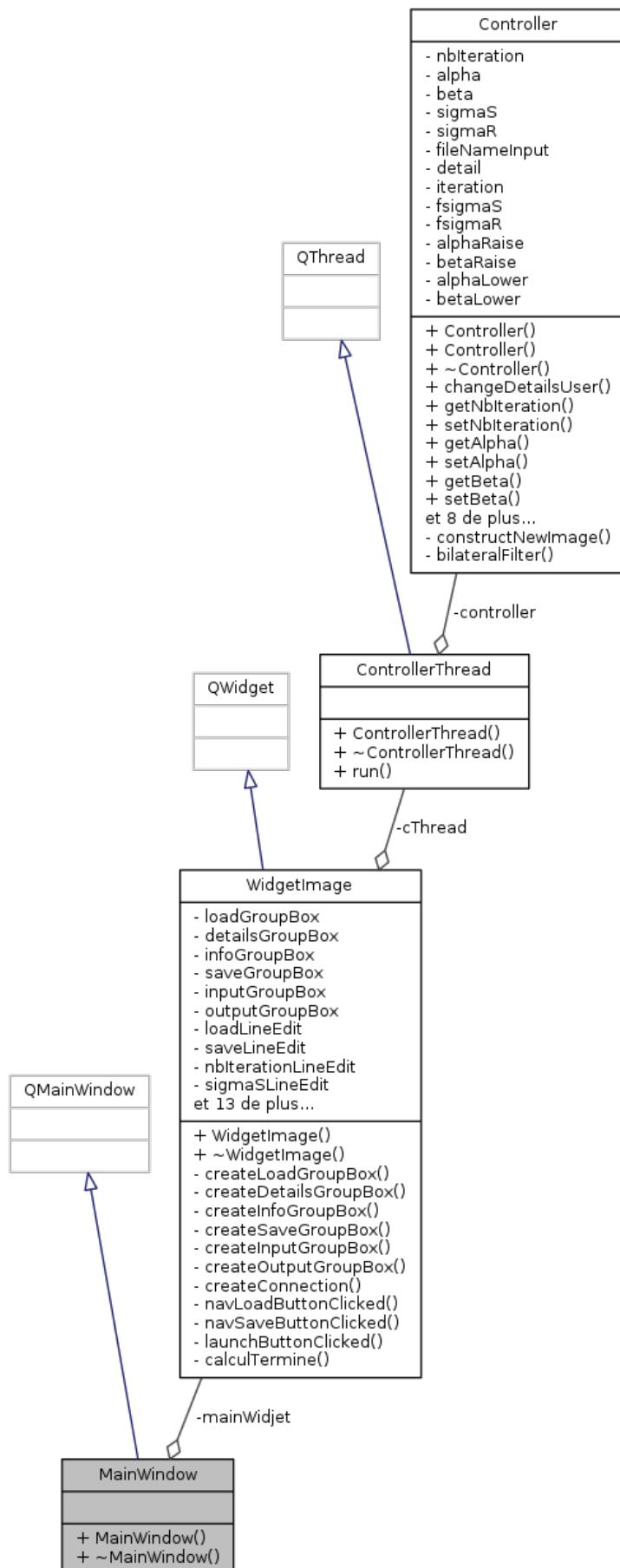


FIGURE A.2 – Graphe de collaboration de la classe **MainWindow**

Index

C++, 35

CImg, 15, 16, 39, 42

Controller, 37

ControllerThread, 37

Débruitage, 12, 15

Espace colorimétrique, 10

FilterBilateral, 37

Filtre bilatéral, 8–12, 37

Filtre gaussien, 8, 12

IHM, 35

Luminance, 10

MainWindow, 35

MVC, 35

Pyramide multi-échelle, 21

QGroupBox, 36

QMainWindow, 36

QPushButton, 36

Qt, 35

QWidget, 36

RGB, 10

WidgetImage, 36

YCbCr, 10

Bibliographie

- [1] V. Aurich and J. Weule. Non-linear gaussian filters performing edge preserving diffusion. *Proceedings of the DAGM Symposium*, 1995.
- [2] S. M. Smith and J. M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1), 1997.
- [3] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Proceedings of the IEEE International Conference on Computer Vision*, 1998.
- [4] The cimg library. cimg.sourceforge.net/.
- [5] Hdri-example. Sous licence CC BY-SA 3.0 via Wikimedia Commons - commons.wikimedia.org/wiki/File:HDRI-Example.jpg#/media/File:HDRI-Example.jpg.
- [6] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3), july 2004.
- [7] Zeev Farbman Raanan Fattal Dani Lischinski Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics*, 27(3), 2008.
- [8] Fast bilateral filter - paris and durand. people.csail.mit.edu/sparis/bf/.
- [9] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. *Computer Vision - ECCV 2006*, 2006.
- [10] Implementation matlab du filtre bilateral par jiawen chen. <http://people.csail.mit.edu/jiawen/>.

Développement d'un outil de traitement d'images par filtrage bilatéral

Département Informatique
5^e année
2014 - 2015

Rapport de projet de fin d'études

Résumé : Ce rapport de projet de fin d'études porte sur l'étude du filtre bilatéral. À partir de ce filtre, un travail sur la décomposition des images a été fait afin de pouvoir manipuler les détails de celle-ci. Par la suite une IHM permettant à l'utilisateur de pouvoir modifier comme il le souhaite une image a été réalisée. Une autre méthode permettant d'accélérer l'exécution du filtre a été étudiée et ajoutée à l'IHM.

Mots clefs : Filtre bilatéral, ClImg, Qt, C++, Traitement d'images, Convolution

Abstract: This graduation report is about the study of the bilateral filter. From this filter, a work on image decomposition was done to be able to manipulate the details from it. Then, a HCI with which the user can change like he wants an image was created. An other method allowing to accelerate the execution of the filter has been studied and added to HCI.

Keywords: Bilateral filter, ClImg, Qt, C++, Image filtering, Convolution

Encadrants
Moncef HIDANE
moncef.hidane@insa-cvl.fr

Étudiante
Natacha MARLIO-MARETTE
natacha.marlio-marette@etu.univ-tours.fr