NXP Semiconductors
Université de Paris Est
École doctorale ICMS
ESIEE, IGM, Lab Info, A3SI

THESE
pour obtenir le grade de Docteur de l'Université de Paris Est
Spécialité Informatique

Méthodes et algorithmes de dématriçage et de filtrage du bruit pour la photographie numérique

présentée et soutenue publiquement par

Harold Phelippeau

le 3 avril 2009

Directeurs de thèse:

Mohamed Akil	Professeur, ESIEE, UPE, UMR CNRS 8049
Hugues Talbot	Professeur, ESIEE, UPE, UMR CNRS 8049

Composition du jury:

Rapporteur:	Jean-Pierre Cocquerez	Professeur, UTC, UMR CNRS 6599
Rapporteur:	Michel Paindavoine	Professeur, Le2i, UMR CNRS 2309 STIC
Rapporteur:	Marc Van Droogenbroeck	Professeur, ULg, DEESC
Examinateur:	Stefan Bara	Ingénieur de recherche, NXP Semicondutors
Examinateur:	Thierry Bernard	Enseignant chercheur, ENSTA, LEI

Résumé

Ces dernières années, les appareils-photos/vidéos numériques grand-public sont devenus omniprésents. On peut aujourd'hui trouver des systèmes de captures d'images dans toutes sortes d'appareils numériques comme les téléphones portables, les assistants personnels numériques etc. Malgré une augmentation croissante de la puissance et de la complexité de ces appareils, la qualité de la chaîne de capture d'image, composée du couple « système optique/capteur » est toujours contrainte à des limitations d'espace et de coût. Les défauts introduits sont nombreux et dégradent considérablement la qualité des images produites : flou, déformations géométriques, artéfacts de couleurs, effets de moiré, bruits statiques et dynamiques, etc. Une idée intéressante est de corriger ces défauts de manière algorithmique en utilisant la puissance toujours croissante des architectures de traitements. Dans cette thèse nous nous intéressons particulièrement à deux problèmes issues de l'acquisition de l'image par le capteur : le dématriçage de la matrice de Bayer et la réduction du bruit. Dans la première partie, nous décrivons la structure générale de la chaîne de capture d'image dans les appareils-photos/vidéos numériques. Nous présentons le rôle, le fonctionnement et les défauts introduits par chacun de ses éléments. Enfin, nous illustrons comment ces défauts peuvent être corrigés par des traitements algorithmiques. Dans la deuxième partie, nous montrons comment l'information de couleur est introduite dans les capteurs numériques. Nous présentons ensuite un état de l'art des algorithmes de dématriçage. Un nouvel algorithme de reconstruction de la matrice de Bayer basé sur le principe de l'interpolation directionnelle est proposé. Il permet d'associer une qualité d'image produite sans artéfacts avec une faible complexité de calculs. Pour mieux comprendre les comportements du bruit dans les capteurs numériques, nous énumérons ses différentes sources et leurs dépendances par rapport aux conditions de prises de vues. Après avoir présenté l'état de l'art des méthodes de restauration des images bruitées, nous nous intéressons particulièrement aux algorithmes de débruitage à voisinage local et plus précisément au filtre bilatéral. Nous proposons un filtre bilatéral pour la mosaïque de Bayer, adaptatif en fonction de la puissance du bruit dans les images. Dans la troisième partie, nous présentons l'implémentation, l'optimisation et la simulation de l'exécution des algorithmes de dématriçage et de réduction du bruit proposés. La plateforme d'implémentation est le processeur TriMedia TM3270 de NXP semiconductors. Nous montrons que nous arrivons à traiter des images de taille 5 méga-pixels en moins de 0,5 secondes et des images de résolution VGA à une cadence supérieure à 25 images par seconde. Finalement, pour des raisons de standardisation, de rapidité d'exécution et de consommation d'énergie, nous avons conçu une architecture dédiée à l'algorithme de dématriçage proposé. Cette architecture permet de multiplier par 10 la rapidité d'exécution obtenue sur le processeur TriMedia TM3270.

Mots-clés : photographie, capteur, restoration, bruit, dématriçage, traitement temps-réel.

Abstract

Digital cameras are now present everywhere. They are commonly included in portable digital devices such as mobile phones and personal digital assistants. In spite of constant improvements in terms of computing power and complexity, the digital imaging chain quality, including sensor and lenses system, is still limited by space and cost constraints. An important number of degradations are introduced by this chain that significantly decrease overall image quality : including blurring effects, geometric distortions, color artifacts, moiré effects, static and dynamic noise. Correcting these defects in an algorithmic way, using the increasing power of embedded processing architecture present in mobile phones and PDAs may appear like an interesting solution. In this thesis we are especially interested in reducing two major defects of the sensor acquisition chain : Bayer matrix demosaicing artifacts and photon noise.

In the first part, we describe the general imaging chain commonly used in digital cameras and video devices. We show the function, the inner working and the defects introduced by each of its elements. Finally we exhibit possible ways to correct these defects using algorithmic solutions. In the second part, we introduce the principle of Bayer demosaicing. We present the state of the art and we propose a new method based on a directed interpolation principle. Our method yields a good image quality while retaining a low computational complexity. We then enumerate several noise sources present in imaging digital sensors and their dependencies with imaging conditions. We are particularly interested in local algorithms and more specifically in the bilateral filter. After presenting the state of the art in denoising algorithm, we propose a new adaptive bilateral filter for sensor colour mosaic denoising. In the third part, we present the implementation, the optimization and the execution simulation of the proposed demosaicing and denoising algorithms. The implementation target is the TM3270 TriMedia processor from NXP Semiconductors. We show that it is possible to process 5 megapixels images in less than 0.5 seconds and more than 25 images per second at VGA resolution. Finally, for standardization, execution speed and power consumption reasons, we describe a dedicated architecture for our proposed demosaicing algorithm. This architecture improves the execution speed by a factor of 10 compared to the TriMedia TM3270 processor.

Keywords : photography, sensor, restoration, noise, demosaicing, real-time processing.

Remerciements

J'exprime mes sincères remerciements à Gilles Bertrand pour m'avoir accueilli au sein du laboratoire A3SI et de m'avoir permis de mener à bien mon projet de recherche.

Mes remerciements s'adressent également à Marc Gavard, Cécile Kohler et Stefan Bara pour m'avoir accueilli au sein de l'équipe IVIC (Image and Video Innovation Center) de NXP Semiconductors.

Je tiens à remercier Mohamed Akil, mon directeur de thèse, pour son encadrement, sa disponibilité, son sens de la communication et sa rigueur scientifique.

Je remercie Hugues Talbot, co-directeur de ma thèse, pour son encadrement, son enthousiasme, sa vivacité d'esprit, ses conseils avertis et les nombreuses impulsions qu'il a apporté à mes travaux.

Je remercie vivement Stefan Bara, pour la confiance qu'il m'a accordée, sa disponibilité et son enthousiasme. Il a su préserver durant ces trois années l'équilibre fragile existant entre les impératifs industriels et les exigences de la recherche.

Je tiens aussi à remercier Jean-Pierre Cocquerez, Michel Paindavoine et Marc Van Droogenbroeck qui m'ont fait l'honneur d'être rapporteurs de mon manuscrit de thèse.

J'exprime ma reconnaissance à Thierry Bernard d'avoir accepté d'être examinateur de ma thèse.

Une part de mes remerciements vont vers mes collègues du laboratoire A3SI et IVIC, je pense particulièrement à Stefan Bara, Eva Dokladalova, Yukiko Kenmochi, Petr Matas, Nicolas Ngan, Olena Tankyevych et tous les autres ...

Je remercie ma famille, mon père, ma mère et mes frères pour l'affection et le soutien constant qu'ils m'ont quotidiennement apporté.

Enfin, je remercie Michaela, pour sa bonne humeur, sa joie de vivre et le soutien qu'elle a su m'apporter dans les moments difficiles.

Table des matières

Table des figures

Liste des tableaux

Tuto disertační práci věnuji své lásce ...

Chapitre 1

Introduction

1.1 Cadre et motivations

Depuis l'apparition des appareil-photos/vidéos numériques sur le marché grand public dans les années 90, la qualité de prise de vue et la puissance de traitement des appareils n'ont cessé d'augmenter. Les fonctionnalités de l'imagerie numérique se sont développées extrêmement rapidement, notamment sur les téléphones et assistants personnels portables. Les exigences en terme de qualité de perception des images produites sont confrontées aux contraintes d'encombrements mécaniques dûs à la miniaturisation, de coûts et d'architectures systèmes. Du fait des contraintes de coût et d'encombrement mécanique, les chaînes de capture optiques et électroniques introduisent de nombreux défauts, tels que, des déformations géométriques, des aberrations chromatiques, un manque de netteté, des artéfacts de couleurs, des effets de moiré, une forte présence de bruits statiques et dynamiques etc. La capture et la numérisation des images sont effectuées par des capteurs CMOS (Complementary Metal Oxyde Semiconductor) ou CCD (Charged-Coupled Device) au travers d'optiques miniatures. Une puce multimédia intégrant une architecture et des logiciels de traitements dédiés, reçoit les données provenant du capteur pour produire une image visualisable (dématriçage de l'image du capteur, réglage de la balance des blancs, zoom, compression, etc.). L'image est ensuite envoyée vers un écran LCD (Liquid Crystal Display) pour l'affichage et vers la mémoire pour le stockage.

Les capteurs d'images numériques sont sujets à deux défauts majeurs influant de manière importante sur la qualité de l'image produite : l'interpolation de la matrice de Bayer [11] et la présence de bruits statiques et dynamiques. En effet, les capteurs ne sont pas naturellement sensibles à la couleur. Celle-ci est introduite par la superposition d'un filtre coloré sur la surface photosensible (le plus populaire est le filtre de Bayer [11]).

Ce filtre est composé d'une mosaïque particulière des trois couleurs primaires, rouge, vert et bleu. Une étape d'interpolation est nécessaire pour produire une image colorée en estimant pour chaque pixel les deux composantes de couleurs manquantes. Cette interpolation est d'une importance primordiale concernant la qualité des images. Elle est susceptible d'apporter de nombreux artéfacts, tels que, des fausses couleurs, des effets de moiré, du flou, des effets de grille et l'introduction de structures en formes de labyrinthes. D'autre part, les capteurs numériques introduisent de nombreux bruits statiques et dynamiques. Ces bruits, d'origines et de statistiques variées, dégradent la qualité visuelle des images. Ils introduisent une perte de netteté dans les détails et font apparaître des granularités et des tâches colorées. Ces deux défauts sont communs à la majorité des capteurs d'images utilisés dans les appareils photos/vidéos numériques. Outre leurs influences sur la qualité des images, ces défauts sont susceptibles de perturber les performances de post-traitements tels que la compression ou la détection et le suivi d'objets, etc.

Dans cette thèse, étant donné la puissance croissante des architectures de traitements utilisées en photographie numérique, nous proposons de corriger ces défauts par des traitements algorithmiques. On s'attachera à mettre au point une méthode de dématriçage n'introduisant pas ou peu d'artéfacts et une méthode de réduction du bruit performante. Ces propositions algorithmiques devront respecter les capacités de traitements des architectures considérées et les contraintes de temps réel imposées par les applications que sont la photographie et la vidéo numérique.

1.2 Organisation de la thèse et contributions

La présente thèse, se décompose en trois parties :

Première partie :

Le chapitre 2, rappelle brièvement l'histoire de la photographie, depuis les premiers systèmes de projections d'images utilisés à l'époque de la renaissance, en passant par la naissance de la photographie argentique dans les années 1820, sa popularisation dans les années 1880 et allant jusqu'à l'apparition des premiers appareil-photos/vidéos numériques dans les années 1980.

Le chapitre 3, décrit les aspects matériels et logiciels des systèmes de captures d'images numériques. Dans la première section, nous présentons les différents éléments constituant la chaîne d'acquisition d'un appareil-photo/vidéo numérique : le système de lentilles, le capteur numérique et la puce multimédia. Nous expliquons le rôle et le fonctionnement

de chacun de ces composants et leurs influences variées sur la qualité des images produites. Dans la deuxième section, nous mettons en avant l'aspect algorithmique des systèmes de captures d'images numériques. Nous montrons comment l'image issue du capteur est manipulée pour produire une image visualisable et comment les traitements algorithmiques permettent de corriger les défauts introduits par la chaîne d'acquisition.

Deuxième partie :

Le chapitre 4, répertorie de façon non-exhaustive les principaux algorithmes de dématriçage proposés dans la littérature. Nous comparons les qualités des images produites en utilisant la mesure objective de l'erreur quadratique moyenne (RMS, Root Mean Squarre) et le critère subjectif de l'appréciation visuelle. Nous comparons aussi les complexités de calculs des algorithmes en comptant le nombre d'opérations effectuées pour traiter un pixel. Nous mettons en avant les bons résultats d'adéquations obtenus entre les qualités d'images produites et les faibles complexités de calculs des algorithmes de dématriçage par interpolations directionnelles. Nous soulignons la forte dépendance existante entre la complexité algorithmique des estimateurs de directions des détails utilisés et la qualité des images produites.

Le chapitre 5, présente un nouvel estimateur de direction d'interpolation basé sur un calcul de gradients dans les plans verts interpolés verticalement et horizontalement. Complémentairement nous proposons une méthode de minimisation des erreurs d'estimations par homogénéisation locale des choix de directions, nous appelons cette méthode LMDC (Local Majority Direction Choice). Nous généralisons l'application de cette méthode à tous les algorithmes fonctionnant par interpolations directionnelles. Nous montrons qu'elle permet d'augmenter de manière importante la qualité des images produites. Enfin, nous présentons une variante de l'algorithme de réduction des artéfacts d'interpolations proposé dans [12]. On remplace l'utilisation du filtre médian sur la différence des canaux de couleurs par l'utilisation d'un filtre bilatéral. Nous montrons, que le filtre bilatéral permet d'améliorer la réduction des artéfacts colorés tout en réduisant la complexité de calculs de l'algorithme. L'association de l'estimateur de directions d'interpolations proposé avec la méthode de minimisation des erreurs LMDC permet de formuler un nouvel estimateur appelé GED (Green Edge Direction). Cet estimateur, utilisé dans le cadre du dématriçage par interpolations directionnelles permet de formuler un nouvel algorithme de dématriçage appelé GEDI (Green Edge Directed Interpolation). Les mesures de qualités objectives du RMS et subjectives de l'appréciation visuelle permettent de classer GEDI parmi les algorithmes de la littérature produisant les meilleures qualités d'images dématriçées. Parmi ces algorithmes, nous montrons que GEDI possède

la plus faible complexité algorithmique. **Ces travaux ont fait l'objet de deux brevets européens [13, 14] et d'une soumission au journal IEEE Transactions on Consumer Electronics [15].**

Le chapitre 6, dans sa première section énumère les différentes sources de bruits présentes dans les capteurs numériques, les propriétés statiques où dynamiques des bruits générés et leurs dépendances variées par rapport aux conditions de prises de vues. Nous présentons dans la deuxième section comment ces bruits sont modélisés dans la littérature. Dans la troisième section, nous présentons l'état de l'art des méthodes de restauration des images bruitées. Nous nous intéressons particulièrement, pour des raisons d'architecture système, aux méthodes de filtrage à voisinage local. Ces méthodes possèdent de faibles complexités de calculs, sont non-itératives et permettent de traiter les images à la volée dans un masque de taille définie, sans avoir besoin de mémoriser l'image entière. Ces méthodes sont bien adaptées aux architectures de traitement des systèmes de captures d'images embarqués de type appareil-photos numériques compacts et téléphones mobiles. En effet, ceux-ci possèdent des puissances de traitements limitées, peu de mémoire et peu de ressources de calculs. Nous comparons les performances des qualités de débruitage produites par ces filtres en utilisant la mesure de RMS et l'appréciation visuelle. Nous montrons que le filtre bilatéral [16] permet d'obtenir le compromis le plus intéressant entre le filtrage du bruit et la préservation des détails de la scène. D'autre part, le filtre bilatéral possède l'avantage de pouvoir régler son paramètre de similarité photométrique. Nous proposons d'exploiter l'ajustement de ce paramètre pour rendre le filtre adaptatif en fonction de la puissance du bruit dans l'image.

Le chapitre 7, rappelle dans la première section, les propriétés du filtre bilatéral, notamment comment influe le réglage du paramètre de similarité photométrique sur le filtrage des images. Dans la deuxième section, nous justifions l'utilisation d'une statistique de Poisson pour modéliser les bruits des capteurs d'images numériques. En considérant les caractéristiques statistiques poissonniennes du signal de l'image et les propriétés de contrôle du gain utilisé dans les appareil-photos/vidéos numériques, nous proposons une méthode d'estimation du meilleur paramètre de similarité photométrique du filtre bilatéral en fonction de la puissance du bruit dans l'image. Cette estimation permet de formuler un filtre bilatéral adaptatif. Nous simulons l'acquisition d'images par un capteur numérique sous différentes illuminations et nous montrons la capacité du filtre proposé à s'adapter au meilleur compromis entre le filtrage du bruit et la préservation des détails en fonction de la puissance du bruit dans l'image. Dans la troisième section, nous proposons une formulation du filtre bilatéral permettant de filtrer la mosaïque de Bayer. Le nouveau filtre formulé est appelé filtre bilatéral Bayer. On suppose que filtrer l'image avant l'étape de dématriçage permettra de mieux discriminer le signal utile du bruit, favorisant ainsi un meilleur filtrage. En effet, filtrer l'image de mosaïque permet d'avoir accès

au données bruts du capteur, avant qu'elles ne soient dispersées à travers les différents canaux de couleurs par l'étape de dématriçage. D'autre part, on suppose que la présence de bruit dans l'image de mosaïque perturbe le fonctionnement des algorithmes de reconstruction. Pour valider cette hypothèse, nous comparons les qualités d'images produites par un filtrage bilatéral appliqué avant et un filtrage bilatéral appliqué après l'étape de reconstruction de la matrice de Bayer. Les mesures de RMS obtenues montrent des qualités d'images produites équivalentes. Cependant, les qualités visuelles de ces images sont très différentes. Les image filtrées avant dématriçage apparaissent moins bruitées, plus nettes et ne contiennent pas d'artéfacts colorés. Nous formulons ensuite un filtre bilatéral Bayer adaptatif en combinant la méthode d'estimation du bruit proposée et le filtre bilatéral Bayer. Nous simulons l'acquisitions d'images par un capteur numérique couleurs sous différentes intensités d'illumination. Les expérimentations réalisées montrent que le nouveau filtre possède les qualités d'adaptation au bruit du filtre bilatéral adaptatif et les avantages du filtre bilatéral Bayer pour le filtrage des images issues des capteurs d'images numériques couleurs. **Ces travaux ont fait l'objet d'une publication [17] à IEEE ICSP2008 (International Conference On Image Processing)**.

Troisième partie :

Dans cette partie, nous détaillons dans les deux premiers chapitres, l'implémentation, l'optimisation et la simulation de l'exécution des algorithmes de dématriçage et de débruitage proposés sur le processeur multimédia TM3270 de NXP Semiconductors. Dans un troisième chapitre, nous proposons une architecture dédiée pour l'algorithme de dématriçage GEDI. Le cahier des charges nous impose de traiter des images de résolution 5 méga-pixels en moins de 0.5 secondes et des images de résolution VGA (640×480) à une cadence minimale de 25 images par seconde.

Le chapitre 8, présente l'implémentation, la simulation et l'optimisation d'algorithmes de dématriçage sur le processeur TriMedia TM3270. Nous comparons les performances d'exécutions des algorithmes GEDI, Hirakawa [18], Hamilton [8] et Hamilton corrigé par la méthode de LMDC. Nous détaillons les différentes étapes d'optimisations appliquées, optimisations algorithmiques, optimisations standards (décalages de bits, utilisations de types de données appropriés, utilisation de Look Up Table, déroulage de boucles, etc.) et l'utilisation du jeu d'instructions TriMedia. Finalement, nous comparons les différents résultats obtenus. Nous montrons, que GEDI permet de traiter une image de 5 méga-pixels en 0,23 secondes et des images de résolution VGA à la cadence de 50,5 images par seconde, soit 9 fois plus rapidement que l'algorithme de Hirakawa [18] qui produit une qualité d'image équivalente. **Ces travaux ont fait l'objet d'une publication [17] à DASIP2007 (Design and Architecture for Signal and Image Processing)**.

Le chapitre 9, présente l'implémentation, la simulation et l'optimisation du filtre bilatéral Bayer sur le processeur TriMedia TM3270. Nous détaillons les différentes étapes d'optimisations utilisées. Nous montrons, que malgré l'application de toutes ces techniques, le cahier des charges imposé n'est pas respecté. Une image de taille 5 méga-pixels est traitée en 0,92 secondes et les images de résolution VGA sont traitées avec une cadence de 17,78 images par seconde. Nous proposons une nouvelle formulation du filtre bilatéral Bayer permettant d'utiliser les ressources du processeur TriMedia TM3270 de façon optimale. La formulation du filtre proposée permet de traiter des images de résolution 5 méga-pixels en 0,35 secondes et de traiter 46,9 images de résoluation VGA par seconde. **Ces travaux ont fait l'objet d'une publication [19] à IS&T/SPIE2009**.

Le chapitre 10 présente une architecture dédiée pour l'algorithme de dématriçage GEDI. L'architecture est divisée en quatre fonctions. Ces fonctions correspondent aux quatre étapes nécessaires pour l'application de l'algorithme. Considérons M la largeur de l'image traitée, l'architecture possède une latence de $10 \times M + 24$ cycles, une taille mémoire de $166 \times M - 562$ bits et utilise 1258 éléments séquentiels. Elle permet de traiter une image de 5 mégas-pixels en 0,0336 secondes et 474,66 images de résolution VGA par seconde, soit une rapidité d'exécution 10 fois plus élevée que sur le processeur TM3270.

Première partie

La photographie numérique

Chapitre 2

Histoire, apparition et généralisation de la photographie numérique

La photographie numérique regroupe l'ensemble des procédés physiques et logiciels permettant d'obtenir une image photographique à partir d'un capteur électronique (CMOS ou CCD) utilisé comme surface photosensible. La photographie numérique est l'aboutissement de plusieurs siècles de recherche et de volonté par l'homme de pouvoir capturer de manière instantanée les images du monde qui l'entoure. Dans ce chapitre, nous rappelons brièvement l'histoire et l'évolution de la photographie numérique, depuis sa naissance jusqu'à nos jours.

2.1 Histoire de la photographie

L'homme perçoit majoritairement le monde extérieur avec son système de vision. Il a depuis toujours essayé de reproduire et de figer ce qu'il voit. Longtemps, le dessin et la peinture sont restés les seuls moyens de représentation du monde. L'ancêtre de l'appareil-photo, la chambre noire, qui permet de projeter sur une surface plane l'image d'une scène était déjà connue à l'époque d'Aristote (384-322 av. J.-C.). Ce principe, couplé à des systèmes optiques, est utilisé par certains artistes à l'époque de la renaissance pour faciliter le tracé des objets en perspective. Le dispositif physique permettant de projeter l'image d'une scène sur un support étant connu, il reste alors une étape importante à franchir : comment faire en sorte que cette image s'imprime sur le support de façon stable et durable ? En 1826, J.N. Niépce (1765-1833), réussit à obtenir et à conserver une image fixe et durable grâce à l'utilisation du chlorure d'argent. En 1833, l'utilisation de plaques

de cuivre recouvertes de iodure d'argent exposées ensuite à des vapeurs de mercure améliore le procédé de fixation et diminue de manière importante le temps d'exposition. L'invention du négatif par W.H.Fox-Talbot(1800-1877) permet de reproduire plusieurs images à partir d'une seule exposition. Vers la fin des années 1880, l'utilisation de la photographie est limitée par son coût et sa complexité. Toutefois, quand en 1888 George Eastman (1854-1932) lance le Kodak, un appareil-photo portatif très maniable et doté d'une pellicule, la voie s'est dégagée pour le photographe amateur. Quand un client avait pris ses photos, il retournait l'appareil entier à l'usine. La pellicule était traitée, l'appareil rechargé, puis réexpédié avec des photos développées. Le slogan « Appuyez sur le bouton, nous ferons le reste » n'avait rien d'exagéré. Les milliards de clichés pris chaque année indiquent que son succès ne s'est jamais démenti. Aujourd'hui, la popularité de la photographie s'est accrue grâce à l'arrivée des appareil-photos numériques. Ils offrent une grande souplesse et une convivialité d'utilisation. Le contrôle du résultat est immédiat par prévisualisation sur un écran LCD. Les images sont instantanément disponibles, il est possible de les effacer, de les reproduire et de les échanger. Le nombre de prises de vues est plus élevé, en effet, l'utilisateur n'a plus la crainte de rater une photographie. Il est aussi possible de corriger et de manipuler ultérieurement les images sur un ordinateur.

2.2 Apparition et généralisation de la photographie numérique

Jusqu'en 1981, il était nécessaire d'utiliser des systèmes d'appareil-photos analogiques pour convertir un signal lumineux en un signal électrique. En 1981, Sony introduit le système Mavica, capable d'enregistrer les images sur une disquette magnétique. Ainsi commença l'histoire de la photographie numérique grand public. En 1988, Fuji Photo Film annonce le premier appareil-photo numérique, le Fujix DS-1P [20], celui-ci utilise une carte mémoire pour enregistrer les images. A l'exposition Photokina [21] de 1990, Fuji Photo Film et Olympus exposent le prototype des appareil-photos numériques d'aujourd'hui. Il est composé d'un ASIC (Application Specific Integreted Circuit) [22] qui permet une compression algorithmique standardisée en 1992 par le JPEG (Joint Photographic Expert Group) [23]. Apple introduit en 1994 le QuickTake 100 [24, 25] comme appareil-photo pour ordinateur. En 1995, le QV-10 de Casio [26] connaît un succès commercial et ouvre une nouvelle ère dans la consommation et l'utilisation des appareil-photos numériques. A la Photokina de 1996, Olympus présente le XGA C-800L [27], cet appareil est capable de traiter des images de 810 000 pixels. Le XGA améliore ainsi la qualité des images obtenues numériquement, en se rapprochant de la qualité des appareils argentiques. En 1997, Olympus présente le C-1400 [28], c'est le premier appareil destiné au marché grand public qui possède une dalle CCD contenant plus de 1 million de pixels.

A partir de 1999, les capteurs de plus de 1 million de pixels se généralisent. En 2003, Canon présente le premier réflexe numérique grand publique, l'EOS 300D [29] aussi appelé Rebel. A partir de 2004, les capteurs d'images numériques font leur apparition dans les téléphones mobiles, principalement avec une résolution VGA (640×480). Suivant une évolution croissante, on trouve aujourd'hui des appareil-photos réflex numériques possédant des dalles photosensibles allant jusqu'à 12 millions de pixels et des téléphones portables intégrants des capteurs de 5 millions de pixels. La figure 2.1 montre différents types d'appareil-photos numériques.

(a) Appareil-photo numérique réflex Nikon D40x, avec optique interchangeable

(b) Appareil-photo numérique compact Canon Ixus 860 IS

(c) Appareil-photo intégré dans un téléphone mobile Sagem myC5-2m

FIGURE 2.1 – Exemples de différents appareil-photos numériques.

Chapitre 3

Aspect matériel et logiciel de la photographie numérique

Dans ce chapitre, nous décrivons les aspects matériel et logiciel de la photographie numérique. Nous présentons dans une première section les principaux composants de la chaîne d'acquisition de l'image, leurs rôles et leurs influences sur la qualité de celle-ci. Dans une deuxième section, nous décrivons l'aspect algorithmique de la photographie numérique, son rôle, son lien direct avec l'aspect matériel et son influence sur la qualité des images produites par l'appareil.

3.1 Photographie numérique : aspect matériel

Un appareil-photo numérique est constitué de trois principaux éléments. Un système optique, dont le rôle est de projeter l'image de la scène sur la surface photosensible. Un capteur numérique, qui joue le rôle de surface photosensible et dont le but est de capturer, d'échantillonner et de numériser l'image projetée à sa surface par le système optique. Enfin, une architecture de traitement du signal qui à pour rôle de produire une image visualisable, cela correspond à l'étape de développement de la photographie argentique. La figure 3.1 illustre la constitution générale d'un appareil-photo numérique avec ses différents éléments. Chacun de ces éléments joue un rôle particulier dans le processus d'acquisition et de formation de l'image. Une description complète des éléments composant les appareil-photos numériques est disponible dans le livre de R.Lukac « Single Sensor Imaging : Methods and Applications for Digital Cameras » [30].

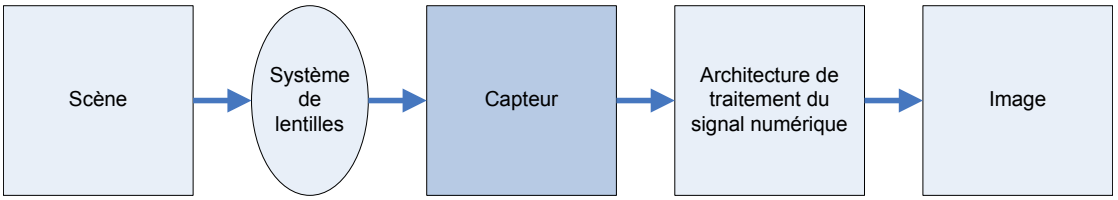


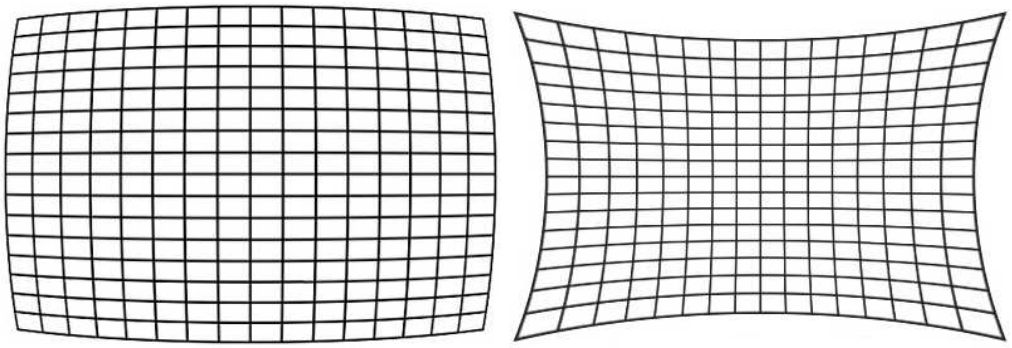
FIGURE 3.1 – Schéma général d'un appareil-photo numérique, il contient un système optique composé de lentilles, un capteur numérique et une architecture de traitement du signal numérique.

3.1.1 Le système optique

Le signal lumineux issu de la scène et formant un flux de photons, passe d'abord à travers un système de lentilles pour être projeté sur la surface du capteur numérique. Cette première étape de capture et de formation de l'image de la scène, commune à la photographie numérique et argentique, est régie par les propriétés de l'optique géométrique, elle est fondamentale et non triviale. Nous en rappelons les principes généraux dans l'annexe A. Les systèmes optiques des appareil-photos sont constitués d'arrangements complexes de lentilles, le choix de ces systèmes à un impact important sur la qualité de prise de vue et représente un fort pourcentage du prix de l'appareil. Il peut être fixe pour les appareils compacts ou interchangeable pour les appareils réflex. On trouvera des études approfondies des systèmes optiques dédiés à la photographie dans l'ouvrage de T.Koyama [31] et dans la thèse de doctorat de P.B. Catrysse [32]. Les systèmes de lentilles n'étant pas idéaux, ceux-ci introduisent de nombreux artéfacts. Parmi les plus important, on citera les aberrations géométriques (distorsions géométriques), les aberrations chromatiques, le vignetage et le flou. Ces aberrations sont inhérentes à la structure des systèmes optiques, à la forme (bombée), aux propriétés optiques du matériau constituant les lentilles ainsi qu'aux propriétés de projection de l'image d'une scène sur une surface plane.

Les déformations géométriques sont introduites par la présence d'un diaphragme situé devant ou derrière le système optique et lorsque les conditions menant à l'approximation de Gauss ne sont pas respectées (angles des rayons d'incidences faibles et point d'incidence proche de l'axe optique), on pourra se référencer aux livres [33, 34] pour plus de détails sur les origines de ces déformations. Ces distorsions se traduisent par une courbure des lignes droites du sujet ou de la scène photographiée. Il existe deux types de distorsions géométriques, en barillet ou en coussinet, suivant la position du diaphragme par rapport au système optique. Si le diaphragme est positionné devant le système optique, la distorsion est en forme de barillet, comme cela est illustré sur la figure 3.2(a). L'objectif produit alors une image plus grande de la partie centrale du sujet. En conséquence, les lignes droites du sujet sont incurvées vers l'extérieur. Si le

diaphragme est positionné après le système optique, la distorsion est en forme de coussinet, comme cela est illustré sur la figure 3.2(b). L'objectif produit une image plus petite de la partie centrale du sujet. En conséquence, les lignes droites du sujet sont incurvées vers l'intérieur. L'image de la figure 3.3 montre un exemple de distorsion géométrique en forme de barillet.



(a) (b)

FIGURE 3.2 – Illustration des distorsions géométriques en forme de barillet (a) et en forme de coussinet (b).

La majorité des systèmes optiques font apparaître un obscurcissement graduel de l'image partant du centre et évoluant vers ses bords. Ce défaut est introduit par une insuffisance des systèmes optiques à concentrer la lumière à leur périphérie, c'est le vignetage. Le vignetage est fort lorsque l'ouverture du diaphragme est forte et s'atténue pour des ouvertures faibles. Ce phénomène est accentué par une augmentation du contraste de l'image. Un exemple est montré sur la figure 3.3. On pourra se référer aux livres [35, 36] pour des explications détaillées de ce phénomène.

Les systèmes optiques introduisent aussi des aberrations chromatiques. En effet, l'indice de réfraction du matériau composant les lentilles (généralement du verre) varie en fonction de la longueur d'onde de la lumière. Il en résulte que la distance focale varie en fonction de la longueur d'onde, de sorte que la mise au point ne peut être effectuée simultanément pour toutes les couleurs du spectre lumineux. Si, par exemple, la mise au point est effectuée pour la longueur d'onde correspondante à la couleur rouge, l'image d'un objet blanc présente alors sur ses bords une irisation bleutée. La figure 3.4 illustre le principe de formation des aberrations chromatiques. On peut voir sur la figure 3.5 un exemple d'aberrations chromatiques.

3.1.2 Les capteurs numériques d'images

En photographie numérique, les films photosensibles de la photographie argentique sont remplacés par des capteurs numériques d'images. Un capteur numérique d'images est

FIGURE 3.3 – Cette image montre à la fois un exemple de vignetage et un exemple de distorsion géométrique. On distingue clairement l'insuffisance de luminosité dans les coins de l'image (vignettage). Les lignes droites de la scène sont projetées comme des courbes (distorsions géométriques).

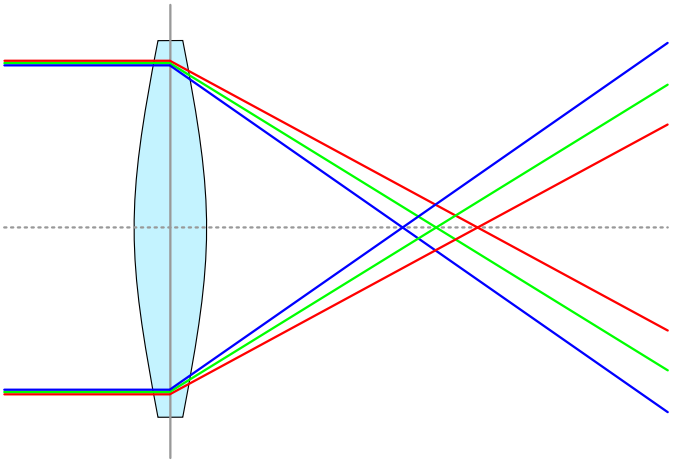


FIGURE 3.4 – Principe de formation des aberrations chromatiques, la distance focale varie en fonction de la longueur d'onde, l'image créée n'est donc pas nette sur l'ensemble du spectre visible.

capable de détecter la lumière dans un large spectre de fréquence, partant des rayons X et allant jusqu'à la lumière infrarouge. En photographie, on s'intéresse uniquement à la bande visible du spectre, correspondante à la réponse spectrale de l'œil humain, c'est à dire, pour des longueurs d'ondes variant entre 380 et 780 nm. Ces capteurs sont constitués d'une dalle de photorécepteurs fabriqués à partir de silicium (cellules MOS (Metal-Oxyde Semiconductor), voir figure 3.7), dont le rôle est de transformer par effet photoélectrique le signal lumineux constitué de photons, en signal électrique (charge)

(a) (b)

FIGURE 3.5 – Cette image montre un exemple d'aberrations chromatiques, les contours du bras de saint Venceslas sont irisés de bleu. Ici, les franges bleus sont non-isotropes car l'objet n'est pas centré sur l'axe optique du système.

constitué d'électrons. Ils permettent ainsi de faire l'acquisition et l'échantillonnage en deux dimensions de l'image projetée sur sa surface par le système optique. Le signal analogique résultant est ensuite numérisé à travers un convertisseur analogique/numérique et envoyé vers une architecture de traitement pour produire une image visualisable. Un capteur numérique est caractérisé par sa résolution spatiale, ses dimensions, la dimension de ses photosites, son rendement quantique (rapport entre le nombre d'électrons produits et le nombre de photons incidents), sa sensibilité spectrale et son rapport signal sur bruit [31]. Le nombre d'applications basées sur les capteurs numériques d'images ont rapidement augmentées ces dernières années. Notamment, on peut noter leur standardisation dans les appareils de téléphonie mobile, la vidéo-surveillance, l'industrie automobile, la détection et le contrôle automatique dans les chaînes industrielles. Il existe aujourd'hui deux principales technologies de capteurs, la technologie des capteurs CCD (Charge-Coupled Device) et la technologie des capteurs CMOS (Complementary Metal Oxyde Semiconductor). De manière générale, ces dernières années, les capteurs CCD sont utilisés dans les systèmes privilégiant la qualité de l'image, et les capteurs CMOS dans les systèmes privilégiant les faibles coûts de production (rentabilité), miniaturisation et faible consommation. Cette règle n'est pas stricte et tend à évoluer avec les progrès de chacune de ces technologies.

3.1.2.1 L'effet photoélectrique

Le fonctionnement des photorécepteurs qui composent les capteurs est basé sur l'emploi du silicium. Le silicium est le matériau de base de la quasi-totalité des circuits intégrés analogiques et numériques et s'avère être un détecteur optique performant. La détection de la lumière se produit par effet photoélectrique. Lorsqu'un photon percutant le capteur possède une énergie $h\nu$ supérieure à la bande d'énergie (band gap) E_g du silicium, tels que :

$$E_{photon} = h.\nu = \frac{h.c}{\lambda} \geq E_g \tag{3.1}$$

où h, c, ν et λ sont respectivement, la constante de Planck, la vitesse de la lumière, la fréquence de la lumière, cela se traduit par la génération d'une paire électron-trou dans le semi-conducteur [37], comme illustré sur la figure 3.6. Les photo-électrons générés sont ensuite collectés dans la zone de déplétion de la capacité MOS (voir figure 3.7). La charge générée est proportionnelle au nombre de photons ayant percutés le photorécepteur et défini l'intensité (niveau de gris) du pixel correspondant dans l'image. Un dispositif de transfert de charges achemine les charges générées vers la sortie du capteur pour l'étape de conversion analogique/numérique.

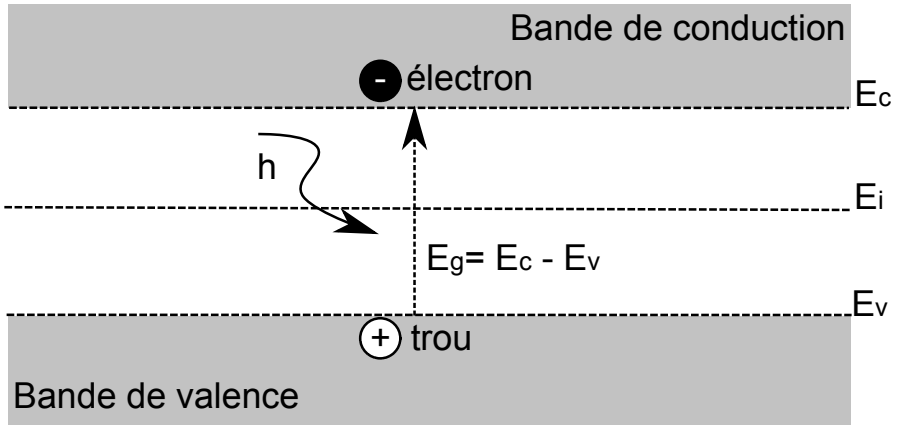


FIGURE 3.6 – Effet photoélectrique dans un semi-conducteur : l'absorption d'un photon d'énergie $h\nu$ se traduit par la génération d'une paire électron-trou [38].

3.1.2.2 Les systèmes de micro-lentilles

L'augmentation constante de la résolution des appareils pour des tailles de capteurs similaires se traduit par la diminution de la surface photosensible des photorécepteurs. La quantité de lumière percutant un photosite devient alors de plus en plus faible, diminuant ainsi sa sensibilité et augmentant le rapport signal/bruit du capteur. Afin d'augmenter

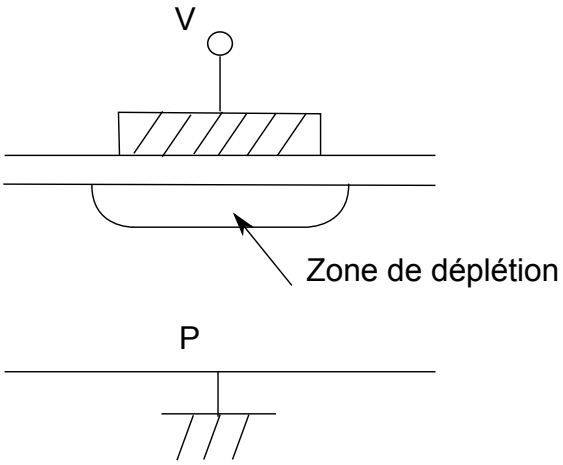


FIGURE 3.7 – Schéma d'une cellule MOS.

la quantité de lumière percutant les photorécepteurs pour une période d'exposition similaire, des systèmes de micros-lentilles positionnés sur la surface des photorécepteurs ont été mis au point. Le premier capteur CCD possédant une dalle de micro-lentilles à été proposé en 1986 par Y.Ishihara et K.Tanigaki dans [39]. La figure 3.8 montre le principe de fonctionnement d'une micro-lentille.

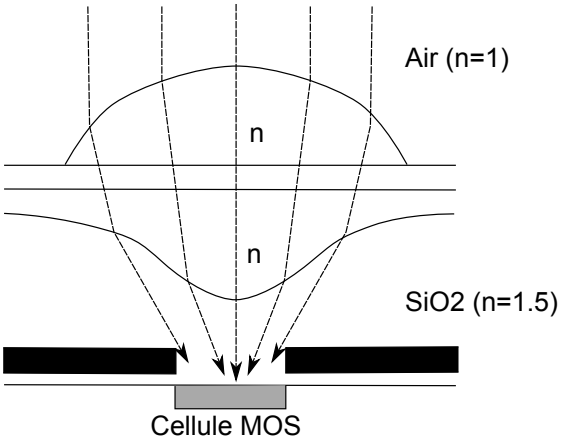


FIGURE 3.8 – Schéma d'une micro-lentille [40].

3.1.2.3 L'introduction de la couleur

Les capteurs CCD et CMOS sont monochromatiques. En effet, les photosites ne font pas de distinctions entre les longueurs d'ondes du signal lumineux. Ils sont seulement capables de mesurer la quantité de lumière (ou le nombre de photons) ayant percuté chaque photosite. Il existe différentes méthodes de sensibilisations des capteurs à la couleur, décrites ci-après.

Dans les systèmes mono-capteurs, la sensibilisation à la couleur est introduite par la superposition d'un filtre coloré sur la surface du capteur, comme on peut le voir sur la

figure 3.9. Ces filtres permettent de multiplexer spatialement l'information chromatique. Différentes mosaïques de couleurs sont proposées dans la littérature [2, 3, 30, 41]. La figure 3.10 montre des exemples de différents arrangements de couleurs [3]. La mosaïque de filtre la plus utilisée est la mosaïque de Bayer [11] introduite par Kodak en 1976, elle est aussi appelée mosaïque GRGB. Ce filtre a la particularité d'utiliser deux fois plus d'éléments verts que d'éléments rouges et bleus pour se rapprocher du système de vision humain, qui possède une plus grande sensibilité pour les longueurs d'ondes proches du vert. Une étape d'interpolation des deux composantes de couleurs manquantes pour chaque pixel est nécessaire pour produire une image en couleurs. Cette étape d'interpolation est appelée demosaicing ou demosaicking, en français, dématriçage ou encore reconstruction. En « fabricant » les deux tiers de l'information de l'image finale, l'étape de dématriçage est d'une importance primordiale pour la qualité visuelle (flou, effet de moiré, apparitions de fausses couleurs, apparitions de détails erronés).

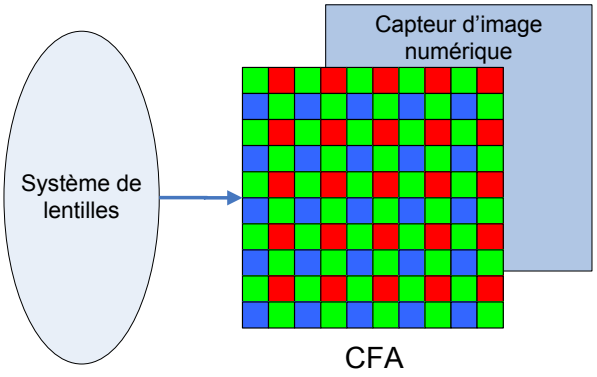
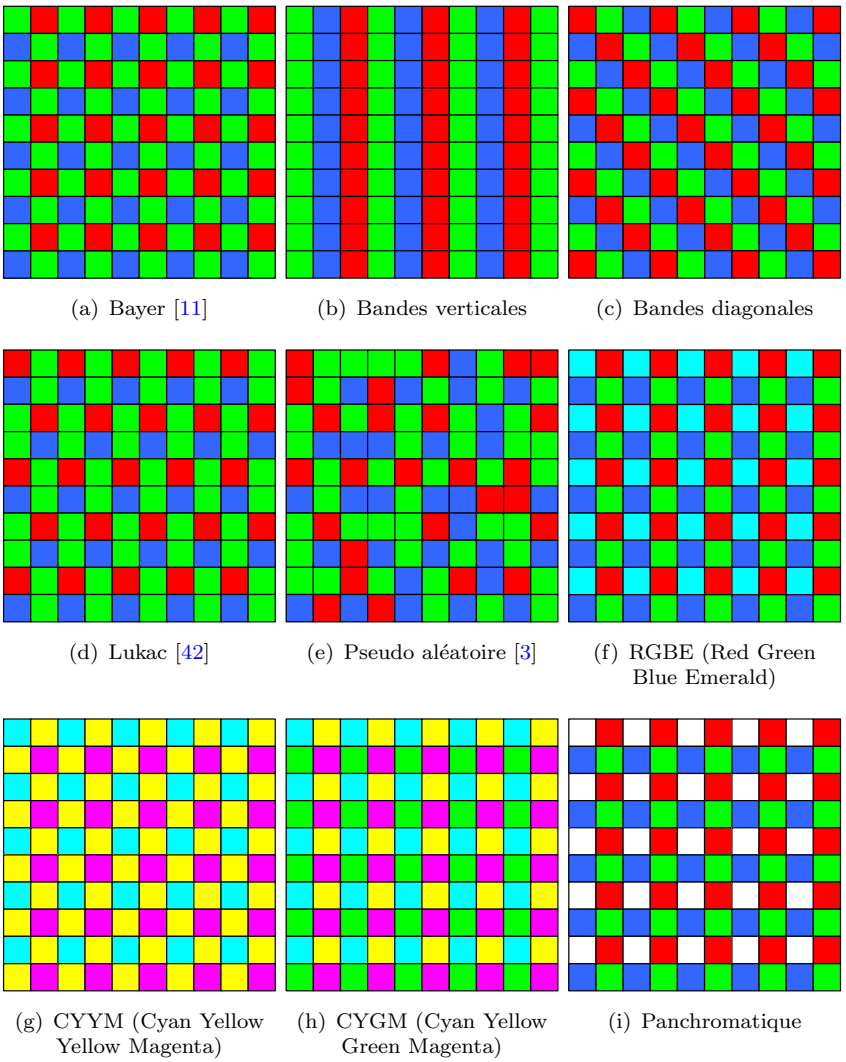


FIGURE 3.9 – Superposition d'un filtre de couleur (CFA : Color Filter Array) sur la surface du capteur d'image pour introduire l'information de couleur.

La société Fuji propose le capteur super-CCD, possédant des photorécepteurs de formes octogonales plutôt que carrées (voir 3.11). Cette géométrie particulière permet de produire une image de plus grande résolution et d'augmenter la sensibilité du capteur à la lumière (augmentant ainsi le rapport signal/bruit). Une filtre coloré est superposé sur ce capteur, et une étape de reconstruction est nécessaire pour produire une image en couleurs.

Pour éviter le multiplexage chromatique spatial et l'étape problématique de la reconstruction, la société Foveon propose le capteur Foveon X3, qui permet d'acquérir avec un seul capteur, les trois composantes de couleurs primaires pour chaque photorécepteur. Par analogie avec le fonctionnement des films photosensibles des appareil-photos argentiques, les capteurs Foveon X3 utilisent sur propriété de la lumière à pénétrer à travers la couche de silicium avec des profondeurs variant en fonction de la longueur d'onde. Un exemple est illustré sur la figure 3.12. Ce capteur permet d'éviter les artéfacts du à l'interpolation de la mosaïque de couleur des systèmes monocapteurs classiques, la

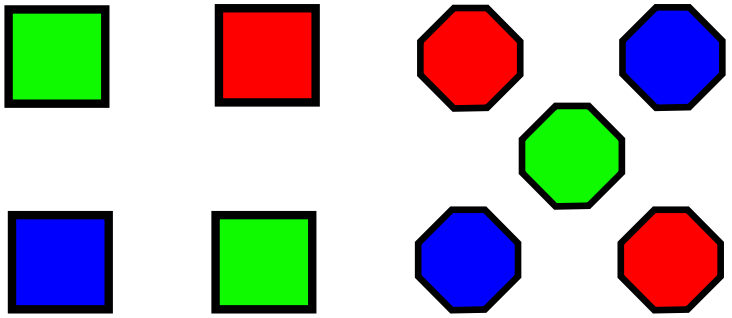


(a) Bayer [11] (b) Bandes verticales (c) Bandes diagonales

(d) Lukac [42] (e) Pseudo aléatoire [3] (f) RGBE (Red Green Blue Emerald)

(g) CYYM (Cyan Yellow Yellow Magenta) (h) CYGM (Cyan Yellow Green Magenta) (i) Panchromatique

FIGURE 3.10 – Exemples de différentes mosaïques de filtres colorés.



(a) Disposition des photosites dans un capteur classique (b) Disposition des photosites dans un capteur super-CCD

FIGURE 3.11 – Comparaison des dispositions des photorécepteurs entre un capteur classique et un capteur super-CCD.

couleur est plus pure, l'image plus nette et la résolution plus importante. Cependant, la transmission et la réflexion des couleurs à travers les différentes couches du silicium

n'étant pas idéales, les images produites sont grisâtres. D'autre part, ces capteurs restent particulièrement sensibles au bruit et leur technologie est encore coûteuse.

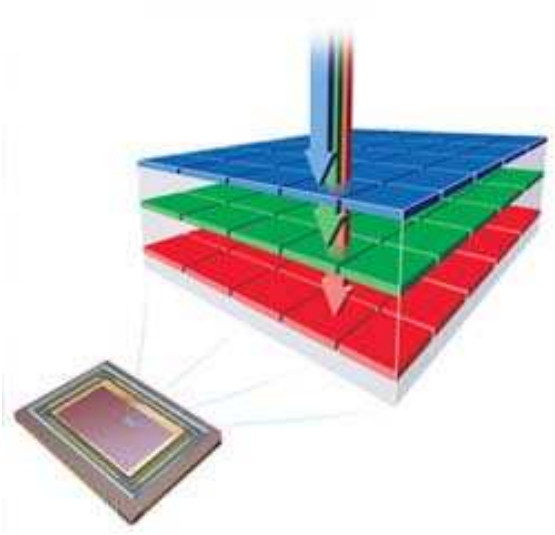


FIGURE 3.12 – Principe de fonctionnement d'un capteur FoveonX3, les ondes lumineuses ayant des longueurs d'ondes correspondant aux couleurs bleu, verte et rouge pénètrent à travers le silicium avec des profondeurs différentes.

Certains appareils de capture d'image, en particulier les caméscopes, sont équipés de systèmes de captures tri-CCD. Cette technologie n'est pas utilisée dans les appareils compacts, car elle est trop coûteuse, trop encombrante et trop fragile. Les caméras couleurs tri-CCD sont équipées d'un dispositif à base de prismes permettant d'orienter le signal lumineux vers trois capteurs CCD. Un capteur CCD est dédié à l'acquisition de chaque composante de couleur primaire (figure 3.13).

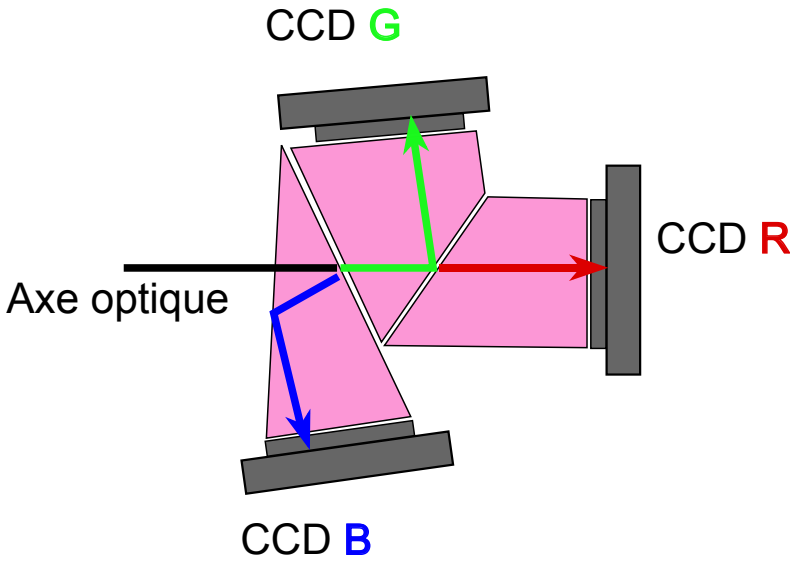


FIGURE 3.13 – Fonctionnement d'un capteur tri-CCD, un système de prisme projette chaque composante de couleur primaire sur un capteur distinct.

3.1.2.4 Formats des capteurs

Les formats des capteurs sont définis par les longueurs de leurs diagonales. Ils sont l'héritage des caméras à tube, dont les diamètres typiques étaient mesurés en pouces « " ». Un tube de 1 pouce avait une fenêtre active rectangulaire de 16 mm de diagonale. Elle n'est pas égale au 24 mm de la métrique anglo-saxonne. On trouve généralement des capteurs de 1/3", 1/2.7" et 1/2". Les tailles des pixels varient de $2.2\mu m \times 2.2\mu m$ à $16\mu m \times 16\mu m$ et leurs nombres de 500×500 à 5000×5000. Le rapport hauteur/largeur est généralement de 4/3. La figure 3.14 montre différentes tailles de capteurs utilisées, le tableau 3.1 répertorie les correspondances des différents formats optiques des capteurs et leurs tailles effectives respectives.

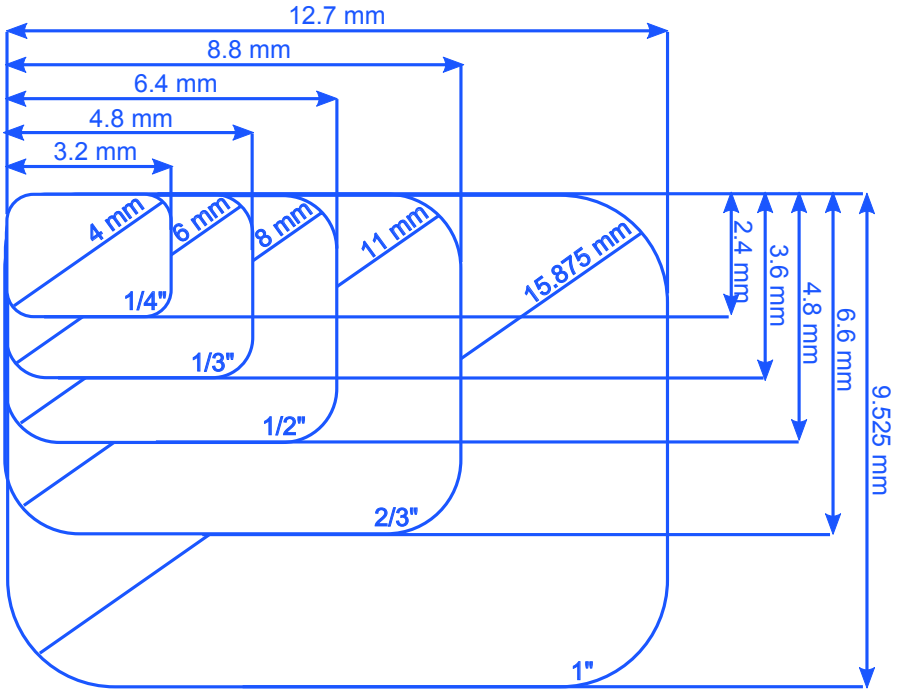


FIGURE 3.14 – Taille des capteurs numériques.

TABLE 3.1 – Formats optiques et tailles des capteurs

Formats en "	Diagonale en mm	Hauteur en mm	Largeur en mm
1	16.0	12.80	9.60
2/3	11.0	8.80	6.60
1/1.8	8.89	7.11	5.33
1/2	8.00	6.40	4.80
1/2.5	7.20	5.76	4.32
1/2.7	6.67	5.33	4.00
1/3	6.00	4.80	3.60
1/3.2	5.63	4.50	3.38
1/4	4.50	3.60	2.70
1/5	3.60	2.88	2.16
1/6	3.00	2.40	1.80

3.1.2.5 Capteurs CCD et CMOS

Un capteur CCD est composé d'une dalle de cellules photosensibles (MOS). Le processus de fabrication impose que ces photorécepteurs soient adressés de manière séquentielle. La conversion de la charge en tension et l'amplification se font à la sortie du capteur. Pour accéder à la valeur d'un pixel, il faut décaler et vider toutes les charges entre le registre de sortie et le pixel à lire. Cette opération est coûteuse en temps et élimine le contenu de tous les pixels situés entre le pixel traité et la sortie du capteur. L'architecture des capteurs CCD est très variée, il existe des capteurs CCD à pleine trame (généralement utilisés en photographie), les capteurs CCD à transfert de trame (systèmes nécessitant de haute dynamiques d'acquisitions), les capteurs CCD interlignes (généralement utilisés pour les systèmes vidéos) et les capteurs CCD en barrette (utilisés par exemple dans les scanners ou encore pour l'imagerie satellitaire (Spot)) [43]. Les utilisations de ces différents capteurs varient en fonction des besoins de l'application, notamment en termes de vitesse d'acquisition et de qualité d'image. La figure 3.15 présente l'architecture d'un capteur CCD pleine trame. Les capteurs CCD ont été utilisés très tôt dans les domaines exigeant de hautes performances de qualité d'image, comme l'astronomie, la photographie, les applications scientifiques et industrielles.

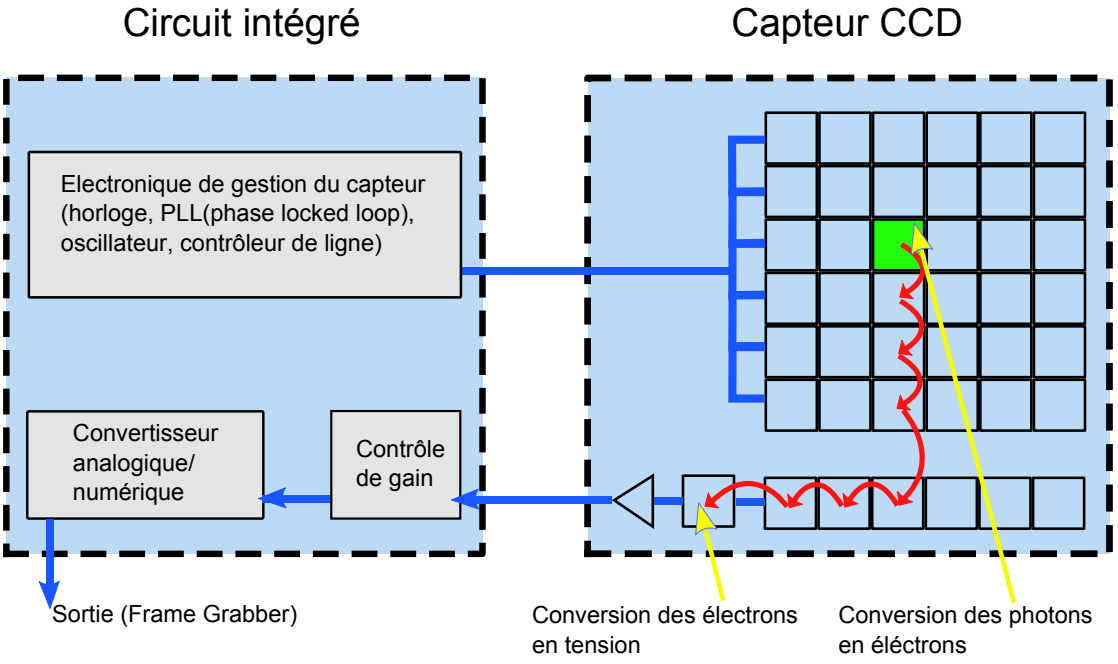


FIGURE 3.15 – Shéma d'un capteur d'image CCD [1]

Les capteurs CMOS utilisent le principe de pixel actif (APS : Active Pixel Sensor), qui associe au sein de chaque photosite, un photorécepteur, une diode de lecture et un circuit d'amplification [38]. Une matrice de commutation répartie sur l'ensemble de la puce permet d'accéder à chaque pixel de manière indépendante. Le principal avantage des capteurs CMOS par rapport aux capteurs CCD réside dans la possibilité d'accès

aléatoire aux pixels dans la matrice de photorécepteurs. En effet, il suffit d'appliquer numériquement l'adresse X et Y du pixel à lire pour en obtenir instantanément sa valeur. Ceci peut s'apparenter à la lecture d'une mémoire. La facilité d'accès aux pixels disponibles dans les capteurs CMOS permet une très grande flexibilité pour les traitements de données en temps réel : suivi d'une cible, focus dans une zone de l'image, etc. En contrepartie, dans un capteur CMOS le facteur de remplissage de photons est diminué par le fait qu'une partie de la surface du pixel est consacrée au circuit d'amplification. Ceci peut être corrigé par l'utilisation de micro-lentilles pour concentrer le flux lumineux. La technologie CMOS étant compatible avec le monde de la VLSI (Very Large Scale Integration), elle permet l'intégration de fonctionnalités sur la même puce que le capteur. Ceci permet d'obtenir des systèmes de taille réduite, une consommation d'énergie plus faible et un coût de fabrication moins élevée [38]. On citera par exemple leurs utilisations dans les rétines artificielles.

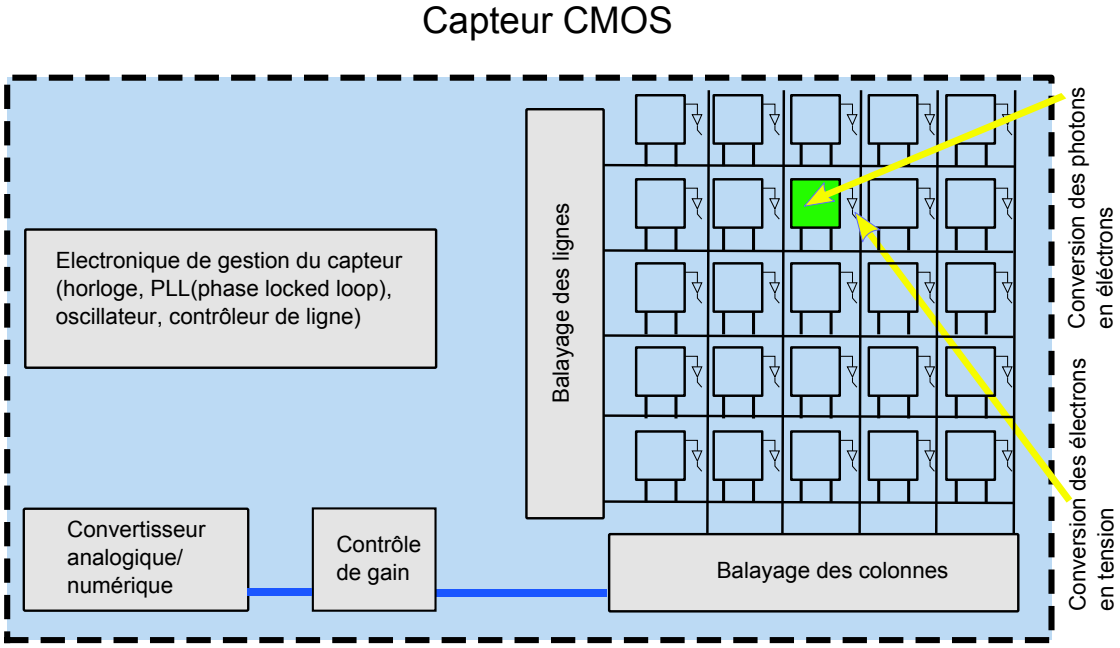


FIGURE 3.16 – Schéma d'un capteur d'image CMOS [1].

Aujourd'hui, les deux technologies permettent d'obtenir d'excellentes qualités d'images. Il n'y a pas de frontières distinctes concernant leurs applications. En effet, les concepteurs de capteurs CMOS multiplient leurs efforts concernant la qualité de l'image. Les concepteurs de capteurs CCD travaillent à réduire la consommation d'énergie. Il en résulte que l'on peut trouver des capteurs CCD dans des systèmes miniatures, à faible coût de production et de basse consommation comme la téléphonie mobile. D'autre part, on peut trouver des capteurs CMOS dans des systèmes industriels demandant de très hautes performances de qualité d'images, contredisant ainsi les stéréotypes habituels. Dans la pratique, les capteurs CCD sont plus chers, plus difficile a fabriquer, consomment plus d'énergie, mais de meilleure qualité. Les capteurs CMOS sont principalement réservé

au marché d'entré de gamme de type webcam, appareils téléphones mobiles et appareil-photos numériques compacts.

3.1.3 Architectures de traitements numériques du signal

Après l'échantillonnage et la conversion en signal électrique de l'image projetée sur la surface du capteur, celui-ci est numérisé à travers un convertisseur analogique/numérique, puis envoyé vers l'architecture de traitement numérique de signal. Dans le cas de la photographie numérique, l'architecture de traitement a pour rôle de produire une image visualisable pour l'utilisateur. Pour ce faire l'image issue du capteur numérique doit être traitée à travers divers algorithmes de construction et de corrections de l'image. On énumère ci-après les principaux algorithmes de traitements d'images que doit être capable d'appliquer une architecture dans un appareil de photographie numérique (nous développerons plus en détails l'aspect algorithmique dans la section 3.2) :

- reconstruction de l'image du capteur
- réduction du bruit
- équilibre des couleurs
- réglage de la luminosité et du gamma
- correction des aberrations géométriques
- redimensionnement des images (downsampling)
- compression
- etc.

Le choix de l'architecture de l'appareil est dicté par les contraintes d'encombrement mécanique (surface de silicium), de consommation d'énergie (autonomie de la batterie), de puissance et de qualité des traitements applicables, de rapidité d'exécution, de coût et de temps de développement, de flexibilité des traitements (mises à jour). On peut distinguer deux architectures de traitements en photographie numérique : l'architecture orientée DSP (Digital Signal Processor) et l'architecture de type ASIC.

Une architecture orientée DSP inclut typiquement un processeur de traitement du signal numérique combinée avec des SOC périphériques dédiés pour les traitements standards, comme par exemple la compression (voir figure 3.17). Le DSP est un microprocesseur optimisé pour les calculs numériques de traitement du signal tels que le filtrage, la détection et l'extraction de signaux, etc. Les architectures orientées DSP sont généralement dominantes dans les systèmes embarqués innovants. En effet, elle induit un temps de développement réduit et offre la possibilité de mise à jour des algorithmes. Ses inconvénients sont des temps de traitements non-optimaux et une consommation d'énergie importante. Dans les appareils dédiés à la photographie numérique (or téléphonie mobile

et PDA), les architectures de traitements sont essentiellement orientées DSP. Chaque fabriquant a d'ailleurs développé son propre processeur. Panasonic, avec la série de processeurs Venus, la dernière version étant le processeur Venus V (ou Venus HD)(présent dans les appareils Lumix DMC G1 et Lumix DMC TZ7). Sony, avec le processeur Bionz. Nikon avec le processeur Expeed. Enfin, Canon avec la série de processeurs Digic, la dernière version étant le processeur Digic III (il est présent dans les modèles compacts et professionnels tels que le Canon EOS 1000D). Chacun de ces fabricants met en avant les capacités de rapidité de traitements, de faible consommation, de qualité des traitements (rendu des couleurs, réduction du bruit, auto focus, etc). On comprend aisément que les architectures de ces processeurs restent hautement confidentielles. On pourra se référer aux derniers modèles d'appareil-photos des constructeurs pour comparer leurs différentes capacités (on pourra entre autres, utiliser le site internet « dpreview » qui permet de comparer la qualité des différents appareils du marché).

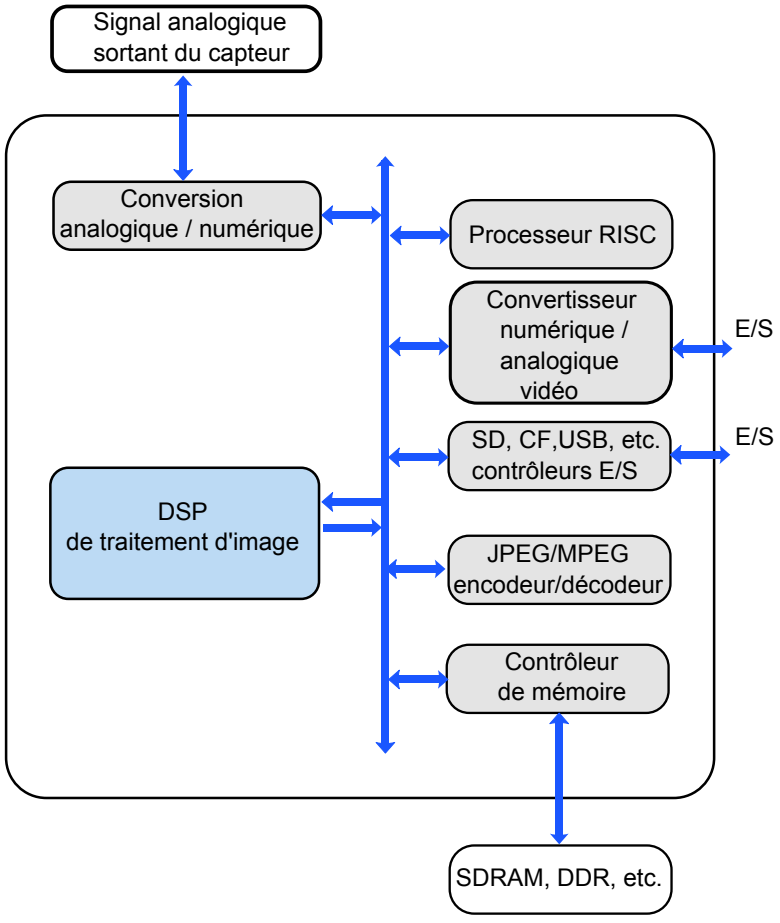


FIGURE 3.17 – Architecture de traitement d'images photographiques orientée DSP.

Une architecture de type ASIC est constituée d'un pipeline de circuits de traitements d'images dédiés, d'un processeur RISC (Reduced Instruction Set Computer) et de SOC périphériques (voir figure 3.18), l'architecture ASIC formant elle même un SOC. Cette

architecture est utilisée pour des applications dédiées, dont les traitements sont standards et bien maîtrisés, lorsque que des innovations algorithmiques sont peu probables. L'approche ASIC permet une exécution des traitements maximale et une consommation d'énergie réduite. Son désavantage est son coût de développement (design) et l'impossibilité de mise à jour des traitements.

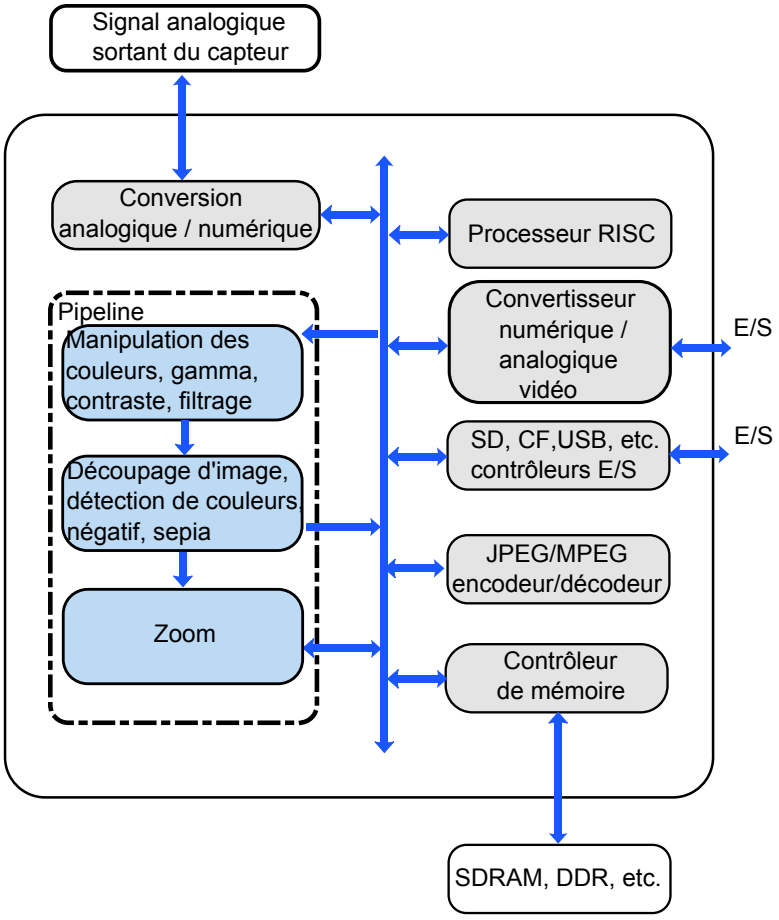


FIGURE 3.18 – Architecture de traitement d'images photographiques orientée ASIC.

3.2 Photographie numérique : aspect algorithmique

En photographie numérique, le signal délivré par le capteur n'est pas directement exploitable pour un utilisateur. Par analogie avec la photographie argentique, le signal imprimé par le capteur doit être développé pour produire une image visualisable. Pour ce faire, plusieurs étapes de traitements sont nécessaires, en passant par l'interpolation d'une image en couleurs, jusqu'à la correction des défauts introduits par le couple d'acquisition système optique/capteur. Ces traitements se font à travers l'architecture de traitement. Une première étape consiste à corriger les défauts introduits par les pixels défectueux. En effet, lors de la fabrication d'un capteur ou pendant sa durée de vie, certains pixels perdent leur sensibilité à la lumière (pixels morts) et créent des points

noirs dans l'image. D'autres pixels deviennent « hypers-sensibles » à la lumière et créent des points très lumineux. Ce défaut est assimilé à un bruit impulsionnel et peut être efficacement corrigé par l'application d'un filtre médian.

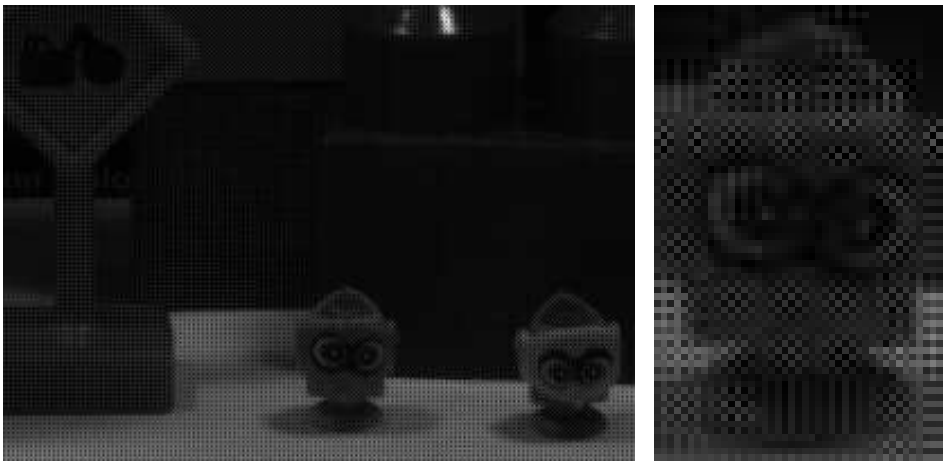
D'autre part, l'image de mosaïque des couleurs primaires, rouge, vert et bleu, doit être interpolée pour produire une image en couleur. Il s'agit dans cette étape de reconstruire les deux composantes de couleur manquantes pour chaque pixel. De nombreux algorithmes de dématriçage existent dans la littérature produisant différentes qualités d'images. Ce point est développé dans la section 4. La figure 3.19(a) montre une image en niveaux d'intensités telle qu'elle sort du capteur. Son équivalent démultiplexé sur les trois plans de couleurs, rouge, vert et bleu, est montré sur la figure 3.19(c). Cette image est ensuite reconstruite pour produire une image en couleur (voir figure 3.19(e)).

Une étape d'équilibrage des couleurs est aussi nécessaire, c'est la balance des blancs. La balance des blancs consiste à ajuster l'intensité de chaque canal de couleur primaire rouge, vert et bleu. L'intérêt de cet ajustement est d'obtenir sur la photographie de la scène le même rendu des couleurs que celui perçu par le système de vision humain. En effet, les couleurs transmises par une scène varient en fonction de la source lumineuse. Chez l'homme, l'œil neutralise la couleur de l'illuminant pour qu'une couleur de référence blanche soit toujours perçue blanche. Afin d'accorder les couleurs des photographies avec les couleurs que nous percevons, l'appareil-photo doit aussi posséder son propre système de neutralisation de l'illuminant. Les images de la figure 3.20 montrent un exemple d'équilibrage des couleurs.

Les traitements algorithmiques sont aussi utilisés pour corriger divers défauts introduits par la chaîne d'acquisition de l'image, tels que les distorsions géométriques, le vignetage, le flou ou encore la réduction du bruit. Ces traitements algorithmiques peuvent être intégrés directement dans l'architecture du système ou appliqués en dehors de l'appareil après avoir transféré les photographies sur un ordinateur personnel. La figure 3.21 montre quelques exemples de ces corrections.

3.3 Discussion

Dans ce chapitre, nous avons présenté la chaîne d'acquisition de l'image dans un capteur numérique et ses différents composants, le système de lentilles, le capteur numérique et l'architecture de traitements numériques. Nous avons montré le rôle de chacun de ces éléments dans le processus d'acquisition et de formation de l'image. Le système de lentilles permet de projeter l'image sur la surface photosensible, cette étape peut introduire de nombreux défauts tels que du flou, des déformations géométriques, des aberrations



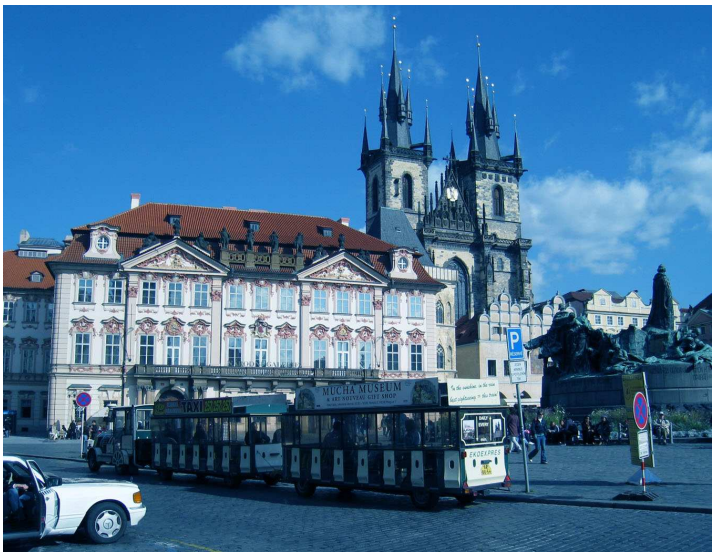
(a) Image en niveaux de gris sortant du capteur (b) Zoom de (a)

(c) Image démultiplexée sur les plans de couleurs rouge, vert et bleu (d) Zoom de (c)

(e) Image dématriçée (f) Zoom de (e)

FIGURE 3.19 – Illustration de l'étape de dématriçage de l'image du capteur : les deux composantes de couleurs manquantes de chaque pixel sont interpolées.

chromatiques, du vignetage, etc. Le capteur numérique permet d'échantillonner le signal lumineux et de le transformer en signal analogique grâce à l'effet photoélectrique.



(a) Image sans balance des blancs

(b) Image avec une balance des blancs correcte

FIGURE 3.20 – Illustration de l'adaptation de la balance des couleurs.

Cette transformation est perturbée par de nombreuses sources de bruit statiques et dynamiques, dépendantes des conditions de prises de vues (intensité lumineuse de la scène, température ambiante, réglage de l'ISO, etc.). Le signal analogique délivré par le capteur est ensuite numérisé puis envoyé vers l'architecture de traitement. Celle-ci va se charger, à travers différentes fonctionnalités hardwares et softwares, de traiter le signal pour produire une image visualisable (dématriçage, équilibre des couleurs, adaptation du contraste, etc.). Des améliorations importantes de la qualité des images peuvent être apportées par des traitements algorithmiques variés, permettant de corriger les défauts introduits par la chaîne de capture de l'image. Cependant ces traitements sont souvent complexes et difficilement supportés par les ressources des architectures de traitements

(a) Image originale (b) Correction du vignetage (c) Correction des distorsions géométriques en forme de barillet

(d) Image originale (e) Correction du flou

(f) Image originale (g) Correction du bruit

FIGURE 3.21 – Exemples de corrections des défauts introduits par les capteurs en utilisant des méthodes algorithmiques.

numériques présentes dans les appareils. Il est donc nécessaire de développer des algorithmes adaptés aux capacités de traitements de ce type d'architectures. Dans cette

thèse, nous nous focalisons sur deux problèmes fondamentaux concernant la qualité des images produites par les appareil-photos/vidéos numériques : le dématriçage de la matrice de Bayer et la réduction du bruit introduit par les capteurs.

Deuxième partie

Approches théoriques et algorithmiques

Chapitre 4

Dématriçage : État de l'art

Dans le chapitre 3, nous avons vu, que pour des raisons de coûts et aussi d'espace, un seul capteur CCD (Charge Coupled Device) ou CMOS (Complementary Metal Oxyde Semiconductor) est généralement utilisé pour convertir le signal lumineux en signal électrique. Ces capteurs ne sont pas naturellement sensibles à l'information de couleurs, ils sont seulement capables de mesurer la quantité de lumière (le nombre de photons) percutant chaque photosite. Pour introduire la sensibilité chromatique, un filtre coloré est superposé sur la surface des capteurs. Ces filtres sont composés d'une mosaïque des trois composantes de couleurs primaires rouge, vert et bleu. La combinaison de ces trois composantes par synthèse additive (voir figure 4.2) permet de reconstruire une partie de l'ensemble des couleurs dans le diagramme CIE 1931 xy du spectre visible, comme cela est illustré sur la figure 4.3. Nous avons montré dans le chapitre précédent (figure 3.10) différents exemples de mosaïques colorées proposées dans la littérature [2, 3, 41].

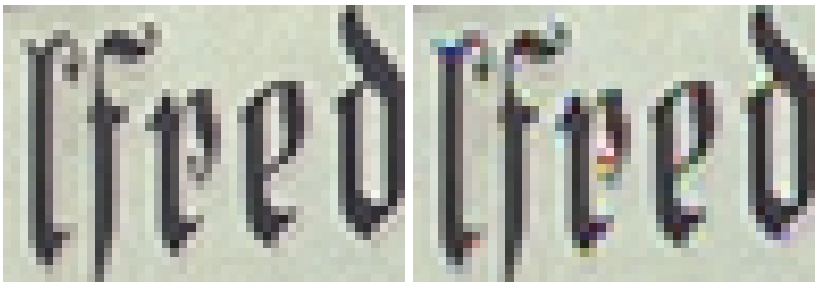
La mosaïque la plus populaire est celle proposée par Bayer dans [11], aussi appelée mosaïque GRGB. L'arrangement des couleurs proposé par Bayer permet d'obtenir une fréquence d'échantillonnage uniforme dans les directions horizontales et verticales. Elle est constituée d'une alternance de filtres rouges et verts une ligne sur deux, et d'une alternance de filtres bleus et verts sur les lignes restantes, de telle sorte que les éléments verts soit disposés en quinconce, comme cela est illustré sur la figure 4.4. On peut voir que l'arrangement de couleurs proposé par Bayer a la particularité d'utiliser deux fois plus d'éléments verts que d'éléments rouges et bleus, sans quoi l'uniformité d'échantillonnage dans les directions horizontale et verticale ne serait pas possible. Bayer justifie le choix du vert comme couleur de filtre dominant par analogie avec le système de vision humain, qui possède une plus grande sensibilité pour les longueurs d'ondes proches du vert [2].

En superposant un filtre de couleur à sa surface, le capteur reçoit une seule composante chromatique par photosite, faisant ainsi l'acquisition de seulement un tiers de l'information nécessaire pour produire une image colorée. Pour chaque pixel, les deux composantes chromatiques manquantes doivent être interpolées par une étape de dématriçage. C'est donc les deux tiers de l'image finale qu'il va falloir estimer, on comprend clairement que cette étape est d'une importance cruciale pour la qualité de l'image produite. Elle a une influence intrinsèque sur la netteté, l'effet de moiré, le rapport signal sur bruit, elle peut introduire des artéfacts de couleurs et des structures en formes de labyrinthes ou encore un effet de grille, tous ces défauts sont illustrés sur la figure 4.1. De nombreuses méthodes ont été mises au point, dont la majorité sont dédiées à la reconstruction de la mosaïque de Bayer [4, 8, 9, 9, 18, 44–52].

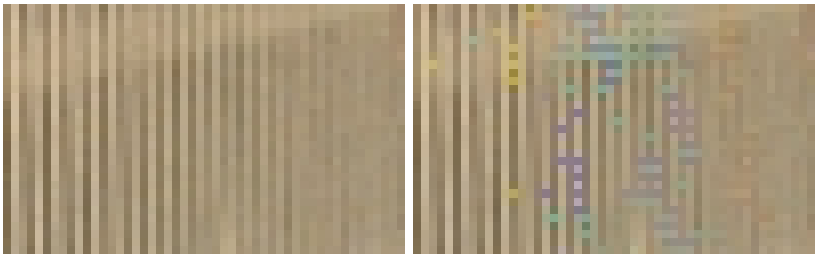
Le dématriçage peut être fait directement dans l'appareil-photo ou en dehors, sur un ordinateur personnel, en récupérant les données du capteur. Si la reconstruction est faite dans l'appareil-photo, il faut chercher le meilleur compromis entre la qualité visuelle produite et la complexité de l'algorithme. Si l'image est reconstruite en dehors de l'appareil-photo en utilisant les données bruts du capteur, l'utilisateur est dégagé des problèmes d'architectures matériels et peut se focaliser sur la qualité de l'image produite (manipuler l'image en dehors de l'appareil, permet à l'utilisateur d'utiliser les algorithmes de son choix et de les adapter en fonction des résultats escomptés). Sur la plupart des appareils du marché grand public, ces données ne sont pas disponibles, les utilisateurs ont seulement accès à l'image à la sortie de la chaîne de traitement. Il apparait donc indispensable pour de tels systèmes d'intégrer un algorithme de dématriçage performant en terme de qualité d'image produite en adéquation avec les capacités de traitement de l'architecture.

Depuis la prolifération des appareil-photos numériques sur le marché grand public et leur introduction dans toutes sortes d'appareils (téléphones mobiles, assistants personnels numériques, etc.), le dématriçage à connu un fort intérêt de la part des industriels et de nombreux algorithmes ont été mis au point. Dans ce chapitre, nous présentons l'état de l'art des algorithmes de dématriçage pour la mosaïque du filtre de Bayer uniquement. Le nombre de ces algorithmes est très important dans la littérature. Les méthodes utilisées sont très variées et exploitent l'ensemble des techniques de traitement du signal, il est impossible de tous les détailler dans cette thèse. Les nouveaux algorithmes proposés sont souvent des petites variantes ou améliorations des concepts de base. De manière générale, ces méthodes exploitent les propriétés de corrélations spatiales et spectrales des images pour interpoler judicieusement les composantes de couleurs manquantes dans la direction la plus homogène. Pour illustrer les calculs, nous utiliserons la numérotation de la matrice de Bayer présentée sur la figure 4.4.

(a) Image de la scène (b) Exemple de flou (c) Image de la scène (d) Exemple d'effet de moiré



(e) Image de la scène (f) Exemple d'artéfacts de couleurs



(g) Image de la scène (h) Exemple de structures en formes de labyrinthes

(i) Image de la scène (j) Exemple d'effet de grille

FIGURE 4.1 – Exemples de défauts introduits par l'étape de dématriçage.

4.1 Interpolation par copies de pixels et interpolation bilinéaire

Des méthodes très simples ont été proposées pour interpoler la matrice de Bayer. Ces méthodes traitent chaque plan de couleur de manière indépendante. Une première idée est de remplacer la couleur manquante par une couleur voisine existante en appliquant

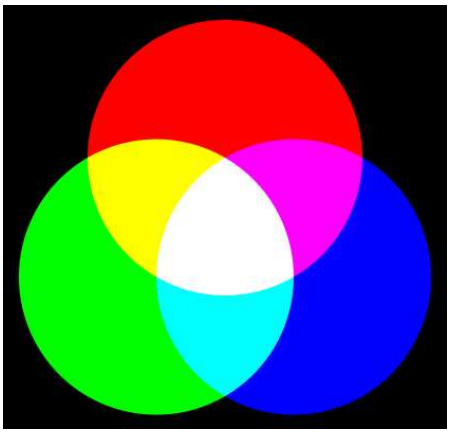


FIGURE 4.2 – Combinaison de trois composantes primaires rouge, vert et bleu par synthèse additive permettant de reconstituer une partie de l'ensemble des couleurs dans le diagramme CIE 1931 xy du spectre visible.

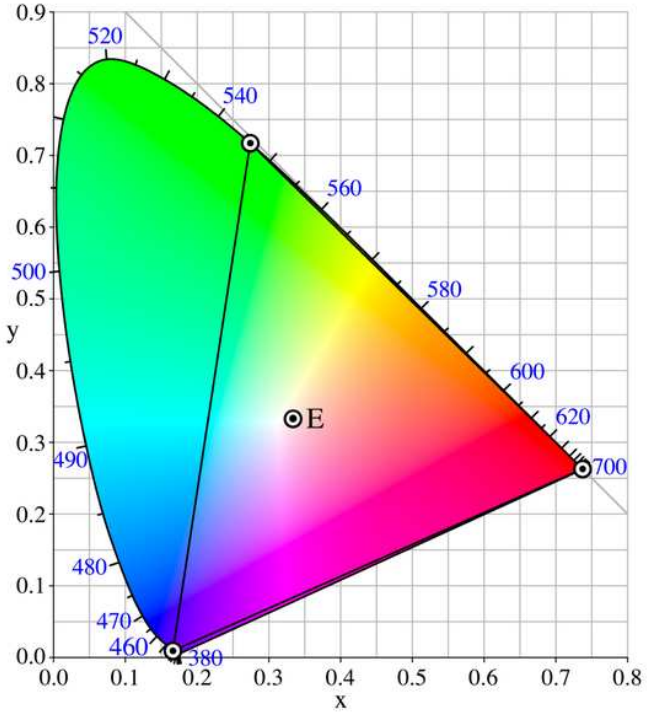


FIGURE 4.3 – Gamut de l'espace de couleur primaire RGB dans le diagramme de chromaticité CIE 1931 xy, E est le point des énergies égales.

un décalage. Cette méthode est illustrée sur la figure 4.5, dans laquelle les canaux rouge, vert et bleu ont été préalablement démultiplexés sur trois plans. Cette méthode est très rapide mais génère de nombreux artefacts de couleurs comme on peut le voir sur la figure 4.7.

Une autre idée simple est d'interpoler linéairement les composantes manquantes de

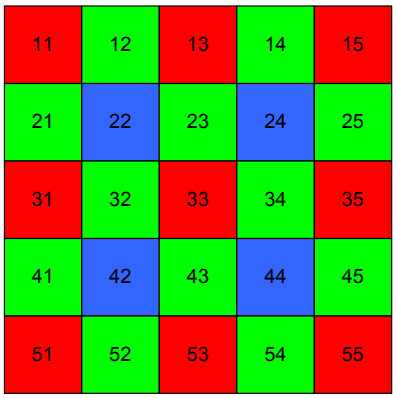


FIGURE 4.4 – Numérotation de la matrice de Bayer, la dénomination de chaque pixel dans l'image reconstruite est faite en utilisant la première lettre de la couleur avec comme indice sa position dans la matrice, par exemple le filtre rouge de la position $(1,1)$ sera noté R_{11}.

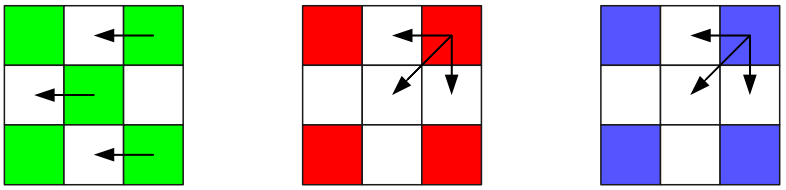


FIGURE 4.5 – Méthode de dématriçage par copies de pixels : chaque composante de couleur manquante est remplacée par une composante voisine déjà existante.

chaque plan de couleur. Cette interpolation peut être faite verticalement, horizontalement ou bilinéairement. Plusieurs lignes de l'image doivent être mémorisées pour pouvoir accomplir ce genre d'opérations. Le calcul est fait de la manière suivante, on utilise la numérotation de la figure 4.4, soit B_{33} la composante bleu manquante à la position (3,3), B_{23} la composante bleu manquante à la position (2,3), B_{32} la composante bleu manquante à la position (3,2) et V_{33} la composante verte manquante à la position (3,3) :

$$B_{33} = \frac{B_{22} + B_{24} + B_{42} + B_{44}}{4} \tag{4.1}$$

$$B_{23} = \frac{B_{22} + B_{24}}{2} \tag{4.2}$$

$$B_{32} = \frac{B_{22} + B_{42}}{2} \tag{4.3}$$

$$V_{33} = \frac{V_{23} + V_{32} + V_{34} + V_{43}}{4} \tag{4.4}$$

Le pas d'échantillonnage dans la matrice de Bayer est régulier. Il permet de calculé l'interpolation par une convolution avec un masque adapté. Par exemple, le masque M_v de la formule 4.5 permet de reconstruire les pixels verts manquants et le masque M_{rb} de la formule 4.6 permet de reconstruire les pixels rouges et bleus manquants. Un exemple de filtres utilisés pour l'interpolation des trois canaux en fonction de la configuration de couleur du pixel traité dans un masque de taille 5×5 est présenté sur la figure 4.6. La méthode d'interpolation bilinéaire est peu complexe, mais elle introduit du flou, des effets de moiré et des artéfacts de couleurs, comme on peut le voir sur la figure 4.7.

$$M_v = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{4.5}$$

$$M_{rb} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{4.6}$$

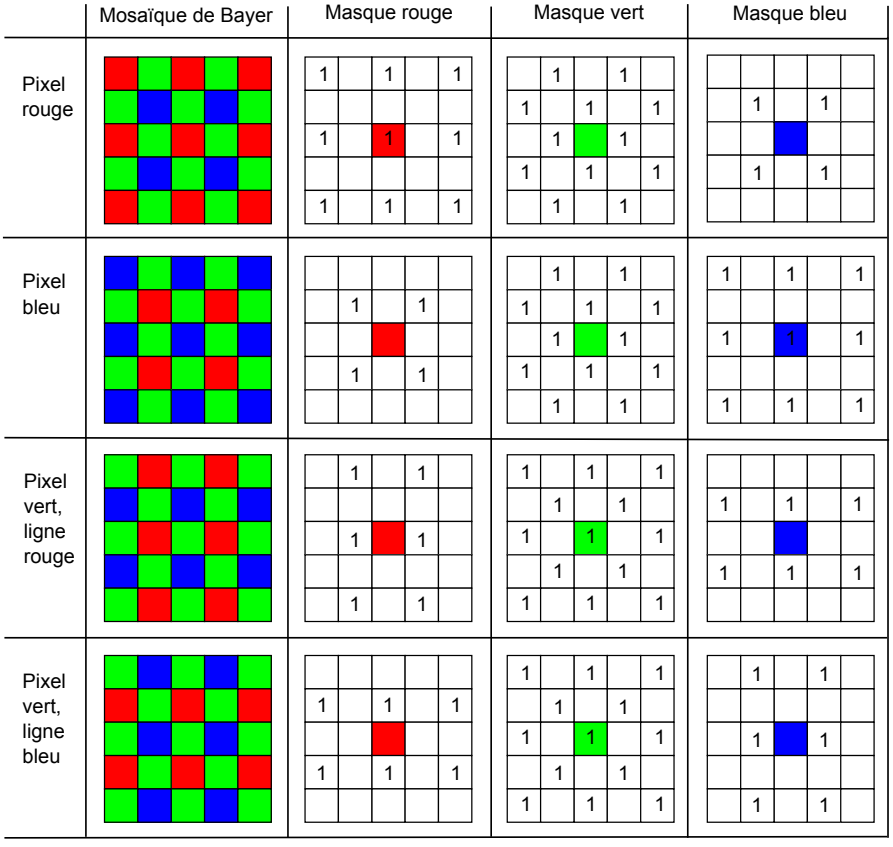


FIGURE 4.6 – Exemple des masques de taille 5×5 utilisés pour l'interpolation bilinéaire en fonction des différentes configurations de couleurs du pixel traité.

FIGURE 4.7 – Illustration des résultats obtenus avec l'interpolation par copies de pixels à gauche et l'interpolation bilinéaire à droite [2, 3].

4.2 Méthode d'interpolation par constance des teintes

Dans [51], Cok et al proposent une méthode d'interpolation par constante des teintes. En effet, ils proposent d'interpoler la teinte des pixels plutôt que leurs couleurs séparément. L'idée intuitive est que dans les images réelles, la teinte ne varie pas à la surface d'un objet. Cela revient à dire que pour un vecteur couleur (r, g, b) les rapports $\frac{r}{g}$, et $\frac{b}{g}$ varient lentement dans l'image. La méthode d'interpolation par constance des teintes est appliquée en quatre étapes :

1. on interpole le canal vert avec une méthode choisie (bilinéaire, interpolation directionnelle ...),
2. on calcul le rapport des plans des canaux $R_{rg} = \frac{r}{g}$ et $R_{bg} = \frac{b}{g}$ aux locations des pixels rouges et bleus existants, en utilisant le plan vert interpolé,
3. on interpole les deux plans de rapports R_{rg} et R_{bg} en utilisant l'interpolation bilinéaire,
4. on additionne le plan vert interpolé dans l'étape (1) à ces deux plans de différences pour construire les plans de couleurs rouge et bleu.

L'équation 4.7, illustre le calcul du point bleu à la position $(3, 3)$.

$$B_{33} = \frac{1}{4} V_{33} \left(\frac{B_{22}}{V_{22}} + \frac{B_{24}}{V_{24}} + \frac{B_{42}}{V_{42}} + \frac{B_{44}}{V_{44}} \right) \tag{4.7}$$

Cet algorithme permet de superposer les hautes fréquences spatiales du canal vert sur les canaux bleu et rouge, corrigeant ainsi les artéfacts de couleurs introduits par les sous-échantillonnages de ces canaux. La qualité de l'image produite dépend essentiellement

de la qualité d'interpolation du canal vert. De nombreux algorithmes on adopté cette méthode, on citera [9, 45–50]. Dans [50], Pei et al proposent d'interpoler la différence des plans de couleur $r - g$ et $b - g$ plutôt que leurs rapports. Ce choix est justifié par deux raisons : le coût des calculs, la soustraction étant exécutée de manière plus rapide que la division dans un processeur de calcul et par le fait que ce concept se rapproche de la représentation luminance/teinte/saturation, utilisée par exemple, dans le codage de télévision NTSC (National Television System Committee, en français : comité du système de télévision nationale). La figure 4.8 illustre la reconstruction des canaux rouge et bleu en utilisant cet algorithme.

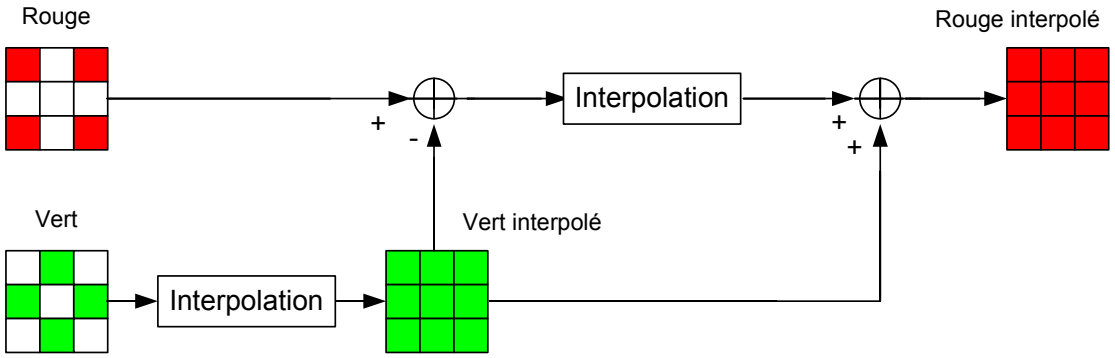


FIGURE 4.8 – Illustration de l'interpolation du canal rouge par la méthode de constance des teintes, en utilisant la différence des plans $r - g$, comme le propose Pei et al dans [50]. Le canal bleu est interpolé de la même manière.

4.3 Utilisation du laplacien comme terme de correction

Dans [8], Hamilton et Adams proposent d'exploiter la corrélation des informations existantes entre les variations spatiales des spectres de couleurs pour corriger les artefacts dûs au sous échantillonnage de l'information du canal vert. Cette hypothèse se traduit par le fait que dans une image réelle les gradients spatiaux sont achromatiques. Un gradient calculé dans le canal rouge a donc la même valeur qu'un gradient calculé dans le canal vert ou dans le canal bleu pour les mêmes coordonnées spatiales. Pour calculer la valeur de la composante verte d'un pixel pour une position de pixel rouge ou bleu dans la matrice de Bayer, le gradient calculé à cette position pour le canal rouge ou bleu sera ajouté pour corriger l'interpolation linéaire du pixel vert, ce calcul est illustré dans l'équation 4.8. Les plans rouge et bleu sont ensuite calculés par la méthode de constance des teintes. Dans [4], Malvar et al. généralisent ce résultat à l'interpolation bilinéaire du canal vert. Pour interpoler des canaux rouge et bleu, ils proposent les masques d'interpolations présentés sur la figure 4.9. Les coefficients des gradients dans les masques sont calculés par une approche de Wiener en déterminant les gains pour lesquels l'erreur quadratique moyenne est minimale sur un échantillon d'images de références.

$$G_{33} = \begin{cases} (G_{32} + G_{34})/2 + (2R_{33} - R_{31} - R_{35})/4 \\ \qquad si\ l'interpolation\ est\ horizontale \\ (G_{23} + G_{43})/2 + (2R_{33} - R_{13} - R_{53})/4 \\ \qquad si\ l'interpolation\ est\ verticale \end{cases} \tag{4.8}$$

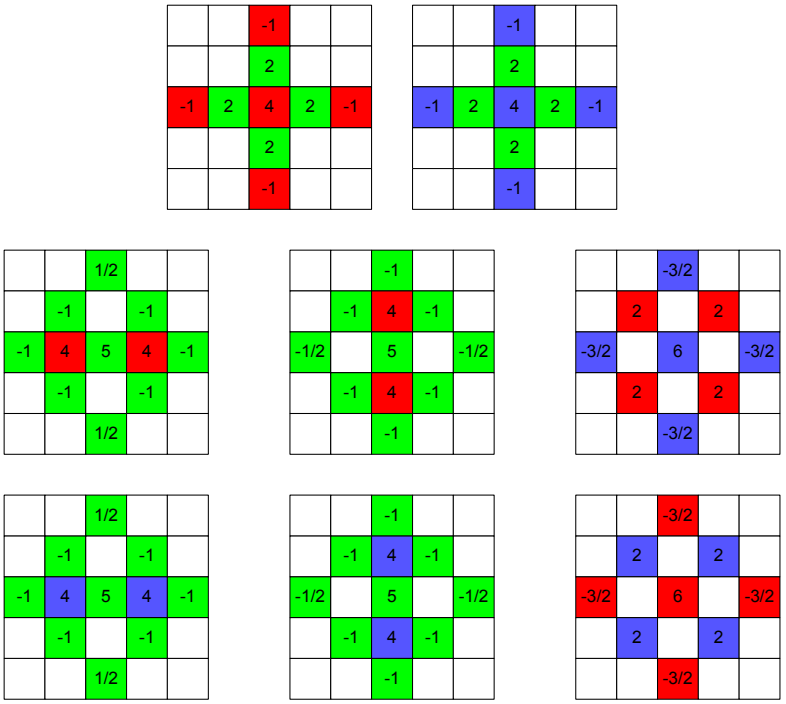


FIGURE 4.9 – Masques utilisés dans l'algorithme proposé par Malvar dans [4] pour chaque position de couleur dans la matrice de Bayer.

4.4 Interpolation à moyenne pondérée adaptative

Des méthodes d'interpolations adaptatives ont été proposées. Elles intègrent une pondération prenant en compte les structures des objets dans l'image. Par exemple, l'interpolation de la composante verte G_{33} est calculée comme dans l'équation 4.9 et l'interpolation de la composante bleu B_{33} est calculée comme dans l'équation 4.10.

$$G_{33} = \frac{E_{23}G_{23} + E_{32}G_{32} + E_{34}G_{34} + E_{43}G_{43}}{E_{23} + E_{32} + W_{34} + E_{43}} \tag{4.9}$$

$$B_{33} = \frac{E_{22}B_{22} + E_{24}G_{24} + E_{44}G_{44} + E_{42}G_{42}}{E_{22} + E_{24} + W_{44} + E_{42}} \tag{4.10}$$

où les coefficients $E_{i,j}$ représentent des poids calculés en fonction des variations d'intensités de l'image. Dans [49], Kimmel propose d'utiliser des poids basés sur un calcul de

gradient. La même méthode est proposée par R.Ramanath et al. dans [53]. Le calcul des poids est illustré dans l'équation 4.11.

$$E_{i+k,j+k} = \frac{1}{\sqrt{1+D(P_{i,j})^2+D(P_{i+k,j+l})^2}} \quad \text{avec} \quad k \in [-1;0;1] \quad \text{et} \quad l \in [-1;0;1] \tag{4.11}$$

Les valeurs des gradients D sont calculées comme dans les équations 4.12 à 4.15. Les gradients horizontaux et verticaux D_x et D_y sont utilisés pour l'interpolation du canal vert et des canaux bleu et rouge aux positions des filtres verts. Les gradients D_{xd} et D_{yd} sont utilisés pour l'interpolation des canaux rouge et bleu respectivement pour les positions des filtres bleu et rouge. Ce calcul de poids pour l'interpolation des composantes bleu et rouge est associé avec l'utilisation de l'algorithme de constance des teintes. On souligne que seuls les pixels d'une même couleur sont pondérés pour le calcul d'une composante. Une itération est utilisée pour améliorer la convergence et diffuser les artéfacts de couleurs.

$$D_x(p_{i,j}) = \frac{P_{i,j-1} - P_{i,j+1}}{2} \tag{4.12}$$

$$D_x(p_{i,j}) = \frac{P_{i-1,j} - P_{i+1,j}}{2} \tag{4.13}$$

$$D_{xd}(p_{i,j}) = \frac{P_{i-1,j+1} - P_{i+1,j-1}}{2\sqrt{2}} \tag{4.14}$$

$$D_{yd}(p_{i,j}) = \frac{P_{i-1,j-1} - P_{i+1,j+1}}{2\sqrt{2}} \tag{4.15}$$

4.5 Méthode par filtrage dans l'espace de Fourier

Dans [2, 54], Alleyson et al. présentent un algorithme de dématriçage par « sélection de fréquences ». Ils proposent d'écrire l'échantillonnage de l'image de la scène à travers la mosaïque de Bayer, comme la somme des produits respectifs des trois plans de couleurs en utilisant des fonctions d'échantillonnage m_i, tels que :

$$I_m(x,y) = \sum_{i \in R,V,B} C_i(x,y) m_i(x,y) \tag{4.16}$$

où la fonction $m_i(x, y)$ est égale à 1 si le filtre de couleur i est positionné sur le pixel (x, y) et égale à 0 dans le cas contraire. Ces fonctions sont mutuellement exclusives, car un seul filtre de couleur existe pour chaque position dans le plan échantillonné de l'image. Dans le cas de la matrice de Bayer, ces fonctions sont définies de la manière suivante :

$$\begin{cases} m_r(x,y) = \frac{1}{4}(1 + cos(\pi x))(1 + cos(\pi y)) \\ m_v(x,y) = \frac{1}{2}(1 - cos(\pi x)cos(\pi y)) \\ m_b(x,y) = \frac{1}{4}(1 - cos(\pi x))(1 - cos(\pi y)) \end{cases} \tag{4.17}$$

Chaque fonction m_i peut être écrite comme l'addition d'une information de luminance p_i et d'une information de chrominance $\tilde{m}_i(x, y)$ tels que :

$$m_i(x, y) = p_i + \tilde{m}_i(x, y) \tag{4.18}$$

L'image du capteur peut alors elle même s'écrire comme la somme d'une fonction de luminance $\phi(x, y)$ et d'une fonction de chrominance ψ_{CFA} tels que :

$$I_{CFA}(x, y) = \phi(x, y) + \psi_{CFA} \tag{4.19}$$

avec :

$$\phi(x, y) = \sum_{i \in R,V,B} p_i C_i(x, y) \tag{4.20}$$

et :

$$\psi_{CFA} = (x, y) \sum_{i \in R,V,B} C_i(x, y)\tilde{m}_i(x, y) \tag{4.21}$$

avec l'utilisation de ce modèle, on constate que seul le signal de chrominance est échantillonné. Le signal de luminance est présent sur chaque pixel. Alleyson et al. proposent de calculer la transformée de Fourier du modèle d'image de CFA présenté dans l'équation 4.21 pour connaître la représentation fréquentielle des composantes de luminances et de chrominances. Ils montrent ainsi que le dématriçage peut être fait en estimant la luminance et la chrominance par sélection de fréquences correspondantes dans le spectre de Fourier. Cette sélection peut être faite en utilisant par exemple le filtre défini dans l'équation 4.22 pour l'interpolation du plan vert considéré comme le plan de luminance.

Ce filtre a été construit en utilisant des fonctions gaussiennes dans le domaine fréquentiel, dont les écarts-types ont été choisis de manière à optimiser le PSNR (Peak Signal Noise Ratio) calculé à partir d'une base d'images de référence. Les chrominances sont ensuite interpolées par filtrage bilinéaire de la même manière que dans la méthode d'interpolation par constance des teintes. On trouvera dans la thèse de B.C de Lavarène [3] une description plus étendue des méthodes de dématriçage utilisant la transformation dans l'espace de Fourrier.

$$\frac{1}{128}\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & -5 & 3 & -9 & 3 & -5 & 1 & -1 & 0 \\ 1 & 0 & 2 & 3 & 1 & 7 & 1 & 3 & 2 & 0 & 1 \\ 0 & -2 & 1 & -9 & 7 & 104 & 7 & -9 & 1 & -2 & 0 \\ 1 & 0 & 2 & 3 & 1 & 7 & 1 & 3 & 2 & 0 & 1 \\ 0 & -1 & 1 & -5 & 3 & -9 & 3 & -5 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.22}$$

4.6 Méthodes de restauration de l'image

Dans [55] Gunturk et al. et dans [56] Glotzbach et al. proposent d'utiliser les corrélations existantes entre les variations spatiales des signaux de couleurs. De manière similaire à l'algorithme de constance des teintes, ils considèrent que la différence des canaux de couleurs produit un signal de basse fréquence spatiale. Une décomposition en ondelette est d'abord utilisée pour séparer les signaux de hautes fréquences spatiales des signaux de basses fréquences spatiales dans chaque canal de couleur. Les signaux de hautes fréquences spatiales du canal vert sont ensuite copiées sur les hautes fréquences spatiales des canaux rouge et bleu. Globzbach et al. dans [56] proposent ensuite d'estimer les recouvrements spatiaux des canaux bleu et rouge par modulation des hautes fréquences du canal vert et de les soustraire à ceux-ci. Dans [55], Gunturk et al. régularisent l'image obtenue par projections alternées sur des ensembles contraints. Les auteurs définissent deux ensembles contraints, l'espace des pixels existants originellement dans le CFA et l'espace du détail, qui force les hautes fréquences spatiales des canaux rouge et bleu à ressembler à celles du canal vert. Les projections se suivent successivement sur ces deux espaces contraints jusqu'à ce qu'un critère d'arrêt soit atteint.

4.7 Approche de formation de l'image

Les algorithmes de cette catégorie prennent en compte le processus de formation de l'image à travers la chaîne d'acquisition composée d'un système de lentilles et d'un capteur numérique. Le dématriçage est alors formulé comme un problème inverse consistant à modéliser les opérations de transformations tels que le filtrage des couleurs, les distorsions géométriques, les bruits de capteurs et déterminent l'image de la scène la plus réaliste étant donné l'image de Bayer obtenue. Dans [57, 58], Bainard modélise la transformation d'une image en couleurs formée de trois composantes colorées par pixel en une image composée d'une seule composante de couleur par pixel et d'une modélisation des bruits de capteurs. Taubmann dans [59], propose d'inclure le filtrage passe-bas introduit par la fonction d'étalement ponctuelle du système de lentilles. Ces méthodes basées sur des modèles statistiques complexes sont difficiles à implémenter et ne présentent pas de simulations dans les publications correspondantes. Nous ne nous intéressons pas à ces méthodes dans cette thèse.

4.8 Interpolation par reconnaissance de formes

Dans [5, 6], Cok propose un algorithme de dématriçage fonctionnant par reconnaissance de formes. Les voisinages dans un masque de taille 3×3 autour d'un point à interpoler sont classifiés en trois catégories : coin, bande, contour. Ces trois catégories sont illustrées sur la figure 4.10. Une interpolation différente est calculée suivant la classification du voisinage. Cok suppose que cette méthode permet de repousser les erreurs d'interpolations dans les zones texturées, là où elles apparaissent le moins visible. Pour interpoler le pixel vert X manquant, Cok propose d'appliquer la méthode suivante [2] :

1. on calcule la moyenne m des quatre plus proches voisins du pixel manquant (donc dans les directions horizontales et verticales), chacun de ces pixels est ensuite classé comme inférieur (L) ; supérieur (H) ou égal (E) à la moyenne. Les quatre pixels sont ensuite classés dans l'ordre décroissant et on calcule M, la médiane de ces quatre valeurs ;
2. on classifie ensuite le pixel à interpoler en coin, bande ou contour de la manière suivante :
   - si, parmi les quatre voisins du pixel à interpoler, on compte : 3 pixels H et 1 pixel L ou, 3 pixels L et 1 pixel H, alors, le voisinage est un contour ;
   - si, on compte : 2 pixels H et 2 pixels L adjacents deux à deux, alors, c'est un coin ;
   - si, on compte : 2 pixels H et 2 pixels L deux à deux, alors, c'est une bande ;

3. pour un contour la valeur du pixel considéré est $X = M$;
4. pour une bande, on calcul $x = 2M - S$, on en déduit la valeur du pixel considéré $X = CLIP_{\beta}^{\gamma}(x)$;
5. Pour un coin, on calcul $x = M - (S^{'} - M)/4$ et on en déduit la valeur du pixel considéré $X = CLIP_{\beta}^{\gamma}(x)$;

où S, est la moyenne des 8 pixels représentés sur la figure 4.11, S' est la moyenne des 4 quatre pixels représentés sur la figure 4.11 et la fonction $CLIP_{\beta}^{\gamma}(x)$ est définie dans l'équation suivante :

$$CLIP_{\beta}^{\gamma}(x) = \begin{cases} x & si\ \gamma \leq x \leq \beta \\ \beta & si\ \beta \leq x \\ \gamma & si\ \leq \gamma \end{cases} \quad (4.23)$$

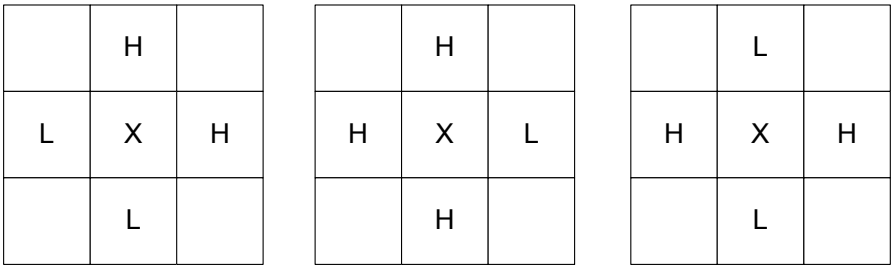


FIGURE 4.10 – Classification des formes dans l'algorithme de Cok [5, 6], de gauche à droite : coin, bande et contour. X est le pixel traité, H désigne une valeur de pixel supérieure à la moyenne des quatre voisins, L désigne une valeur de pixel inférieure à la moyenne des quatre voisins.

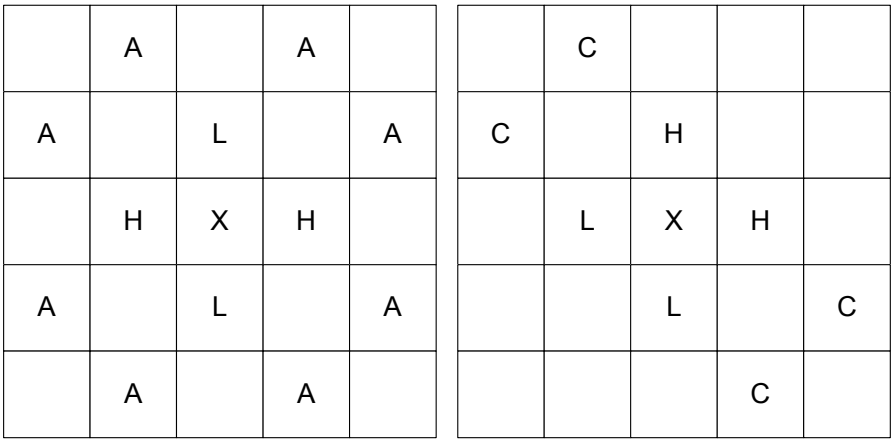


FIGURE 4.11 – Sur le masque de gauche, on représente et on nomme A, les 8 pixels utilisés dans le calcul de la moyenne S dans le cas d'une bande, sur le masque de droite, on représente et on nomme C, les 4 pixels utilisés dans le calcul de la moyenne S' dans le cas d'un coin.

Cet algorithme est utilisé seulement pour les pixels verts, les pixels rouges et bleus sont interpolés par la méthode de constance des teintes. L'effort de classification proposé par

Cok s'avère couteux en nombre de calculs et de tests, de plus, les images dématriçées avec cet algorithme contiennent de nombreux artéfacts de couleurs.

4.9 Interpolations directionnelles

Les algorithmes de dématriçage par interpolations directionnelles consistent à interpoler les pixels manquants en suivant la direction locale des détails dans l'image. Dans ces méthodes, le plan vert est reconstruit en premier, les plans de couleurs rouge et bleu sont ensuite construits en utilisant la méthode de constance des teintes. De manière générale, Hirakawa dans [18] met en avant le fait que les combinaisons d'interpolations horizontales et verticales sont suffisantes pour produire une image de très bonne qualité (voir figure 4.12). Les choix de directions d'interpolations sont opérés avec des estimateurs. Différentes méthodes d'estimations des directions sont proposées dans la littérature. Dans [47], Hibbard calcule les gradients des pixels verts dans les directions verticales et horizontales. Par exemple, pour interpoler le pixel vert à la position $(2, 2)$ dans la figure 4.4, Hibbard propose de procéder de la manière suivante :

1. on calcule les gradients verticaux et horizontaux :
   $\Delta H = |G_{21} - G_{23}|$ et
   $\Delta V = |G_{12} - G_{32}|$
2. si $\Delta H > \Delta V$, alors on interpole verticalement, soit :
   $G_{22} = \frac{G_{12}+G_{32}}{2}$
   sinon si $\Delta V > \Delta H$ alors on interpole horizontalement, soit :
   $G_{22} = \frac{G_{21}+G_{23}}{2}$
   sinon on interpole bilinéairement, soit :
   $G_{22} = \frac{G_{21}+G_{23}+G_{12}+G_{32}}{4}$

Dans [9], Laroche et al. utilisent le calcul du laplacien des pixels rouges et bleus pour augmenter la précision de l'estimateur. Hamilton et al. dans [8] proposent de fusionner la méthode de Hibbard et Laroche et al. en calculant à la fois le gradient du premier ordre du canal vert et le laplacien du canal bleu ou rouge. Après avoir fait le choix de la direction d'interpolation, les pixels verts sont calculés en utilisant la méthode de correction utilisant le laplacien décrite dans la partie 4.3. Par exemple, pour interpoler la composante verte G_{33}, l'estimateur proposé par Hamilton fonctionne de la manière suivante :

1. on calcule les gradients verticaux et horizontaux corrigés par le laplacien du canal bleu ou rouge (suivant la configuration de couleur du pixel traité) :

$\Delta H = |G_{32} - G_{34}| + |2R_{33} - R31 - R_{35}|$ et
$\Delta V = |G_{23} - G_{43}| + |2R_{33} - R13 - R_{53}|$

2. si $\Delta H > \Delta V$, alors on interpole verticalement, soit :
   $G_{33} = \frac{G_{23}+G_{43}}{2} + \frac{2R_{33}-R13-R_{53}}{4}$
   sinon si $\Delta V > \Delta H$ alors on interpole horizontalement, soit :
   $G_{33} = \frac{G_{32}+G_{34}}{2} + \frac{2R_{33}-R31-R_{35}}{4}$
   sinon on interpole bilinéairement, soit :
   $G_{33} = \frac{G_{32}+G_{34}+G_{23}+G_{43}}{4} + \frac{4R_{33}-R31-R_{35}-R13-R_{53}}{8}$

Dans [7], Hirakawa et al. proposent un estimateur basé sur des calculs d'homogénéités des informations de luminances et de chrominances dans l'espace CIELab. Ces calculs sont fait localement sur les images en couleurs interpolées verticalement et horizontalement. Les pixels verts sont interpolés en utilisant la méthode décrite dans la partie 4.3, de la même manière que dans l'algorithme proposé par Hamilton et al. dans [8]. Les canaux bleu et rouge sont obtenus par la méthode de constance des teintes. Ils supposent qu'une région interpolée dans la mauvaise direction est moins homogène qu'une région interpolée dans la bonne direction. Un exemple de dématriçage par la méthode de Hirakawa est montré sur la figure 4.12.

La méthode d'interpolation directionnelle est une méthode de reconstruction simple, qui, lorsqu'elle est utilisée avec un estimateur de direction des détails idéal permet de produire la meilleure qualité d'image dématriçée parmi les algorithmes de la littérature. Le choix de l'estimateur à une influence directe sur la qualité de l'image produite et sur la complexité de l'algorithme. Cette dualité est intéressante, elle permet directement de choisir un estimateur, en fonction des contraintes de qualités et/ou de complexités algorithmiques exigées.

4.10 Réduction des artéfacts de couleurs

Même avec l'utilisation d'un algorithme de dématriçage optimal, l'étape de reconstruction des canaux vert, bleu et rouge introduit des artefacts de couleurs qui sont inhérents à l'interpolation de la mosaïque de Bayer. Ces artefacts de couleurs apparaissent lorsque la résolution des détails de l'image est proche de la résolution du capteur. Ils sont dûs au non-respect du théorème d'échantillonnage de Shannon-Nyquist. En effet, la fréquence d'échantillonnage d'un signal continu doit être au moins le double de la fréquence maximale du signal échantillonné pour éviter les effets de moiré, introduisant eux mêmes des artéfacts de couleurs dans le cas du dématriçage. Une méthode de réduction des artéfacts de couleurs est proposée dans [12]. Celle-ci consiste à diminuer les variations

(a) (b) (c)

FIGURE 4.12 – Illustration des résultats obtenus avec l'algorithme de dématriçage proposé par Hirakawa et al. dans [7]. (a) image interpolée horizontalement, (b) image interpolée verticalement, (c) image mixée d'interpolation horizontales et verticales dans laquelle la direction d'interpolation est fait en utilisant le critère de Hirakawa [18].

des couleurs à la surface d'un objet en appliquant un filtre médian sur la différence des plans vert/rouge et vert/bleu. La justification est la même que pour l'algorithme de constance des teintes (voir section 4.2), c'est à dire que la teinte à la surface d'un objet varie lentement. L'algorithme est appliqué de la manière suivante :

$$\begin{aligned} R' &= m(R-G)+G \\ B' &= m(B-G)+G \\ G' &= \frac{[m(G-R')+R']+[m(G-B')+B']}{2} \end{aligned} \tag{4.24}$$

où, R, G et B, représentent respectivement les canaux de couleurs rouge, vert et bleu de l'image en couleurs dématriçée. R', G' et B', représentent respectivement les canaux de couleurs rouge, vert et bleu filtrés et m est le filtre médian appliqué dans un masque carré de taille $M \times M$.

4.11 Évaluation de la complexité et de la qualité des images produites par les algorithmes

Pour évaluer et comparer la complexité des différents algorithmes, nous calculons le nombre d'opérations de multiplications, d'additions, de comparaisons et de valeurs absolues nécessaires à la construction du vecteur couleur (r, g, b) d'un pixel de l'image.

Le tableau 4.1 montre la complexité des filtres étudiés. L'estimation de la qualité des images est un problème récurrent dans le domaine du traitement d'image. Il est nécessaire d'avoir à disposition des outils de mesures, afin d'évaluer et de comparer, la qualité des images produites par les différents algorithmes. Il est généralement admis que les mesures de qualités d'images existantes ne peuvent pas efficacement quantifier la qualité de perception psycho-visuelle obtenu après la restoration d'une image ou sa reconstruction [60–62]. Cependant une quantification globale de la qualité des images peut être obtenue par la combinaison de l'appréciation visuelle avec la mesure de l'erreur quadratique moyenne RMS (Root Mean Squarre) entre l'image d'origine représentant la scène et l'image reconstruite. Le calcul du RMS est présenté dans l'équation 4.25, où $u(.)$ est l'image de la scène, $\widehat{u}(.)$ est l'image estimée, N est le nombre de pixels dans l'image et x est la coordonnée $2D$ d'un pixel dans le plan de l'image. La mesure du RMS est faite sur les 24 images de la base de données Kodak PhotoCD (voir annexe D). Ces images sont échantillonnées spatialement en respectant l'arrangement des couleurs de la mosaïque du filtre de Bayer. On modélise ainsi l'échantillonnage de l'image de la scène lorsque celle-ci est projetée par le système de lentilles sur la surface photosensible du capteur. Pour améliorer la précision des mesures, l'erreur quadratique moyenne est mesurée séparément dans les zones de contours et dans les zones plates comme le proposent Lu et al. dans [63]. Nous mesurons ensuite le RMS sur chaque plan de couleur séparément comme le propose Gunturk et al. dans [44]. Les résultats de ces calculs sont présentés dans les tableaux 4.2 et 4.3. Pour une meilleure visualisation des résultats, la figure 4.13 montre l'histogramme du tableau 4.3. On évalue aussi la qualité des images reconstruites de manière subjective par appréciation visuelle. La figure 4.14 montre les résultats de la qualité d'image produite par les différents algorithmes de dématriçage sur une image de référence, contenant un motif de bandes verticales.

$$RMS = \sqrt{\frac{1}{N}\sum_{N}[u(x) - \widehat{u}(x)]^2} \tag{4.25}$$

Dans le tableau 4.1, on peut voir que l'algorithme proposé par Gunturk [55] possède la complexité la plus importante avec 480 multiplications et 480 additions. Ensuite l'algorithme proposé par Kimmel [49] nécessite 120 multiplications et 180 additions. L'algorithme proposé par Hirakawa [18] est moins complexe, avec 48 additions, 88 multiplications, 71 comparaisons et 12 valeurs absolues. Ensuite, vient l'algorithme proposé par Alleyson [54], avec 61 multiplications et 70 additions, puis l'algorithme proposé par Malvar [4], avec 19 multiplications et 21 additions qui est plus complexe que l'algorithme proposé par Hamilton [8], nécessitant en moyenne 13/2 multiplications, 14 additions, 1

comparaison et 4 valeurs absolues, suivi par l'algorithme de Hibbard [47] avec 6 multiplications, 12 additions, 1 comparaison et 2 valeurs absolues, enfin l'algorithme le moins complexe est l'interpolation bilinéaire avec 4 multiplications et 3 additions.

TABLE 4.1 – Complexité des algorithmes de dématriçage étudiés

	multiplications	additions	comparaisons	valeurs absolues
Bilinéaire 3x3	4	3		
Malvar [4]	19	21		
Alleyson [54]	61	70		
Kimmel [49]	120	180		
Hibbard [47]	6	12	1	2
Hamilton [8]	13/2	14	1	4
Hirakawa [18]	48	88	71	12
Gunturk [55]	480	480		

A partir des résultats des calculs du RMS présentés dans le tableau 4.2, on peut voir que l'algorithme de Gunturk donne les meilleurs résultats de qualité d'image objective, aussi bien dans les régions de contours que dans les régions plates. Il est suivi par les algorithmes de Hirakawa, Hamilton, Hibbard, Kimmel, Alleyson et Malvar. L'algorithme d'interpolation bilinéaire produit les moins bons résultats. On peut remarquer que les résultats du calcul du RMS varient de manière beaucoup plus importante dans les zones de contours que dans les zones plates. Ce résultat est conforté par le fait que les images sont beaucoup plus sensibles à l'apparition d'artéfacts dans les zones de contours que dans les zones plates. Dans le tableau 4.3 on peut voir que de manière générale, la classification des résultats du calcul du RMS dans les trois plans de couleurs est la même que dans le tableau 4.2. On remarque, que pour chaque algorithme, le plan vert possède toujours un résultat de RMS plus faible. En effet, l'information de couleur verte possède une fréquence d'échantillonnage deux fois plus importante que l'information des canaux de couleurs rouge et bleu, réduisant ainsi de 50% la probabilité d'introduire des erreurs d'interpolations.

TABLE 4.2 – Moyenne des calculs du RMS à travers 24 images de références dans les régions de contours, les régions plates et sur l'image entière

algorithme	régions de contours	régions plates	image entière
bilinéaire 3x3	168,06	23,07	47,86
Malvar [4]	56,70	8,23	17,97
Alleyson [54]	58,34	8,43	19,30
Kimmel [49]	48,48	8,56	16,37
Hibbard [47]	44,66	7,55	14,74
Hamilton [8]	38,21	6,95	13,06
Hirakawa [18]	31,53	5,90	11,34
Gunturk [55]	25,26	4,42	8,84

TABLE 4.3 – Moyenne des calculs du RMS à travers 24 images de références dans les plans rouge, vert et bleu

algorithme	canal rouge	canal vert	canal bleu
bilinéaire 3x3	44,25	52,23	47,11
Malvar [4]	20,36	8,38	25,17
Alleyson [54]	19,36	16,38	22,17
Kimmel [49]	17,52	12,53	19,05
Hibbard [47]	14	11,53	16,87
Hamilton [8]	12,94	10,53	15,70
Hirakawa [18]	11,96	8,07	14
Gunturk [55]	9,41	5,69	11,41

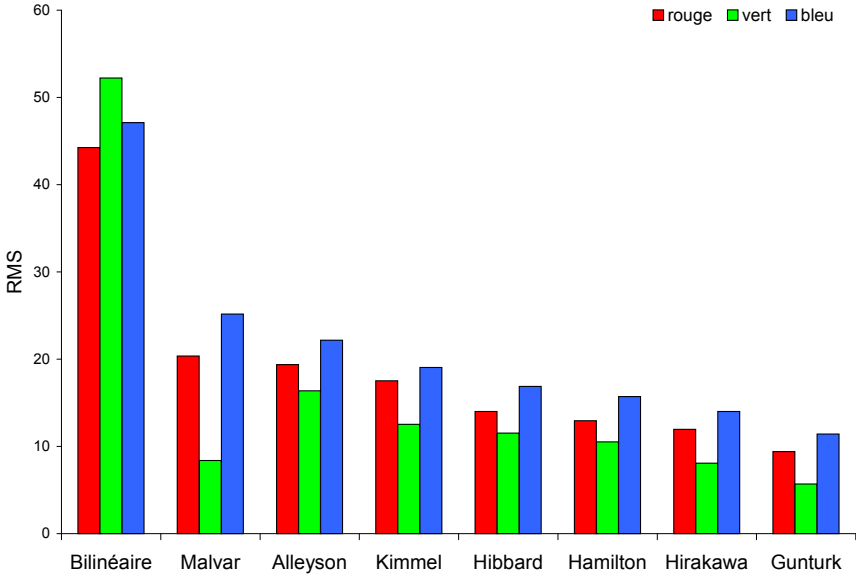


FIGURE 4.13 – Histogramme de la moyenne des calculs du RMS à travers 24 images de références dans les canaux rouge, vert et bleu, obtenu à partir des données du tableau 4.3.

La figure 4.14 montre les résultats visuels produits par les algorithmes de dématriçage étudiés sur une image test de référence issue de la base d'images Kodak PhotoCD (annexe D). Cette image contient des détails verticaux dont les variations horizontales sont proches de la résolution de l'image. Elle est donc propice à l'apparition d'artéfacts d'interpolations. Pour estimer la qualité des images produites, on évalue la présence des artéfacts décrits au début de ce chapitre et présentés sur la figure 4.1, le flou, les effets de moiré, les artéfacts de couleurs, la création de structures en formes de labyrinthes et l'apparition d'un effet de grille. Sur la figure 4.14, on peut voir que l'algorithme de Hirakawa produit la meilleure qualité d'image. En effet, l'image produite ne possède quasiment aucuns artéfacts. La deuxième meilleure qualité d'image est produite par l'algorithme de Kimmel, suivi par l'algorithme de Gunturk. Ces deux algorithmes font apparaître des effets de moiré importants, des artéfacts de couleurs et du flou. Ensuite, les algorithmes proposés par Hamilton et Hibbard font apparaître des structures en formes

de labyrinthes. Enfin, les algorithmes proposés par Alleyson, Malvar et l'interpolation bilinéaire font apparaître des effets de moiré importants, des artéfacts de couleurs et du flou. On peut noter que l'évaluation subjective des résultats par l'appréciation visuelle est différente de l'évaluation des résultats obtenus avec le calcul du RMS. Il est donc très important de prendre en compte ces deux paramètres lorsque l'on veut estimer la qualité des images produites par un algorithme, en favorisant toujours l'aspect visuel.

Les résultats obtenus montrent clairement que les algorithmes de Gunturk, Hirakawa, Hamilton et Kimmel produisent les meilleures qualités d'images. Gunturk produit la meilleure qualité d'image objective estimée avec la mesure du RMS, suivi par l'algorithme de Hirakawa, Hamilton et Kimmel. La qualité d'image rendu par Hirakawa est subjectivement la meilleure par appréciation visuelle, suivie par l'algorithme de Kimmel, Gunturk et Hamilton. Les algorithmes de Gunturk et Kimmel possèdent des complexités importantes comparées à l'algorithme de Hirakawa, possédant lui même une complexité importante par rapport aux algorithmes de Hamilton et Hibbard. Finalement, l'algorithme proposé par Hirakawa et l'algorithme proposé par Hamilton fournissent un bon compromis entre complexité algorithmique et qualité d'image produite. On note aussi que ces deux algorithmes fonctionnent par interpolations directionnelles.

(a) Image originale (b) Interpolation bilinéaire (c) Alleyson [54]

(d) Malvar [4] (e) Hibbard [47] (f) Hamilton [8]

(g) Kimmel [49] (h) Gunturk [55] (i) Hirakawa [18]

FIGURE 4.14 – Comparaison des qualités visuelles des différents algorithmes de dématriçage étudiés.

4.12 Résumé

Dans un système de camera numérique, le capteur est seulement capable de mesurer une composante de couleur par photosite. Une étape de dématriçage est nécessaire pour produire une image en couleurs visualisable. En construisant les deux tiers de l'image, le dématriçage a une importance fondamentale sur la qualité d'image produite. En effet, elle peut introduire de nombreux artéfacts (voir figure 4.1). Les différents algorithmes de la littérature produisent des qualités d'image variées et possèdent des complexités algorithmiques différentes. Parmi les algorithmes étudiés, nous avons vu que les algorithmes proposés par Gunturk et al. [55], Hirakawa et al. [18], Hamilton et al. [8] et Kimmel [49], produisent les meilleurs qualités d'images en termes de RMS et d'appréciation visuelle. L'étude des complexités algorithmiques montre que les propositions de Gunturk et Kimmel sont très complexes par rapport aux solutions proposés par Hirakawa et Hamilton. Finalement l'algorithme de Hamilton et l'algorithme de Hirakawa montrent de bons compromis entre qualités d'images produites et complexité algorithmique. Ces deux algorithmes fonctionnent par interpolations directionnelles. Le seul point qui les différencie est le calcul de classification des directions d'interpolations. Ce calcul est primordial, il influe à la fois sur la complexité algorithmique et sur la qualité des images produites. En effet, Hirakawa propose un calcul complexe privilégiant la qualité de l'image. Inversement Hamilton propose un calcul peu complexe privilégiant la rapidité d'exécution. Cette constatation nous amène à la question suivante : peut-on proposer un estimateur peu complexe permettant de produire une image sans artéfacts ?

Chapitre 5

Dématriçage : proposition d'un nouvel algorithme, GEDI (Green Edge Directed Interpolation)

Hamilton et al. dans [8] et Hirakawa et al. dans [18] proposent deux algorithmes de dématriçage par interpolations directionnelles possédant respectivement des qualités de faible complexité de calculs et d'image produite. La seule différence existante entre ces deux algorithmes est la manière d'estimer la direction des détails pour interpoler un pixel horizontalement ou verticalement. Hamilton et al. proposent un estimateur calculant directement un gradient sur la matrice de Bayer. Cet estimateur possède une faible complexité algorithmique (voir tableau 4.1) mais génère de faux choix de directions d'interpolations, introduisant des artéfacts de couleurs et des structures en formes de labyrinthes (voir figure 4.14). L'estimateur proposé dans [18] par Hirakawa et al. est basé sur des calculs d'homogénéités de luminance et de chrominance sur les images en couleurs interpolées horizontalement et verticalement (voir section 4.9). Cet estimateur permet de minimiser la présence d'artéfacts et produit une très bonne qualité d'image (voir figure 4.14). Cependant, les calculs des images en couleurs horizontale et verticale associés au calculs d'homogénéités introduisent une très grande complexité de algorithmique (voir tableau 4.1). Dans ce chapitre nous nous inspirons de ces deux algorithmes pour proposer un nouvel estimateur performant et peu complexe en utilisant les informations des gradients des objets dans les plans verts interpolés verticalement et horizontalement.

5.1 Proposition d'un nouvel estimateur : GED (Green Edge Direction)

Dans la mosaïque de Bayer, les filtres verts sont deux fois plus nombreux que les filtres bleus et rouges. C'est donc dans le canal vert que l'information de structure des objets est la plus importante. Pour estimer la direction d'interpolation, nous choisissons donc de travailler essentiellement sur le canal vert. Particulièrement, nous allons utiliser l'information des pixels verts interpolés verticalement et horizontalement. On suppose qu'un objet de direction verticale, interpolé dans la direction verticale possède une valeur de gradient vertical plus faible que la valeur du gradient vertical de ce même objet interpolé horizontalement. De la même manière, un objet de direction horizontale, interpolé dans la direction horizontale possède une valeur de gradient horizontal plus faible, que la valeur du gradient horizontal de ce même objet interpolé verticalement. D'autre part, on estime que dans une image réelle les directions des gradients varient lentement. En s'appuyant sur ces deux hypothèses, nous proposons un nouvel estimateur basé sur une comparaison des gradients des plans verts interpolés dans les directions verticales et horizontales et une homogénéisation locale des choix de directions des objets.

5.1.1 Estimation de la direction des objets

Dans cette section, nous proposons un nouvel estimateur de directions des objets. Considérons $G_v(.)$ le plan vert interpolé verticalement et $G_h(.)$ le plan vert interpolé horizontalement. Considérons $(i,j) \in X$ où X est un ensemble de positions à deux dimensions dans le plan de la matrice de Bayer et $G(.)$ est le plan vert reconstruit. L'estimateur proposé est exprimé dans l'équation 5.1, où $\nabla_h G(i,j)$ et $\nabla_v G(i,j)$ sont respectivement définis dans les équations 5.4 et 5.3.

$$G(i,j) = \begin{cases} G_h(i,j) & si \ [\nabla_h G_h(i,j) + \nabla_h G_v(i,j)] \\ & \quad \leq [\nabla_v G_h(i,j) + \nabla_v G_v(i,j)], \\ G_v(i,j) & sinon \end{cases} \tag{5.1}$$

$$\nabla_h G_h(i,j) = |G(i-1,j) - G(i,j)| + |G(i+1,j) - G(i,j)| \tag{5.2}$$

$$\nabla_v G_h(i,j) = |G(i,j-1) - G(i,j)| + |G(i,j+1) - G(i,j)| \tag{5.3}$$

5.1.2 Fonctionnement de l'estimateur

Pour expliquer le fonctionnement de l'estimateur, nous utilisons les schémas des figures 5.1, 5.2, 5.3. La figure 5.1(a), montre le plan vert d'un motif vertical de l'image d'une scène. La figure 5.1(b) montre l'échantillonnage de ce motif par les filtres verts de la mosaïque de Bayer, lorsque le pixel dont on veut estimer la direction du détail auquel il appartient est situé sur un filtre rouge ou bleu (c'est alors un trou), sinon, lorsque le pixel est placé sur un filtre vert c'est un point. Les figures 5.2 et 5.3 montrent respectivement les cas des interpolations verticales et horizontales dans le cas d'un trou et dans le cas d'un point. En utilisant les notations des figures 5.2(a), 5.2(b), 5.3(a) et 5.3(b), on peut respectivement écrire les expressions de $\nabla_h(.)$ et $\nabla_v(.)$, tels que :

$$\nabla_h G_1 = |G_2 - G_1| + |G_4 - G_1| \tag{5.4}$$

et

$$\nabla_v G_1 = |G_3 - G_1| + |G_5 - G_1| \tag{5.5}$$

où G_i représente un point du plan vert G à la position i. Dans les cas présentés, il s'agit d'estimer la direction d'interpolation du point à l'emplacement d'indice 1, aussi bien dans le cas d'un trou (filtre non vert) que dans le cas d'un point (pixel vert déjà existant). Pour ces deux cas, lorsque le point est interpolé verticalement, on vérifie que :

$$\nabla_v G_v(i,j) < \nabla_h G_v(i,j) \tag{5.6}$$

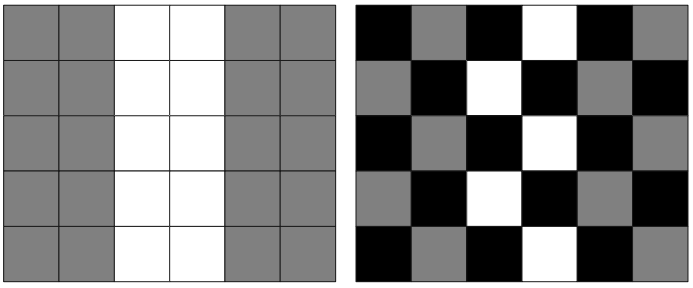
De la même manière, pour les résultats des deux interpolations horizontales (figures 5.2(b) et 5.2(a)) on peut clairement estimer que :

$$\nabla_h G_h(i,j) \approx \nabla_v G_h(i,j) \tag{5.7}$$

Nous justifierons le calcul du choix de la direction d'interpolation pour un pixel vert déjà existant dans la section 5.1.3. En utilisant les conditions de l'estimateur dans l'équation 5.1 et les relations des équations 5.6 et 5.7, on peut alors déduire que la direction d'interpolation est verticale. On peut vérifier de la même façon que l'estimateur fonctionne dans le cas d'un détails possédant une direction horizontale. De manière générale, on peut déduire les relations 5.8 et 5.9, respectivement lorsque le pixel de coordonnées (i,j) appartient à un détail vertical et lorsque le pixel appartient à un détail horizontal :

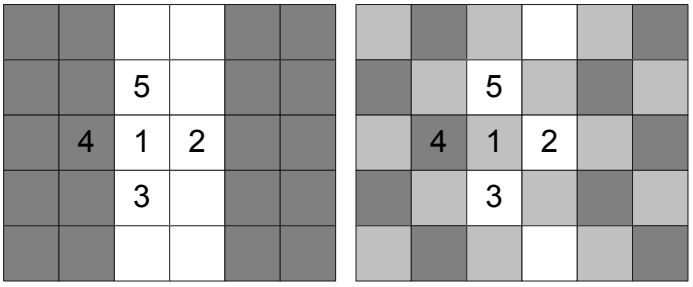
$$\{\nabla_h G_h(i,j) \approx \nabla_v G_h(i,j);\ \ \nabla_v G_v(i,j) < \nabla_h G_v(i,j)\} \tag{5.8}$$

$$\{\nabla_h G_h(i,j) < \nabla_v G_h(i,j);\ \ \nabla_v G_v(i,j) \approx \nabla_h G_v(i,j)\} \tag{5.9}$$



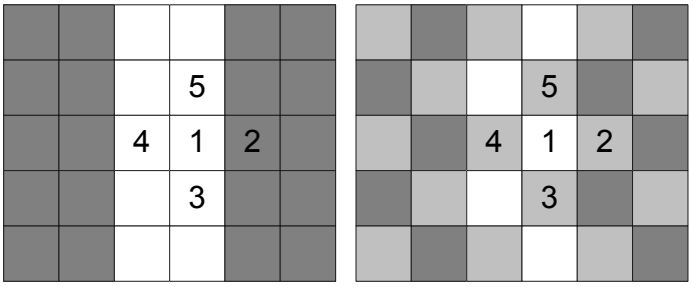
(a) motif du plan vert d'origine. (b) échantillonnage de Bayer.

FIGURE 5.1 – Détail vertical dans le plan vert et sont échantillonnage de Bayer.



(a) interpolation verticale dans le cas d'un trou. (b) interpolation horizontale dans le cas d'un trou.

FIGURE 5.2 – Estimation de la direction du motif dans le cas d'un trou.



(a) interpolation verticale dans le cas d'un point. (b) interpolation horizontale dans le cas d'un point.

FIGURE 5.3 – Estimation de la direction du motif dans le cas d'un point.

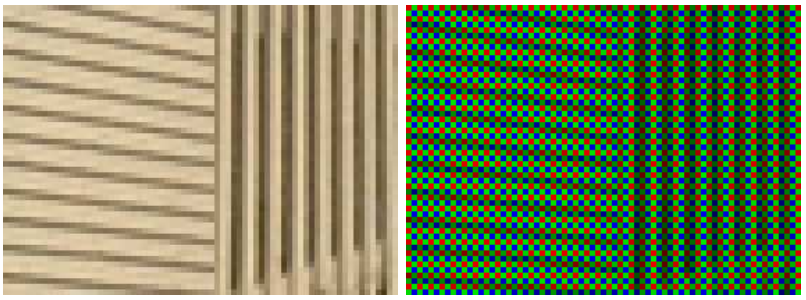
Illustrons maintenant le fonctionnement de cet estimateur sur une image structurée. L'image de la figure 5.4(a) à été construite spécialement pour contenir des objets possédants à la fois des détails verticaux et des détails horizontaux. Considérons $H(.)$ les zones de détails horizontaux et $V(.)$, les zones de détails verticaux. La figure 5.4(b) montre

l'échantillonnage de cette image par la mosaïque de Bayer. Les figures 5.4(c) et 5.4(d) présentent respectivement les résultats des interpolations horizontales du plan vert et de l'image en couleurs correspondante après reconstruction des canaux rouge et bleu par la méthode de constance des teintes. Sur l'image de la figure 5.4(c), on peut voir que l'interpolation horizontale renforce les détails horizontaux, soit $\nabla_h G_h(i,j) < \nabla_v G_h(i,j)$, pour un pixel de coordonnées $(i,j) \in H(.)$. Les détails verticaux sont dégradés, conformément à l'hypothèse émise au début de ce chapitre, soit $\nabla_h G_v(i,j) \approx \nabla_v G_v(i,j)$, pour un pixel de coordonnées $(i,j) \in H(.)$. Sur l'image en couleurs (figure 5.4(d)), on peut voir les artéfacts de couleurs introduits par la dégradation des détails verticaux. Les figures 5.4(e) et 5.4(f) montrent respectivement les résultats des interpolations verticales du plan vert et de l'image correspondante en couleurs. Inversement au cas de l'interpolation horizontale, les détails verticaux sont renforcés, soit $\nabla_v G_v(i,j) < \nabla_h G_v(i,j)$, pour un pixel de coordonnées $(i,j) \in V(.)$. Tandis que les détails horizontaux sont dégradés, soit $\nabla_h G_h(i,j) \approx \nabla_v G_h(i,j)$, pour un pixel de coordonnées $(i,j) \in V(.)$, faisant ainsi apparaître des artefacts de couleurs. En appliquant ces résultats à l'équation 5.1 de l'estimateur proposé, on peut voir que les bonnes directions d'interpolations sont choisies. L'image de la figure 5.4(h) présente le résultat du dématriçage de l'image de la figure 5.4(a) en utilisant l'estimateur.

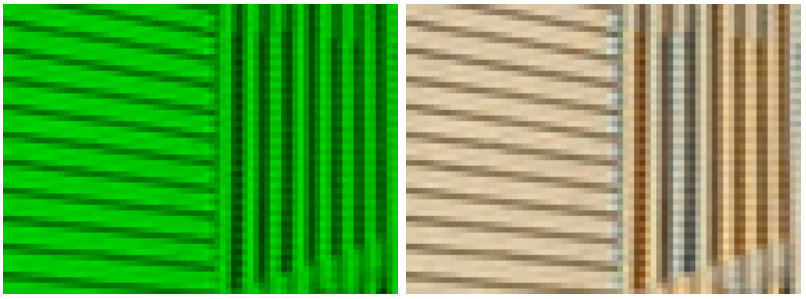
5.1.3 Correction par LMDC (Local Majority Direction Choice)

5.1.3.1 Application à l'estimateur proposé

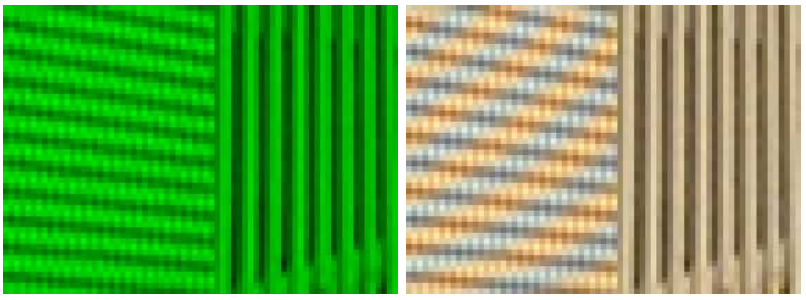
Lors de l'estimation de la direction d'interpolation, le manque de précision de l'estimateur et/ou le manque d'informations dans l'image peuvent introduire des erreurs. Ces erreurs de directions dégradent la qualité de l'image en faisant apparaître des artéfacts de couleurs et des structures en formes de labyrinthes (voir section 4.1). Pour minimiser ces erreurs, nous proposons de corriger le choix de l'estimateur par le choix de direction majoritaire dans une zone locale centrée sur le pixel courant. On appel cette méthode LMDC pour Local Majority Direction Choice (choix de direction majoritaire local). On justifie son utilisation par le fait que dans les images réelles, les directions des gradients des objets évoluent de façon continue. Si un choix de direction d'interpolation est isolé, alors il est fortement probable qu'il soit erroné. Il faut donc le corriger et l'adapter à la direction majoritaire locale. On minimise ainsi la présence d'artefacts de couleurs dans l'image dématriçée. Les éléments structurants utilisés pour le calcul du LMDC peuvent être différents : croix, losange, carré, rond. C'est dans cette étape de l'algorithme que les directions d'interpolations calculées pour les pixels verts déjà existants sont utilisées. Elles vont contribuer à la correction des erreurs d'estimations. On procède de la manière



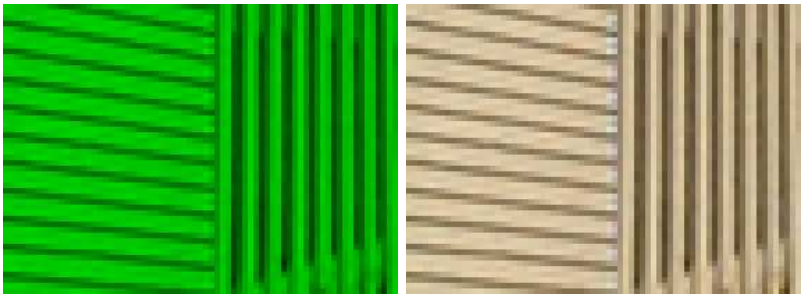
(a) image originale (b) échantillonnage de Bayer



(c) interpolation horizontale du plan vert (G_h) (d) image en couleurs de l'interpolation horizontale



(e) Interpolation verticale (G_v) (f) image en couleurs de l'interpolation verticale



(g) utilisation de l'estimateur GEDI (h) utilisation du nouvel estimateur

FIGURE 5.4 – Illustration du fonctionnement de l'estimateur proposé.

suivante : si le choix de direction d'interpolation du point traité est différent du LMDC, alors il est remplacé par le LMDC, dans le cas contraire, si le choix de direction est égal au LMDC, il n'est pas remplacé. La figure 5.5 illustre ce fonctionnement. On utilise cette méthode de façon complémentaire à l'estimateur proposé dans la section 5.1.1 pour créer le nouveau classificateur GED.

5.1.3.2 Généralisation aux algorithmes de dématriçage par interpolations directionnelles

La méthode d'homogénéisation locale des choix de directions peut être généralisée à tous les algorithmes fonctionnant par interpolations directionnelles. Nous avons testé les performances de la méthode LMDC avec les algorithmes de Hamilton [8], Prescott [9] et Hibbard [47]. Pour les tests, nous avons utilisé un élément structurant de forme carrée et de taille 5×5. Le tableau 5.1 montre les résultats des calculs du RMS sur les 24 images de références de l'annexe D. Le RMS est calculé dans les régions de contours, dans les régions plates et dans l'image entière en utilisant la méthode proposée Lu et al. dans [63]. On peut voir, que pour tous les algorithmes étudiés, la méthode LMDC améliore la qualité des images en réduisant le RMS, aussi bien dans les zones de contours que dans les zones plates. Les figures 5.6 et 5.7, montrent deux exemples d'images dématriçées obtenues en utilisant la méthode de correction par LMDC respectivement avec les algorithmes de Hamilton [8] et Prescott [9]. Pour mieux visualiser l'influence du filtrage on montre les cartes de directions résultantes avant et après filtrage (les valeurs utilisées pour la représentation des cartes sont respectivement 1 pour la direction verticale et 0 pour la direction horizontale). On peut voir que la méthode proposée permet de corriger un nombre important d'erreurs d'estimations et optimise la qualité des images produites. Concernant la complexité des calculs, en utilisant un masque carré de taille n, l'application de la correction nécessite le calcul de $n^2 - 1$ additions et 1 comparaison (on compte le nombre cas positifs pour un choix et on le compare à $(n^2 - 1)/2$). La méthode de LMDC permet donc de minimiser de manière importante les erreurs de choix de directions dans les algorithmes fonctionnant par interpolations directionnelles sans apport significatif de complexité algorithmique.

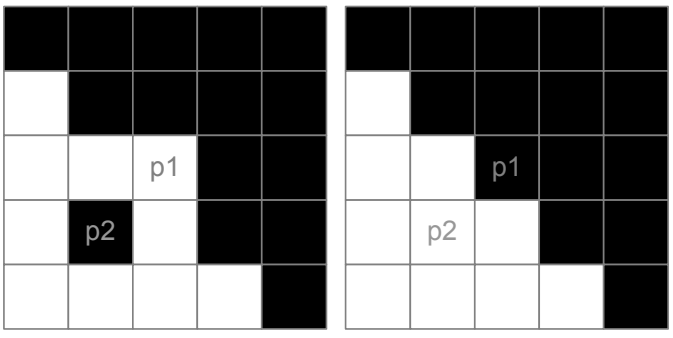
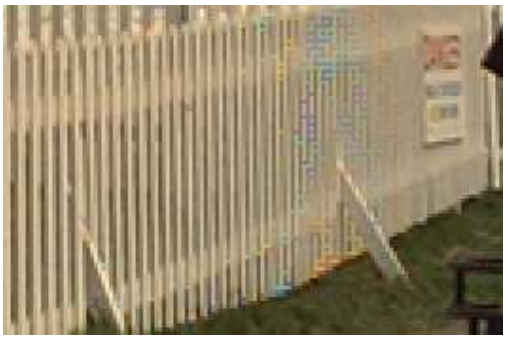


FIGURE 5.5 – Fonctionnement de la correction des erreurs de choix des directions d'interpolations par la méthode de LMCD. Dans cet exemple les pixels noirs représentent, par exemple, le choix d'interpolation horizontale et les pixels blancs représentent le choix d'interpolation verticale. On peut voir que les choix isolés sont considérés comme du « bruit » et corrigés. Dans ce cas, les directions d'interpolations des points $p1$ et $p2$ sont modifiées. La correction par LMDC fonctionne comme un filtrage médian sur les choix des directions d'interpolations.

TABLE 5.1 – Moyenne des calculs des RMS sur 24 images de références, dans les régions de contours, les régions plates et sur l'image entière

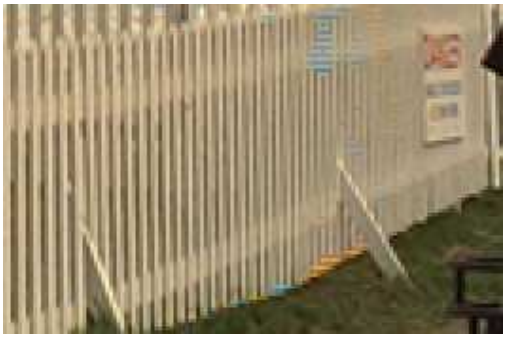
algorithmes	régions de contours	régions plates	image entière
Hamilton [8]	38,21	6,95	13,06
Hamilton+LMDC	35,15	6,28	11,91
Hibbard [47]	44,66	7,55	14,74
Hibbard+LMDC	36,23	6,41	12,22
Prescott [9]	41,78	7,39	14,13
Prescott+LMDC	35,87	6,40	12,14

(a) Carte de décision de Hamilton pour la direction d'interpolation



(b) Hamilton

(c) Carte de décision de Hamilton pour la direction d'interpolation avec correction par LMDC



(d) Hamilton avec correction par LMDC

FIGURE 5.6 – Exemple de réductions des artéfacts par la méthode de LMDC avec l'algorithme de Hamilton [8].

5.2 Réduction des artéfacts d'interpolation (IAR, Interpolation Artefacts Reduction)

Nous avons vu dans la section 4.10, que même avec un algorithme de dématriçage « parfait », l'étape de reconstruction introduit des artefacts de couleurs inhérents à l'interpolation elle-même. Une méthode de réduction des artefacts de couleurs est proposée dans [12]. Elle consiste à diminuer les variations des couleurs à la surface des objets en appliquant un filtre médian sur la différence des plans vert/rouge et vert/-bleu. Cet algorithme comporte quelques désavantages : la forte complexité algorithmique

(a) Carte de décision de Prescott pour la direction d'interpolation

(b) Prescott

(c) Carte de décision de Prescott pour la direction d'interpolation avec correction par LMDC

(d) Prescott avec correction par LMDC

FIGURE 5.7 – Exemple de réductions des artéfacts avec la méthode de LMDC avec l'algorithme de Hamilton [9].

et la non-convergence du filtre médian, limitant l'efficacité de correction des artéfacts. Afin d'améliorer les performances de réduction des artéfacts et de diminuer la complexité du filtre, nous proposons de remplacer la fonction médiane par une fonction de moyenne. Dans cette section, nous comparons les performances du filtre médian, du filtre de moyenne et du filtre bilatéral proposé dans [16] par C.Tomasi et al. On rappelle, que le filtre bilatéral consiste à calculer une moyenne pondérée dans une zone locale centrée

sur le pixel courant. Les poids sont adaptatifs, ils dépendent de la similitude entre les intensités des pixels. Pour les tests nous utilisons un masque de convolution de taille 5×5. Pour évaluer la qualité des images nous utilisons la mesure du RMS et l'appréciation visuelle.

Le tableau 5.2 montre la comparaison des résultats de la moyenne des calculs du RMS sur 24 images de références (voir annexe D) pour l'utilisation des filtres médian, bilatéral et moyenne. Ces images sont préalablement dématriçées en utilisant l'algorithme de Hamilton [8]. On peut voir que le filtre de moyenne augmente les résultats des mesures du RMS par rapport au filtre médian, traduisant ainsi une dégradation de la qualité des images. Inversement, on peut voir que le filtre bilatéral abaisse les résultats des mesures du RMS dans les régions de contours, dans les régions plates et sur l'image entière par rapport au filtre médian, mettant ainsi en évidence une amélioration de la qualité de l'image.

TABLE 5.2 – Moyenne des calculs du RMS sur 24 images de références dans les régions de contours, les régions plates et sur l'image entière

algorithmes	régions de contours	régions plates	image entière
médian	39,98	7,36	13,68
moyenne	71,77	18,27	28,39
bilatéral	34,77	7,28	12,61

Les figures 5.8 et 5.9 montrent les exemples de résultats obtenus. Sur la figure 5.8(c) on peut voir que le filtre de moyenne réduit efficacement les artéfacts de couleurs, cependant il réduit aussi la saturation chromatique. Sur la figure 5.9(c) on peut voir que le filtre de moyenne diffuse les couleurs des objets en dehors de leurs frontières. On peut supposer que les propriétés adaptatives du filtre bilatéral vont permettre d'atténuer ces deux défauts tout en gardant les bonnes propriétés de réduction des artéfacts de couleurs. Sur la figure 5.8(d), on peut voir que le filtre bilatéral réduit les artéfacts de couleurs de manière aussi efficace que le filtre de moyenne et mieux que le filtre médian. Sur la figure 5.9(d) on peut voir que le filtre bilatéral ne diffuse pas les couleurs des objets au delà de leurs frontières. Il possède donc de bonnes propriétés pour cette application.

Le tableau 5.3 montre les complexités algorithmiques des filtres étudiés pour un masque carré de taille $n \times n$. On peut voir que la complexité du filtre bilatéral (avec l'utilisation d'une LUT contenant les poids pré-calculés) est inférieure à celle du filtre médian avec n^2+1 multiplications et n^2-1 additions contre $n^2 \times 2$ additions et $n^2+n \times 4$ comparaisons pour le filtre médian.

On peut conclure que l'utilisation du filtre bilatéral sur les différences des plans $R-G$ et $B-G$ permet d'améliorer la qualité de réduction des artéfacts de couleurs tout en

(a) image originale (b) médian

(c) moyenne (d) bilatéral

FIGURE 5.8 – Résultats obtenus en appliquant des filtrages médian, moyenne et bilatéral sur les différences des plans vert/rouge et vert/bleu.

permettant de réduire la complexité algorithmique. Il est donc une bonne alternative à l'utilisation du filtre médian.

TABLE 5.3 – Complexités algorithmiques des algorithmes de réduction des artéfacts de couleurs.

algorithmes	multiplications	additions	comparaisons
médian $n \times n$		$n^2 \times 2$	$n^2 + n \times 4$
médian 3×3		18	48
moyenne $n \times n$	1	$n^2 - 1$	
moyenne 3×3	1	8	
bilatéral $n \times n$	$n^2 + 1$	$n^2 - 1$	
bilatéral 3×3	10	8	

5.3 Résumé de l'algorithme GEDI

En utilisant le nouvel estimateur GED, nous proposons un nouvel algorithme de dématriçage par interpolations directionnelles GEDI (Green Edge Directed Interpolation). L'algorithme de dématriçage GEDI s'applique en 6 étapes, le pseudo code de l'algorithme est présenté dans l'annexe D, algorithme 6 :

(a) image originale (b) médian

(c) moyenne (d) bilatéral

FIGURE 5.9 – Résultats obtenus en appliquant des filtrages médian, moyenne et bilatéral sur les différences des plans vert/rouge et vert/bleu.

1. interpoler le plan vert verticalement et horizontalement
2. utiliser l'estimateur présenté dans la section 5.1 pour estimer la direction d'interpolation : verticale ou horizontale
3. corriger les erreurs d'estimations de directions d'interpolations par la méthode de LMDC présentée dans la section 5.1.3
4. interpoler le plan vert avec les directions d'interpolations choisies
5. interpoler les plans rouge et bleu par la méthode de constance des teintes présentée dans la section 4.2
6. réduire les artéfacts en utilisant l'algorithme proposé dans la section 5.2

5.4 Évaluation et comparaison de la qualité des images produites

Dans cette section, nous comparons la qualité des images produites avec l'algorithme GEDI par rapport aux qualités des images produites avec les principaux algorithmes de la littérature. Nous comparons aussi les résultats de l'algorithme de Hamilton corrigé par la méthode de LMDC (voir section 5.1.3). Pour la comparaison nous utilisons le critère objectif de la mesure du RMS et le critère subjectif de la qualité visuelle. Le RMS est d'abord mesuré dans les régions de contours, dans les régions plates et sur les trois canaux de couleurs séparément. Pour les tests nous utilisons les images de l'annexe D. Ces images sont échantillonnées en suivant l'arrangement de la mosaïque de Bayer pour simuler l'acquisition par un capteur numérique.

5.4.1 Calcul de l'erreur quadratique moyenne

Le tableau 5.4 montre les résultats des mesures du RMS dans les régions de contours, dans les régions plates et dans l'image entière. On voit que GEDI se classe parmi les meilleurs filtres. Les résultats obtenus sont équivalents à ceux des algorithmes de Hirakawa et Gunturk. D'autre part, on peut voir que la méthode de LMDC permet d'améliorer légèrement la qualité de l'algorithme de Hamilton (voir Hamilton+LMDC). Le tableau 5.5, montre les résultats des mesures du RMS sur les canaux vert, rouge et bleu. Pour une meilleure lisibilité des performances, on représente l'histogramme du tableau sur la figure 5.10. On peut encore voir les bonnes performances de l'algorithme GEDI, équivalentes aux algorithmes de Hirakawa et Gunturk. On peut voir aussi l'amélioration de qualité apportée par la méthode de LMDC sur l'algorithme de Hamilton (voir Hamilton +LMDC).

TABLE 5.4 – Moyenne du calcul des RMS sur 24 images de références dans les régions de contours, les régions plates et sur l'image entière

algorithmes	régions de contours	régions plates	image entière
bilinéaire 3x3	155,18	21,74	45,11
Malvar [4]	45,36	7,16	14,82
Alleyson [54]	58,34	8,43	19,30
Kimmel [49]	51,21	10,60	18,53
Hibbard [47]	40,44	8,03	14,27
Hamilton [8]	35,28	7,60	12,9
Hamilton+LMDC	32,83	7,08	12,06
Hirakawa [18]	29,28	6,54	11,35
GEDI	30,86	6,64	11,58
Gunturk [55]	23,65	5,33	9,19

TABLE 5.5 – Moyenne des calculs des RMS sur 24 images de références dans les canaux rouge, vert et bleu

algorithmes	canal rouge	canal vert	canal bleu
bilinéaire 3x3	44,25	52,23	47,11
Malvar [4]	20,36	8,38	25,17
Alleyson [54]	19,36	16,38	22,17
Kimmel [49]	17,52	12,53	19,05
Hibbard [47]	14	11,53	16,87
Hamilton [8]	12,94	10,53	15,70
Hamilton+LMDC	11,91	9,17	14,64
Hirakawa [18]	11,96	8,07	14
GEDI	11,29	8,69	13,94
Gunturk [55]	9,41	5,69	11,41

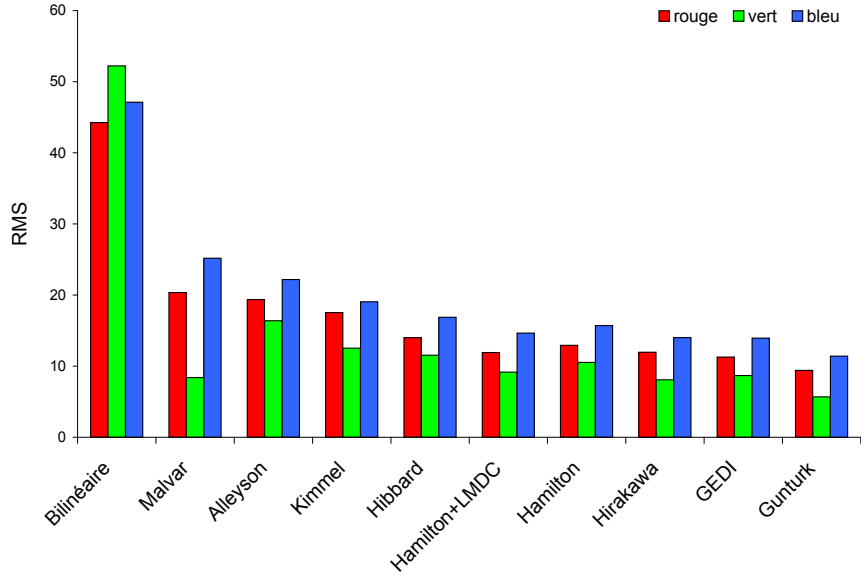
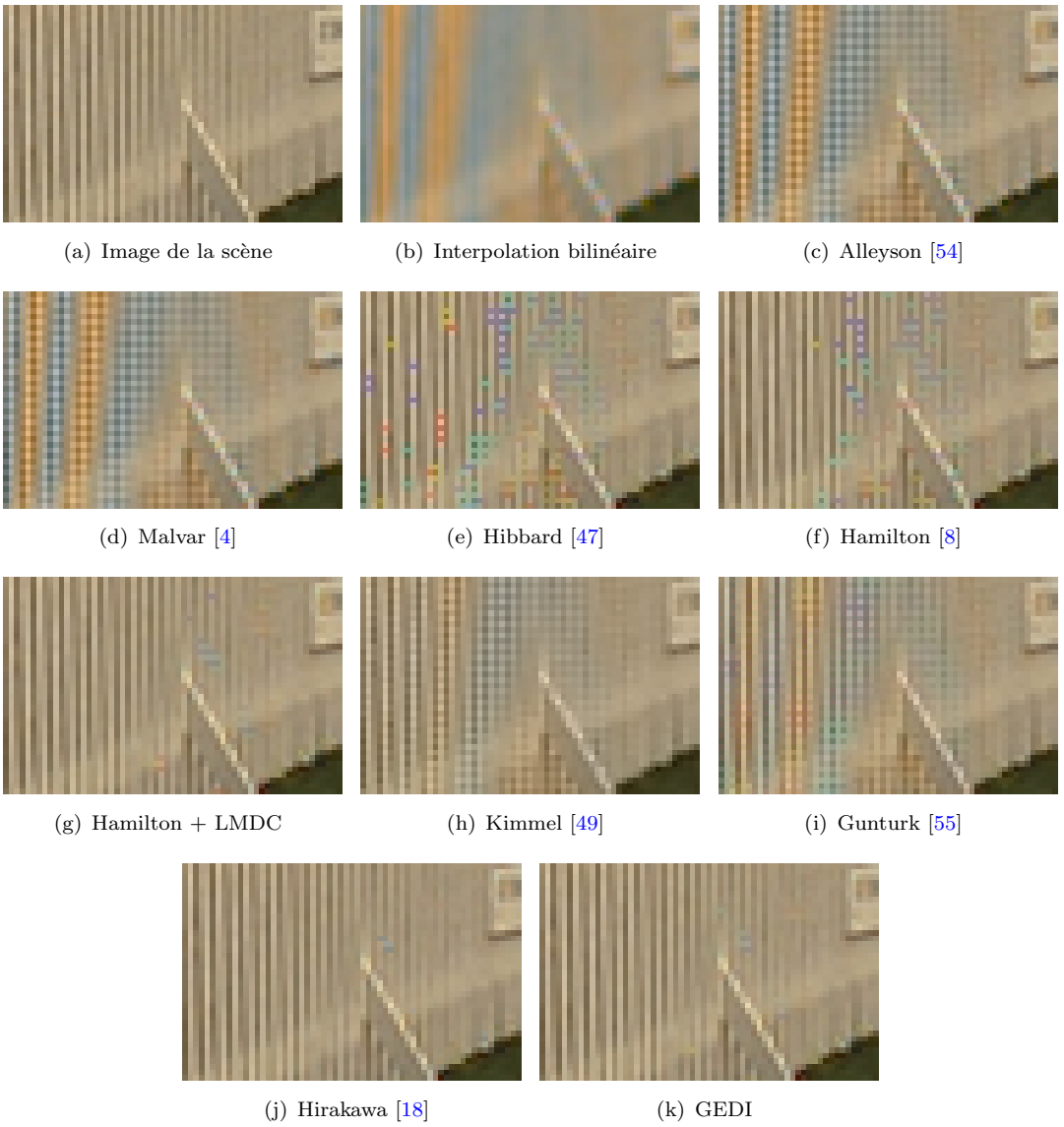


FIGURE 5.10 – Histogramme de la moyenne du calcul du RMS à travers 24 images de références dans les canaux rouge, vert et bleu, obtenue avec les données du tableau 5.5.

5.4.2 Évaluation de la qualité visuelle

Pour évaluer la qualité des images dématriçées, nous utilisons un échantillon d'image issue de la base d'image de l'annexe D. Cet échantillon contient des structures verticales dont les résolutions horizontales sont proches de la résolution de l'image. Cette condition est favorable à l'apparition d'artéfacts de moiré, de fausses couleurs, de structures en formes de labyrinthes et de quadrillage. La figure 5.11 montre les résultats obtenus. On peut voir que l'algorithme GEDI produit une image de qualité équivalente à la qualité d'image produite par l'algorithme de Hirakawa. L'algorithme GEDI, n'introduit pas de flou, pas d'effets de moiré, pas de structures en formes de labyrinthes et très peu d'artéfacts de couleurs. On voit aussi que la correction par LMDC minimise de manière importante les artéfacts introduits par l'algorithme de Hamilton.



(a) Image de la scène (b) Interpolation bilinéaire (c) Alleyson [54]

(d) Malvar [4] (e) Hibbard [47] (f) Hamilton [8]

(g) Hamilton + LMDC (h) Kimmel [49] (i) Gunturk [55]

(j) Hirakawa [18] (k) GEDI

FIGURE 5.11 – Comparaison de la qualité visuelle des différents algorithmes de dématriçage étudiés sur un motif de bandes verticales.

La figure 5.13 montre les résultats de la reconstruction sur une image réelle. La prise de vue a été réalisée avec l'appareil-photo numérique Panasonic DMC-LX1. Comme image de scène, on utilise une mire professionnelle indicatrice de netteté. Pour corriger les artéfacts de couleurs, on utilise la méthode IAR dans sa version bilatérale. La figure 5.12 montre une photographie de la mire utilisée. La figure 5.12 montre la partie centrale de la mire susceptible de faire apparaître des artéfacts de dématriçage. Sur la figure 5.13(b), on peut voir, que la reconstruction par interpolation bilinéaire efface complètement les cercles concentriques visibles sur l'image d'origine, elle introduit du flou, des artéfacts de couleurs et des effets de moiré. Les algorithmes de Alleyson, Malvar, Kimmel et Gunturk introduisent des artéfacts de grille, de couleurs et des effets de moiré. Les algorithmes de Hamilton, Hamilton corrigé par LMDC, Hibbard et Hirakawa, introduisent des artéfacts

de couleurs et des structures en formes de labyrinthes. Dans ce cas, on remarque que la méthode de LMDC utilisée avec l'algorithme de Hamilton n'améliore pas nettement la qualité de l'image. Finalement, on voit que l'algorithme GEDI n'introduit aucuns défauts de dématriçage et délivre la meilleure qualité d'image. L'ensemble des images de l'annexe D ont été échantillonnées puis dématriçées avec l'algorithme GEDI, les résultats sont présentés dans l'annexe E.

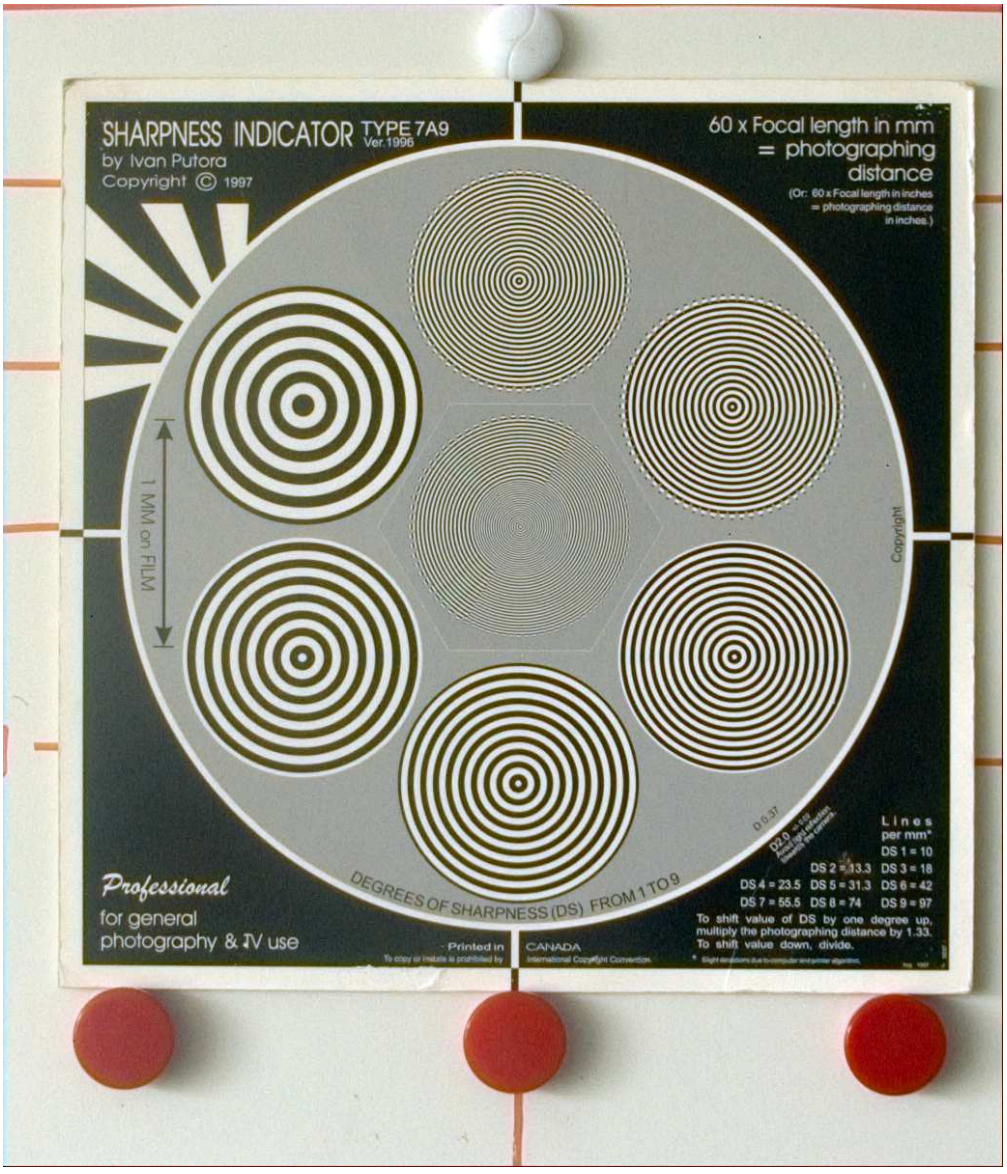


FIGURE 5.12 – Mire indicatrice de netteté utilisée pour évaluer les qualités des algorithmes de dématriçage.

5.4.3 Complexité algorithmique

Le tableau 5.6 montre les complexités algorithmiques de chacune des méthodes étudiées. On compte le nombre d'opérations d'additions, de multiplications, de comparaisons et de

valeurs absolues nécessaires pour construire un pixel coloré. On ne prend pas en compte l'étape de réduction des artéfacts de couleurs qui est commune à tous les algorithmes. Le compte détaillé des opérations de l'algorithme GEDI est présenté dans le tableau 5.7. Dans le tableau 5.6, on voit que la complexité de l'algorithme GEDI est beaucoup plus faible que celle des algorithmes de Alleyson, Hirakawa, Gunturk et Kimmel, avec un nombre d'opérations de calculs se rapprochant de celui des méthodes peu complexes tels que l'algorithme de Hamilton, Malvar et Hibbard.

TABLE 5.6 – Comparaison des complexités de calculs des algorithmes

	multiplications	additions	comparaisons	valeurs absolues
bilineaire 3x3	4	3		
Malvar [4]	19	21		
Alleyson [54]	61	70		
Kimmel [49]	120	180		
Hibbard [47]	6	12	1	2
Hamilton [8]	13/2	14	1	4
Hamilton+LMDC	13/2	38	2	8
Hirakawa [18]	48	88	71	12
GEDI	7	32	2	8
Gunturk [55]	480	480		

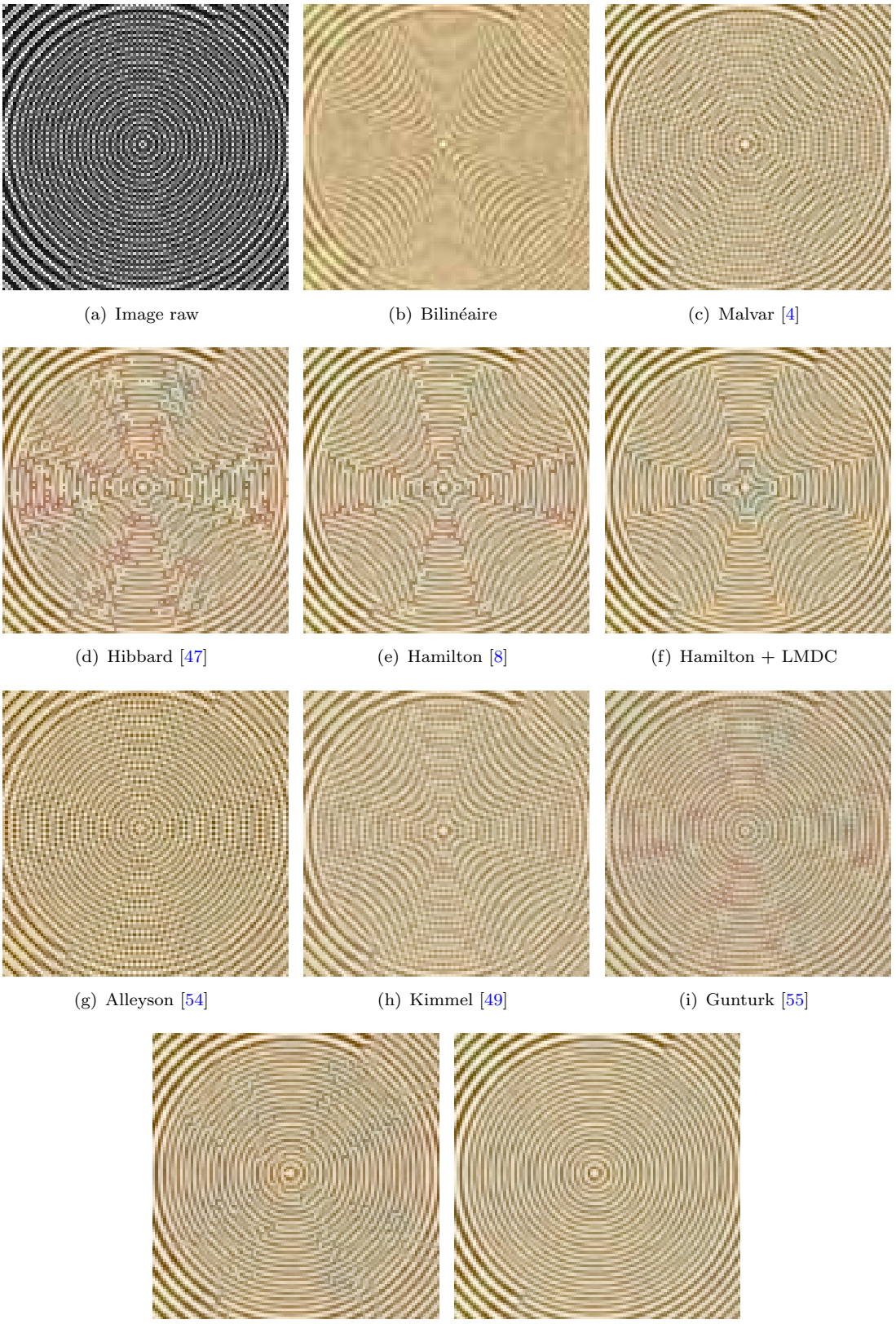
TABLE 5.7 – Complexité de calculs détaillée de l'algorithme de GEDI

		multiplications	additions	comparaisons	valeurs absolues
$G(.)$	$G_h(.)$ et $G_v(.)$	6	8		
	estimateur		14	1	8
LMDC			8	1	
$R(.)\&B(.)$		1	2		
Total/pixel		7	32	2	8

5.5 Conclusion

Dans ce chapitre nous avons proposé un estimateur de directions d'interpolations simple et efficace. Cet estimateur fonctionne en deux étapes. Une première étape estime la direction des objets en utilisant judicieusement les informations des gradients des plans verts interpolés verticalement et horizontalement. Une deuxième étape homogénéise localement les directions choisies pour minimiser les erreurs d'estimations. Cet estimateur permet de formuler un nouvel algorithme de dématriçage par interpolations directionnelles GEDI (Green Edge Directed Interpolation). Les mesures de qualités des images produites montrent que GEDI se classe parmi les meilleurs algorithmes de la littérature. L'étude des complexités algorithmiques montrent que GEDI possède une complexité de

calculs très inférieure aux algorithmes de sa catégorie. Ces deux qualités remarquables semblent prometteuses pour une implémentation sur des systèmes embarqués pour appareils mobiles. Dans le chapitre 8, nous allons implémenter les algorithmes de Hirakawa [18], Hamilton [8], Hamilton corrigé par LMDC et GEDI sur le processeur multimédia TM3270 et comparer leurs performances en temps d'exécutions. **Les travaux présentés dans ce chapitre ont mené au dépôt de deux brevets européens. Un premier brevet décris la méthode de minimisation des erreurs de choix de directions LMDC [14]. Un deuxième brevet est consacré à la description de l'algorithme de dématriçage GEDI [13]. Ces travaux ont aussi mené à la soumission d'une publication dans le journal IEEE Transactions On Consumer Electronics [15].**



(a) Image raw (b) Bilinéaire (c) Malvar [4]

(d) Hibbard [47] (e) Hamilton [8] (f) Hamilton + LMDC

(g) Alleyson [54] (h) Kimmel [49] (i) Gunturk [55]

(j) Hirakawa [18] (k) GEDI

FIGURE 5.13 – Comparaison de la qualité visuelle des différents algorithmes de dématriçage étudiés.

Chapitre 6

Bruit : État de l'art

Une image photographique est le résultat de la projection d'un signal lumineux sur une surface photosensible. Cette surface photosensible à pour rôle d'échantillonner et de transformer l'information de lumière en signal électrique. Une fois numérisé, le signal est traité au travers d'une architecture multimédia pour produire une image visualisable. Il existe trois sources fondamentales de bruits apparaissant au cours du processus de formation d'une image photographique ou vidéo. La première source de bruit est introduite par la projection du signal lumineux sur la surface photosensible à travers un système de lentilles. Cette étape est susceptible d'introduire des distorsions géométriques, des aberrations chromatiques, du flou, du vignetage, etc. (3). La deuxième source de bruit associée à la transformation du signal lumineux en signal électrique à travers le capteur numérique. Cette étape introduit principalement des bruits électroniques d'origines diverses et du bruit de photon lié à l'impossibilité fondamentale d'associer une luminosité à un nombre précis de photons. La troisième source de bruit est introduite par le traitement du signal issu du capteur pour produire une image visualisable. Parmi ces traitements, on peut citer, le dématriçage, l'adaptation de la balance des blancs, la correction des aberrations géométriques, etc. Chacune de ces étapes transforme l'image en manipulant les valeurs des pixels, et introduit des artéfacts assimilables à du bruit, c'est le bruit algorithmique. Dans ce chapitre, nous nous intéressons principalement aux bruits introduits par les capteurs. En effet, malgré des améliorations constantes, les appareils photographiques numériques souffrent toujours d'une importante dégradation de la qualité des images, qui est introduite par les bruits de capteurs. L'importance de la présence de ces bruits dans l'image est directement liée au processus de fabrication industriel, à l'intensité d'illumination de la scène, la sensibilité du capteur (sensibilité ISO), le temps d'exposition, la taille des photosites, etc. Le bruit est particulièrement présent dans les appareils miniatures et fabriqués à faibles coûts, tels que ceux que l'on trouve dans les téléphones mobiles. La présence de bruit dans une image est subjectivement désagréable et c'est un facteur

majeur dans l'estimation de la qualité des appareil-photos numériques. D'autre part, la présence de bruit peut perturber le fonctionnement de certains algorithmes comme le dématriçage (voir chapitre 4), la compression où des applications telles que la détection et le suivi d'objets. Le bruit peut aussi être amplifié par certains post-traitements, comme le renforcement des contours et l'amélioration du contraste, etc. Pour toutes ces raisons il est important d'éliminer ou de réduire le bruit. La réduction du bruit dans les images est un problème délicat et abondamment traité dans la littérature. La problématique majeure, est de réussir à éliminer le bruit, tout en préservant les structures et les détails de l'image. La réduction du bruit peut être améliorée par la compréhension des origines et du comportement du bruit. Dans la première section, nous présentons les différentes sources de bruits dans les capteurs photographiques et l'influence des conditions de prises de vues sur leurs comportements. Nous présentons ensuite comment ces bruits sont modélisés dans la littérature. Dans une deuxième section, nous présentons un état de l'art non-exhaustif des algorithmes de réduction du bruit. En raison des contraintes d'architectures matériels imposées par les systèmes d'implémentations considérés (appareils photos-numériques compacts, téléphones mobiles, etc.), nous nous intéresserons particulièrement aux techniques de filtrages à voisinage local.

6.1 Les sources de bruit dans les capteurs numériques et leurs modélisations

6.1.1 Les sources de bruit dans les capteurs d'images numériques

L'acquisition et la transformation du signal lumineux de la scène en signal électrique à partir d'un capteur numérique n'est pas un processus idéal. Le signal est perturbé par de nombreuses sources de bruit, issues à la fois des propriétés physiques de la lumière (nature corpusculaire de la lumière) et des propriétés électroniques des capteurs. Dans cette section nous répertorions les différentes sources de bruit présentes dans les capteurs CMOS et CCD. Tout d'abord nous présentons le bruit introduit par la non-uniformité de réponse des photosites puis, le bruit de photon, le bruit de courant d'obscurité, le bruit de lecture et enfin le bruit introduit par la non-uniformité d'amplification des gains des photosites. Des descriptions complètes concernant les bruits des capteurs CCD et CMOS sont détaillées dans [64–67].

Le processus de fabrication des capteurs CCD et CMOS n'est pas idéal et les caractéristiques de sensibilités (rendement quantique et facteur de remplissage (chapitre 3)) à l'intensité lumineuse varient légèrement d'un photorécepteur à un autre [67]. Ce phénomène est connu sous le nom de non-uniformité de réponse des photorécepteurs,

aussi appelé PRNU (Photon Response Non Uniformity) ou encore bruit de motif (Fixed Pattern Noise). Ce bruit est nul lorsque le capteur n'est pas exposé à la lumière ou lorsqu'un pixel est considéré individuellement. L'amplitude de ce bruit est proportionnelle à l'amplitude du signal lumineux[65].

La distribution des photons sur la surface d'un capteur est gouvernée par un processus de Poisson. Ce processus introduit le bruit de photon, aussi appelé bruit de grenaille de photon ou encore photon shot noise. Afin de mieux comprendre et visualiser ce phénomène, on peut assimiler la distribution des photons sur la surface d'un capteur à la distribution des gouttes d'eau de pluie (représentant les photons) tombant sur une matrice de puits (représentant les photosites du capteur). En considérant les propriétés d'un processus de Poisson, le nombre de photons N percutant une surface donnée se disperse autour d'une valeur moyenne λ possédant un écart type σ tels que $\sigma = \sqrt{\lambda}$. Dans cette expression, la valeur moyenne λ est égale à $\lambda = I.S.T$ avec I représentant l'intensité de la source de lumière, S représentant la surface du photorécepteur et T représentant le temps d'intégration de la lumière par le capteur. Le bruit de photon est une limitation fondamentale et intrinsèque à la nature corpusculaire de la lumière, il est impossible à éliminer physiquement. Ce bruit est commun à tous les appareils de captures d'images numériques [67, 68].

Le bruit de courant d'obscurité est aussi connu sous le nom de « dark shot noise » ou encore bruit de grenaille du courant d'obscurité. Le courant d'obscurité est affecté par les recombinaisons d'électrons et de trous dans la zone de déplétion d'un photosite (zone d'accumulation des éléctrons générés par l'absorption des photons, voir section 3.1.2). Comme le soulignent E. Healey et al. dans [67], les électrons produits par ces recombinaisons sont indistinguables des électrons générés par effet photoélectrique. Il produisent un bruit de grenaille du courant d'obscurité N_d. Ce bruit est proportionnel à la racine carrée du nombre d'électrons générés dans l'obscurité par l'agitation thermique. Il est indépendant de la quantité de photons percutant le capteur. Un moyen de réduire efficacement ce bruit est de réduire le courant d'obscurité en refroidissant le capteur. Il est important de noter que le courant d'obscurité n'est pas spatialement uniforme et que l'écart de courant d'obscurité entre les photosites est beaucoup plus grand que l'écart de leur sensibilité (voir [64]).

Le bruit de lecture est un bruit thermique provenant de la structure de l'amplificateur de sortie (MOS, Metal Oxide Semiconductor). Il est constant et indépendant du nombre de photons incidents. Il peut-être réduit par un refroidissement de cette structure (coupure de l'alimentation pendant le temps d'intégration, c'est le refroidissement forcé). Le bruit de lecture est indirectement dépendant des courants qui circulent dans la structure de l'amplificateur. L'effet Joule produit par la circulation du courant de sortie est

proportionnel au carré de celui-ci, l'énergie dissipée est égale à : $W = R \times I^2 \times t$, avec R représentant la résistance du matériau, I est le courant de sortie et t est le temps d'intégration du signal. La tension du bruit est proportionnelle à la racine carrée de la température, telle que : $U = \sqrt{4kTR\Delta f}$, avec k représentant la constante de Boltzman, T est la température absolue en Kelvin et Δf est la bande passante utile. Plus ce courant est faible, moins il circule longtemps dans le circuit et plus le bruit thermique est faible. Au bruit thermique s'ajoute le bruit en $1/f$ dont l'origine est mal connu. Ce bruit est commun à tous les semiconducteurs et ses effets ne sont sensibles que pour les signaux de faibles fréquences ($f < 1KHz$). D'autre part, dans un capteur CMOS, chaque photorécepteur possède son propre circuit de conversion, introduisant de cette manière une non-uniformité d'amplification du gain [64, 66].

Le bruit de quantification ou de conversion analogique/numérique est un processus non-linéaire et il est difficile de le décrire de manière analytique. L'approximation la plus courante est basée sur la considération que le signal analogique est stochastique et que toutes les amplitudes de signal possèdent la même probabilité d'apparition. Le bruit de quantification peut alors être décrit avec une moyenne :

$\mu_q = \mu$

et une variance :

$\sigma_q^2 = \sigma^2 + \frac{1}{12}$

Dans ces formules, μ et σ représentent respectivement la moyenne et la variance du signal analogique [69, 70].

Dans [66], Costantini et al. proposent une classification des bruits distinguant les bruits spatiaux fixes (FPN, Fixed Pattern Noise) et les bruits aléatoires (random noise). Cette classification est présentée dans le tableau 6.1. L' opposition entre le bruit spatial fixe et le bruit aléatoire est aussi mise en avant par R.L.Baer dans [71]. Le bruit spatial fixe est constant pour un capteur donné, il reste le même d'une prise de vue à une autre. Les perturbations introduites par le bruit aléatoire changent au cours du temps, le bruit généré est différent pour chaque prise de vue. Idéalement, une estimation du bruit spatial fixe peut être obtenue par l'acquisition d'une série d'images en maintenant l'obturateur fermé. Cette estimation de l'image du bruit peut ensuite être soustraite aux photographies produites par ce même appareil pour éliminer le bruit spatial fixe de manière efficace. D'autre part, le bruit aléatoire, de part sa nature, est difficile à estimer et donc difficile à supprimer ou à atténuer sans dégrader le signal utile de l'image. Une deuxième classification des signaux de bruit est considérée dans le tableau 6.2. Dans cette classification, les sources de bruit sont ordonnées par rapport à leurs dépendances à certains paramètres de prise de vue tels que la température, le temps d'exposition ou la

dépendance au signal. On remarque qu'un même bruit peut être dépendant de plusieurs paramètres. Par exemple, on peut voir que le bruit de photon est à la fois dépendant du signal et du temps d'exposition. Il est important de noter que tous les bruits cités dans les tableaux 6.1 et 6.2 ne contribuent pas avec la même importance dans l'image.

TABLE 6.1 – Classification des bruits des capteurs numériques d'images en fonction de leurs propriétés de variances temporelles

Bruit aléatoire	Bruit spatial fixe
Bruit de photon	Non uniformité du gain
Courant d'obscurité	PRNU
Bruit de lecture	Non uniformité du bruit d'obscurité

TABLE 6.2 – Classification des bruits de capteurs par rapport aux conditions de prises de vues

Signal	Température	Temps d'exposition
Bruit de photon	Bruit de lecture	Bruit de photon
PRNU	Courant d'obscurité	
Non uniformité du gain		

6.1.2 Modélisation des bruits des capteurs d'images numériques

Nous avons vu, que l'image produite à la sortie d'un capteur souffre de la présence de composantes de bruit d'origines variées qu'il est très difficile d'identifier. L'influence de ces bruits sur l'image produite dépend des paramètres d'acquisition, tels que la température, la luminosité de la scène, la sensibilité ISO, etc. De plus, étant donné sa dépendance au processus de fabrication et à l'architecture des capteurs, le bruit visible dans une image varie d'un capteur à un autre. Nous avons vu qu'une estimation des composantes du bruit spatial fixe peut être obtenue en moyennant les images d'une série de prises de vues acquises en maintenant l'obturateur fermé. Cependant, l'élimination du bruit dynamique reste un problème difficile. Il apparaît nécessaire de modéliser ce bruit de façon à évaluer et comprendre son influence sur la qualité des images produites. On pourra ainsi proposer des estimateurs de bruit judicieux et exploiter de manière plus efficace les propriétés des filtres de débruitage existants. Dans la littérature l'ensemble des bruits d'un capteur d'image numérique est généralement considéré comme un bruit global modélisé par un bruit blanc gaussien additif et stationnaire. L'équation 6.1 décrit la fonction de densité de probabilité de la loi normale, d'espérance μ et d'écart type σ. Dans [72], H.Faraji et al. justifient l'utilisation d'un modèle gaussien pour un certain intervalle d'intensité lumineuse de la scène. Dans ces conditions, le calibrage par la courbe de transfert des photons utilisés pour les modèles de capteur CCD [73] montre que le bruit dominant est poissonnien, principalement combiné par du bruit de photon et du

bruit de courant d'obscurité. Ils considèrent que ce bruit peut-être approximé par un bruit blanc gaussien stationnaire.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{6.1}$$

Dans [67], Healey et al. proposent un modèle de formation d'image prenant en compte les sources de bruit ajoutées à chaque étape de l'acquisition du signal. Idéalement, ils proposent d'exprimer l'intensité, c'est à dire le nombre d'électrons collectés par un capteur, de la manière suivante :

$$Y = T \int_{\lambda}\int_{x}\int_{y} B(x,y,\lambda) S_r(x,y,\lambda) q(\lambda) dx dy d\lambda \tag{6.2}$$

ou T représente le temps d'exposition, (x, y) sont des coordonnées continues dans le plan du capteur, $B(x, y, \lambda)$ est l'amplitude de la lumière irradiante, $q(\lambda)$ est le rendement quantique (nombre de photons collectés par rapport au nombre de photons percutant le photosite). $S_r(x, y)$ est la sensibilité du capteur à la position (x, y). Le signal délivré par le capteur est ensuite modélisé par :

$$Y_G = (KY + N_{photons} + N_f + N_z)G + N_Q \tag{6.3}$$

où Y est définie dans 6.2, K représente la sensibilité du photosite, K est caractérisé comme ayant une moyenne ègale à 1 et une variance σ_K^2 sur l'ensemble de la matrice. Plus la fabrication du capteur est minutieuse, plus la variance σ_K^2 est petite. $N_{photons}$ est le nombre d'électrons introduits par le bruit de photon. N_F représente le nombre d'électrons introduits par le courant d'obscurité. N_Z représente le nombre d'électrons introduits par le bruit de lecture. G représente le gain d'amplification du signal. N_Q est le nombre d'électrons introduits par le bruit de quantification de la conversion analogique/numérique. Dans [66], Costantini et al. proposent de modéliser l'image produite par le capteur d'une manière similaire. Cependant, plutôt que de considérer le nombre de photons introduits par chaque source de bruit, les auteurs proposent de considérer l'aspect additif ou multiplicatif des bruits sur le signal d'origine, leurs dépendances et leurs statistiques. Ils proposent d'exprimer Y_G l'image bruitée, tels que :

$$Y_G = G.[P_{poiss}(F) + G_{PRNU}.P_{poiss}(u)] + Z \tag{6.4}$$

où G, F, G_{PRNU} et Z modélisent respectivement la non-uniformité d'amplification du gain, le courant d'obscurité, la non uniformité de la sensibilité des photorécepteurs et les bruits introduits par les circuits (bruits de lecture). L'opérateur P_{poiss} induit une statistique poissonnienne, il est utilisé pour modéliser les bruits de courant de lecture et le bruit de photon. En développant cette expression, on obtient la formulation suivante :

$$Y_G = G.G_{PRNU}.P_{poiss}(u) + G.P_{poiss}(F) + Z \tag{6.5}$$

cette formulation permet d'exprimer l'image bruitée comme la somme d'un signal dépendant de l'image $Y_{Gsignal}$ et d'un signal dépendant de la température Y_{Gtemp}, tels que :

$$Y_{Gsignal} = G.G_{PRNU}.P_{poiss}(u) \tag{6.6}$$

$$Y_{Gtemp} = G.P_{poiss}(F) + Z \tag{6.7}$$

Dans la section 3, nous avons vu que pour qu'un appareil-photo numérique produise une image visualisable, il est nécessaire d'appliquer plusieurs traitements numériques sur le signal délivré par le capteur tels que le dématriçage, la correction des aberrations chromatiques et géométriques, le rehaussement du contraste, la balance des blancs, etc. Chacun de ces traitements modifie l'image, introduit des artéfacts et modifie les statistiques du bruit. La majorité des publications traitant les bruits de capteurs ne prennent pas en compte l'impact de la chaîne de traitement d'image sur le modèle du bruit. Les articles mentionnant ce point [7, 72, 74–77] préconisent que l'image doit être filtrée le plus tôt possible dans la chaîne de traitement, c'est à dire avant l'étape de dématriçage. Dans la section suivante, nous présentons un état de l'art non-exhaustif, des méthodes de restauration des images bruitées en imagerie numérique.

6.2 Etat de l'art des algorithmes de réduction du bruit

Malgré les évolutions constantes des capteurs numériques et les propositions récentes de méthodes sophistiquées de débruitage, les images numériques souffrent toujours de la présence du bruit et la recherche d'algorithmes efficaces de filtrage du bruit reste un domaine de recherche très actif. La réduction du bruit est un vaste et délicat sujet du traitement des images et constitue un outil fondamental dans ce domaine. Dans le cadre de la restauration des images bruitées, il a pour but d'éliminer les composantes parasites introduites par les différents processus de formation de l'image. De nombreuses méthodes lui ont été consacrées. Des états de l'art sont présenté dans les thèses de N.Azzabou [78]

et A.Buades [79]. Dans cette section, nous présentons les méthodes les plus utilisées pour la restauration des images bruitées. Nous commençons par décrire les méthodes de filtrages linéaires, les méthodes de filtrages optimales comme le filtre de Wiener, les filtres d'ordre ou de rang, les filtres adaptatifs, les filtres basés sur la minimisation de l'énergie, les filtres utilisant l'analyse des coefficients des transformées en ondelettes et enfin les filtres utilisant les propriétés de redondance dans l'image.

6.2.1 Filtres linéaires

Dans les conditions les plus triviales, on considère que le signal de l'image est stationnaire et on adopte une approche par filtrage linéaire. On rappelle qu'un filtrage est linéaire si un signal filtré composé de la combinaison linéaire de deux signaux est égale à la combinaison linéaire des deux signaux filtrés séparément. De manière intuitive, on cherche des filtres passe-bas permettant d'éliminer les variations de très hautes fréquences du signal devant correspondre au bruit. Le signal est donc convolué par une fonction de transfert spatiale, tels que le signal estimé $\hat{u}$ du signal u soit égal à :

$$\hat{u}(x) = h \star u(x) \tag{6.8}$$

Dans laquelle $\star$ représente le symbole de la convolution. En limitant le noyau de convolution à un voisinage local $B\rho(x)$ autour de la position du pixel x et en choisissant une fonction de transfert gaussienne, tels que $h(x) = e^{-\frac{|x|}{\rho^2}}$, on peut alors écrire le filtre de la manière suivante :

$$\hat{u}(x) = \frac{1}{C(x)} \int_{B\rho(x)} e^{-\frac{|x-y|}{\rho^2}} u(y)dy \tag{6.9}$$

où $C(x)$ est un paramètre de normalisation et ρ définit la bande passante du filtre, si ρ tend vers 0, alors la fonction de transfert se rapproche de l'identité et si ρ tend vers l'infini, la fonction de transfert se rapproche d'un filtre de moyenne. Ces filtres ne sont pas adaptés aux débruitage des images photographiques, car ils dégradent de façon considérable les contours et rendent les images floues. Les filtres linéaires sont souvent caractérisés par leur fonction de transfert, qui permet de relier l'entrée et la sortie d'un signal en se plaçant dans l'espace de Fourrier ou de Laplace. Les filtres optimaux répondent à certains critères d'optimisations comme par exemple, la minimisation de l'erreur quadratique moyenne. Prenons l'exemple du filtre de Wiener.

6.2.1.1 Le filtre de Wiener

Le filtre de Wiener est un filtre linéaire largement utilisé pour la restauration des images. Considérons une image $u(x)$, convoluée par la fonction de transfert $h(x)$ auquel nous avons ajouté un bruit additif ξ. Le signal de sortie $Y(x)$ au point x s'écrit alors :

$$Y(x) = (h)(x) + \xi(x) \tag{6.10}$$

Cette équation peut être écrite de manière vectorielle en notant Y l'ensemble des données du signal de sortie, u l'ensemble des données du signal d'entrée, H la matrice de transfert et ξ le vecteur de bruit. L'équation 6.10 devient alors :

$$\mathbf{Y} = \mathbf{Hu} + \xi \tag{6.11}$$

Le filtre de Wiener minimise l'erreur quadratique moyenne tels que :

$$\hat{\mathbf{u}} = \text{argmin} E[(\hat{\mathbf{u}} - \mathbf{u})^T(\hat{\mathbf{u}} - \mathbf{u})] \tag{6.12}$$

En appliquant le principe d'orthogonalité, on obtient :

$$E[(\hat{\mathbf{u}} - \mathbf{u})\mathbf{Y}^T] = 0 \tag{6.13}$$

En cherchant une solution linéaire du type $\hat{\mathbf{u}} = \mathbf{WY}$ et sous l'hypothèse que u et ξ ne sont pas corrélés, ont obtient l'expression vectorielle du filtre de Wiener :

$$\mathbf{W} = \mathbf{R}_{uu}\mathbf{H}^T(\mathbf{H}\mathbf{R}_{uu}\mathbf{H}^T + \mathbf{R}_{\xi\xi})^{-1} \tag{6.14}$$

Dans cette expression $\mathbf{R}_{uu} = E[\mathbf{u}\mathbf{u}^T - E[\mathbf{u}]]$ et $\mathbf{R}_{\xi\xi} = E[\xi\xi^T - E[\xi]]$ les matrices d'autocorrélations respectives du signal u et du bruit ξ. Le filtre de Wiener s'exprime dans le domaine de fourrier comme :

$$W = \frac{H^+ S_{uu}}{H^2 S_{uu} + S_{\xi\xi}} \tag{6.15}$$

avec W, H et S_{uu} les transformées de Fourrier respectives du filtre de Wiener, h et u.

Le filtrage des images par le filtre de Wiener n'est pas direct. En effet, il est nécessaire d'estimer la densité spectrale de puissance du signal. De plus, en pratique la stationnarité du signal n'est pas vérifiée.

6.2.1.2 Le filtre de Wiener local adaptatif

Dans [80], Lee propose une version adaptée du filtre de Wiener pour les images. Cet algorithme est aussi appelé LLMMSE (Local Linear Minimum Mean Squarre Error), ou encore filtre de Lee. Lee considère un modèle de bruit blanc gaussien, possédant une variance σ^2 et considère que le signal est stationnaire au voisinage d'un point $u(x)$ de l'image. Dans ce voisinage le signal peut alors s'écrire comme [81, 82] :

$$u(x) = m(x) + s(x)\epsilon(x) \tag{6.16}$$

où $m(x)$ et $s(x)^2$ représentent respectivement, la moyenne et la variance du signal au voisinage de x et $\epsilon(x)$ représente un bruit blanc de variance 1. En estimant $m(x)$ et $s(x)^2$ pour chaque point de coordonnée x, la solution du filtre est donnée par :

$$\hat{u}(x) = m(x) + \frac{s(x)^2}{s(x)^2 + \sigma^2}\left(Y(x) - m(x)\right) \tag{6.17}$$

On peut voir qu'il est nécessaire d'estimer la variance σ^2 du bruit dans l'image pour pouvoir appliquer le filtre. Ce filtre se comporte comme un filtre de moyenne dans les zones homogènes et s'adapte pour conserver des discontinuités dans les zones de détails.

6.2.2 Les filtres d'ordre ou de rang

Pour les filtrages de rang, on considère que le pixel doit être estimé à partir des pixels bruités dans l'image. Ces filtres consistent à trier par amplitude croissante les valeurs des pixels appartenant à une fenêtre centrée sur le pixel à traiter. La valeur du pixel est remplacée par la valeur d'un des pixels choisi dans la liste ordonnée. Si on choisi la valeur de plus grande amplitude, cela revient à appliquer la dilatation en morphologie mathématique (voir [83]), si on choisi la valeur de plus petite amplitude, cela revient à appliquer une érosion en morphologie mathématique (voir [83]), enfin, si on choisi la valeur centrale, cela revient à appliquer un filtre médian [84].

6.2.3 Filtres adaptatifs

6.2.3.1 Filtres à fenêtres adaptatives

Dans le cas des filtres à fenêtres adaptatives, on cherche à sélectionner une fenêtre dans une famille de fenêtres existantes placées autour du pixel à filtrer et dans laquelle le signal

est le plus adapté pour le filtrage du pixel considéré. Clairement, on cherche le détail local auquel appartient le pixel à traiter. Dans [85], Nagao définit 9 fenêtres de voisinage, ces 9 fenêtres sont montrées sur la figure 6.1. Dans chaque fenêtre, Nagao propose de calculer la moyenne et la variance des intensités des pixels. Il propose de remplacer le pixel considéré par la valeur moyenne de la fenêtre dont la variance est la plus faible. Dans [86], Wu et al. proposent une fenêtre dynamique s'adaptant par croissance aux régions homogènes autour du pixel considéré, tout en empêchant la croissance de cette fenêtre dans les zones hétérogènes. Lorsque la fenêtre ne peut plus croître, le point traité est remplacé par la moyenne des intensités des pixels calculés dans la plus grande fenêtre obtenue. Ces méthodes de filtrage améliorent la netteté des contours, cependant, elle détruisent les détails fins et créent des zones d'intensités plates dans l'image.

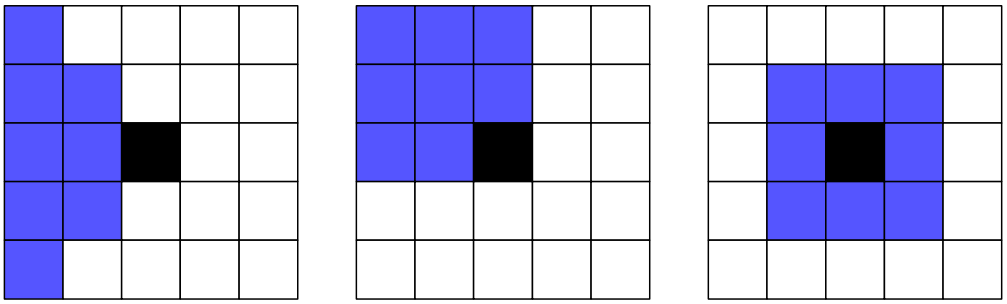


FIGURE 6.1 – Les 9 fenêtres du filtre de Nagao : 4 se déduisent de la fenêtre représentée à gauche et de la fenêtre centrale par 4 rotations consécutives de 90°

6.2.3.2 Filtres à coefficients adaptatifs

Les filtres à coefficients adaptatifs sont inspirés des filtres de moyenne. Dans ces filtres, le pixel considéré est remplacé par la moyenne pondérée des intensités des pixels appartenant au masque de traitement. Différentes méthodes sont proposées pour estimer le poids de chaque pixel dans le calcul de la moyenne. Certaines méthodes utilisent les propriétés de similarités de positions et d'intensités entre le pixel traité et les pixels appartenant au masque, d'autre méthodes exploitent les propriétés de symétries des informations d'intensités des pixels à l'intérieur du masque.

Dans [87], D.C.C Wang propose un opérateur de poids de gradients inversés pour calculer la similarité entre le pixel considéré et les pixels du masque. Les valeurs des poids dépendent de la norme de la différence des intensités entre le pixel central et le pixel à pondérer. De cette manière, plus la valeur absolue de la différence d'intensité entre les points considérés est importante, plus le poids du pixel est faible dans le calcul de la moyenne. Généralement la moyenne est calculée dans une fenêtre glissante β_ρ de taille $(2m+1) \times (2m+1)$. Il n'y a aucun seuil à régler. Ce filtre permet de réduire le bruit gaussien tout en conservant les structures dans l'image. On peut noter qu'une variante

de ce filtre ne considérant pas le pixel traité dans la moyenne permettrait de mieux filtrer le bruit impulsionnel. Ce filtre est appliqué de manière itérative, typiquement 5 fois pour obtenir un filtrage satisfaisant (voir [88]).

$$\hat{u}(x) = \frac{\sum_{\beta_\rho} \frac{Y(y)}{max\{\frac{1}{2}, |Y(x)-Y(y)|\}}}{\sum_{\beta_\rho} \frac{1}{max\{\frac{1}{2}, |Y(x)-Y(y)|\}}} \tag{6.18}$$

Dans [89], D.Harwood et al. proposent un filtrage pondéré par sélection de symétrie de la structure de l'image dans un masque carré de taille fixe (SNN filter : "Symmetric nearest neighbors filter"). Les poids utilisés sont binaires. D.Harwood et al. proposent ce filtre pour réduire la détérioration des contours des objets dans l'image. Le filtre fonctionne de la manière suivante : on compare chaque pixel de la fenêtre avec son pixel symétrique par rapport au point traité en calculant la valeur absolue de la différence des intensités. Le pixel sélectionné pour le calcul de la moyenne est le pixel dont l'intensité est la plus proche du pixel considéré. Si les différences calculées pour un couple de pixels considérés sont égales, alors, c'est l'intensité du pixel considéré qui est prise en compte dans le calcul de la moyenne. Ce filtre utilise principalement des fenêtres carrées de taille $(2m+1)(2m+1)$. La figure 6.2 illustre le fonctionnement de ce filtre. Considérons les coordonnées y et y_s de deux pixels symétriques dans le masque par rapport au point à filtrer, de coordonné x. La valeur $v_m(x)$ prise en compte pour le calcul de la moyenne est sélectionnée en utilisant les conditions suivantes :

$$v_m(x) = \begin{cases} v(y) & si\ |v(x)-v(y)| < |v(x)-v(y_s)|, \\ v(y_s) & si\ |v(x)-v(y)| > |v(x)-v(y_s)|, \\ v(x) & sinon \end{cases} \tag{6.19}$$

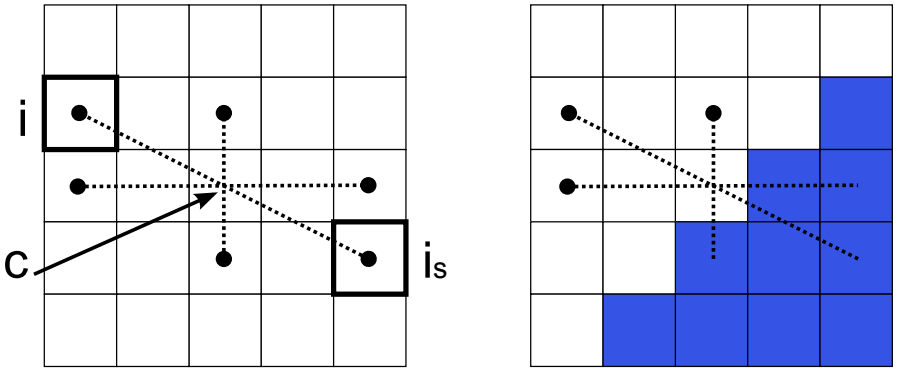


FIGURE 6.2 – Illustration du fonctionnent du filtre SNN, i et i_s sont deux pixels symétriques par rapport au pixel central c, les points sélectionnés sont les points dont la similarité d'intensité avec le pixel central sont les plus importantes, sur la figure de droite les pixels sélectionnés sont mis en évidence par un point noir.

Dans [90], Lee propose aussi un filtre adaptatif utilisant des poids binaires. Le critère de sélection pour le choix des intensités des pixels introduits dans le calcul de la moyenne est dépendant de l'écart type σ du signal dans le masque de filtrage. Lee propose de sélectionner les pixels dont la valeur de l'intensité est comprise dans un intervalle σ comparé à la luminance du pixel traité dans une fenêtre $\beta_\rho(x)$ autour de la position x du point traité et de taille $(2m+1)\times(2m+1)$. L'équation 6.20 décrit le fonctionnement de ce filtre. Cette méthode permet de préserver les contours des objets tout en filtrant les zones homogènes des images.

$$\hat{u}(x) = \frac{1}{C(x)} \int_{\beta_\rho(x)} \delta_y Y(y) dy \tag{6.20}$$

sa formulation discrète est la suivante :

$$\hat{u}(x) = \frac{1}{C(x)} \sum_{\beta_\rho(x)} \delta_y Y(y) \tag{6.21}$$

avec δ_y tels que :

$$\delta_y = \begin{cases} 1 & si\ |Y(x) - Y(y)| \leq \sigma, \\ 0 & si\ |Y(x) - Y(y)| > \sigma \end{cases} \tag{6.22}$$

et $C(x)$ est une fonction de normalisation.

Dans [91], Aurich et Weule introduisent le filtre bilatéral sous le nom de "filtre gaussien non-linéaire". Ce filtre est ensuite une nouvelle fois proposé dans [88] par Smith et Brady. Il est redécouvert dans [16], par Tomasi et Manduchi, qui formalisent l'expression et le comportement de ce filtre et le nomment le "filtre bilatéral". Le filtrage bilatéral est un filtrage local adaptatif, dans une fenêtre $\beta_\rho(x)$ autour de la position x du point traité. Les poids des intensités des pixels dans le calcul de la moyenne varient à la fois en fonction de la similarité de la distance dans le plan de l'image et de la similarité de l'intensité des pixels par rapport au point traité. Une fonction gaussienne est généralement utilisée pour le calcul de ces similarités. L'expression de ce filtre est présentée dans l'équation 6.23, dans laquelle on utilise pour le calcul des poids une fonction gaussienne.

$$\hat{u}(x) = \frac{1}{C(x)} \int_{\beta_\rho(x)} \exp\left(-\frac{|x-y|^2}{\rho^2}\right) \exp\left(-\frac{|Y(x)-Y(y)|^2}{h^2}\right) Y(y) dy \tag{6.23}$$

et sa formulation discrète :

$$\hat{u} = \frac{1}{C(x)} \sum_{\beta_\rho(x)} \exp\left(-\frac{|x-y|^2}{\rho^2}\right) \exp\left(-\frac{|Y(x)-Y(y)|^2}{h^2}\right) Y(y) \tag{6.24}$$

6.2.4 Minimisation de l'énergie : équations aux dérivées partielles

Les équations aux dérivées partielles (EDP) apparaissent naturellement dans de nombreux domaines de la physique. Un exemple classique est l'équation de la chaleur. Appliquées au traitement des images, les EDP apparaissent dans le cadres d'analyses multi-échelles. Ces équations sont essentielles dans le cadre de problèmes variationnels ou l'on cherche une image minimisant une certaine fonction d'énergie du type :

$$E(u) = \int_\Omega \phi(\|\nabla u\|) d\Omega \tag{6.25}$$

où ∇u représente le gradient de u et ϕ est une fonction calculant la norme. En utilisant une norme L_2 on obtient une énergie de type Tikonov [92], tandis qu'en utilisant une norme L_1 on obtient un problème de variation totale introduite par Rudin, Osher et Fatemi dans [93], après les travaux importants de Malik et Perona dans [94] sur l'introduction des équations de diffusion non-linéaire. La minimisation de cette énergie peut être faite en résolvant les équations d'Euler-Lagrange :

$$\begin{cases} \frac{\partial u}{\partial t} = div\left(\frac{\phi'(\|\nabla u\|)}{\|\nabla u\|}.\nabla u\right) \\ u_t = u_{bruitée} \end{cases} \tag{6.26}$$

où t représente le pas de discrétisation de l'équation différentielle. L'ajout d'une contrainte relative au terme d'attache des données de l'approche bayésienne est également possible :

$$\int_\Omega \left(Y(x) - u(x)\right)^2 d\Omega \leq \sigma^2 \tag{6.27}$$

où σ^2 représente la variance du bruit. On pourra trouver un panorama des méthodes récentes de débruitage s'appuyant sur les équations aux dérivées partielles dans la thèse de D.Tschumperlé [95].

6.2.5 Méthodes par transformées en ondelettes

La décomposition en ondelettes vise à supprimer de manière plus fine les fréquences du signal correspondantes à du bruit tout en prenant en compte les aspects des détails locaux dans l'image. Le principe de la décomposition en ondelettes est une représentation de l'image à plusieurs échelles permettant une analyse fréquentielle localisée (espace-échelle). En traitement du signal et en traitement d'image en particulier, la notion d'échelle est primordiale. La transformée en ondelettes se révèle être une méthode d'analyse idéale. Une fois que la projection dans la base d'ondelettes est réalisée, plusieurs opérations sont possibles. En particulier, le seuillage des coefficients de la transformée en ondelettes introduit par Donoho dans [96] constitue une méthode efficace pour régulariser les signaux. De manière plus générale l'analyse de ces coefficients permet de discriminer dans l'espace position-échelle, les composantes de l'image associées au bruit des composantes liées au signal originel. La transformée en ondelettes étant inversible, on peut alors obtenir une estimation non bruitée du signal d'entrée.

6.2.6 Exploration des redondances dans les images

Dans [97], J.S. De Bonet propose de restaurer les images bruitées en exploitant la redondance des informations des pixels. Il considère que dans les images naturelles il existe de nombreuses régions ayant des structures similaires. L'exploitation de ces similarités va permettre d'extraire de manière plus efficace le signal utile du signal du bruit. Le principe est de moyenner les régions similaires en utilisant une mesure de redondance adaptée [98, 99]. Dans [100], Wang et al. utilisent et approfondissent ce principe pour éliminer le bruit impulsionnel dans les images. Wang et al. proposent de remplacer le pixel central d'un motif considéré bruité par le pixel central d'un motif similaire considéré non-bruité. Cette démarche est à nouveau approfondie par D.Zhang et al. dans [101] et généralisée aux bruits additifs gaussiens. Dans [102], Criminisi et Perez exploitent la notion de redondance des pixels dans l'image pour estimer les valeurs des données manquantes ou bien pour éliminer des objets de l'image. Enfin Buades et al. dans [103], formalisent cette idée et décrivent une méthode complète de restauration des images. Ils proposent un filtrage des « moyennes non-locales », formulé de la manière suivante :

$$\hat{u}(x) = \frac{1}{C(x)} \int e^{\frac{-(G_a * |Y(x)-Y(y)|^2)}{h^2}} Y(y) dy \tag{6.28}$$

où $C(x) = \int e^{\frac{-(G_a * |Y(x)-Y(z)|^2)}{h^2}}$ est un facteur de normalisation, et G_a est une fonction gaussienne d'écart type a. Une description discrète du filtre à moyenne non-locale est donnée dans l'équation suivante :

$$\hat{u}(x) = \sum_{y \in \Omega} w(x,y) Y(y) \tag{6.29}$$

où w(x,y)$_{\beta_\rho(i)}$ est une famille de poids correspondant à la mesure de similarité entre les pixels dans un voisinage $\beta_\rho(i)$ autour du point de coordonnée i, les valeurs des pixels dans ce voisinage forment un vecteur N_i. Enfin, les poids de cette moyenne sont calculés par $w(x,y) = \frac{1}{Z_x} e^{\frac{-|Y(N_x)-Y(N_y)|^2}{h^2}}$, avec $Z_x = \sum_y e^{\frac{-|Y(N_x)-Y(N_y)|^2}{h^2}}$.

Buades et al. proposent d'utiliser le filtrage non-local en association avec d'autres méthodes existantes. Ils proposent de restaurer préalablement les motifs par un seuillage des coefficients de la transformée en ondelette de l'image (voir [104]). Dans la plupart des cas, l'utilisation des pixels de toute l'image n'est pas nécessaire et Buades et al. proposent de restreindre cette recherche à un voisinage local (voir [104]). La taille de ce voisinage a une influence sur la qualité du filtrage. Dans la plupart des cas, ce choix ainsi que le choix de la sélection du paramètre de similarité photométrique de la fonction de pondération des noyaux dans la moyenne est réalisé à l'aide d'heuristiques rendant la mise en œuvre de ces méthodes délicate.

6.3 Evaluation de la qualité de restauration des filtres adaptatifs

Dans cette thèse, nous nous intéressons aux applications de restaurations des images pour la photographie et la vidéo embarquée (appareil-photos/vidéos compacts, téléphone mobiles, PDA). Les processeurs multimédias utilisés dans ces systèmes sont limités en termes de puissances de calculs et de mémoire. Ces contraintes matériels, ne sont pas favorables à l'utilisation d'algorithmes itératifs et/ou nécessitant la mémorisation entière ou d'une partie de l'image. L'utilisation des filtres à voisinage locaux apparait comme une solution appropriée. Ces algorithmes sont basés sur des principes simples et généralement peu couteux en nombre d'opérations de calculs. Ils opèrent dans des masques carrés, de taille $(m+1) \times (m+1)$, permettant un filtrage de l'image à la volée. Nous allons maintenant nous intéresser aux performances de restauration des images bruitées des algorithmes de filtrage à voisinage local (voir section 6.2.3).

Nous étudions les filtres décrits dans la section 6.2.3. Pour évaluer les performances des filtres nous utilisons un modèle de bruit blanc gaussien additif avec différentes valeurs de variances et les 24 images de l'annexe D. Pour mesurer les qualités d'images produites nous utilisons la mesure objective du RMS et la mesure subjective de l'appréciation visuelle. La figure 6.3 montre les courbes de RMS obtenues en fonction de la variance

du bruit blanc gaussien. La droite noire nommée « Noise », représente le RMS des images bruitées. On rappelle qu'un filtre possède de bonnes propriétés de filtrage du bruit lorsque sa courbe de RMS est située en dessous de la droite de RMS du bruit. Ce comportement montre que le filtre élimine plus le bruit qu'il ne dégrade les détails dans l'image. Nous cherchons le filtre possédant les meilleures propriétés combinées de filtrage du bruit et de conservation des détails. Sur un intervalle d'écart type de bruit donné, on cherche donc le filtre f possédant la plus petite valeur d'intégrale $\int_\sigma \text{Noise}(x) - f(x)dx$, où σ représente l'écart type du bruit. Tous les filtres sont appliqués avec des masques de taille 5×5. Les résultats de la figure 6.3 montrent que le filtre médian possède les plus mauvaises performances de filtrage du bruit gaussien parmi les filtres étudiés. Les filtres SNN et Nagao possèdent des propriétés de filtrage légèrement supérieures pour des bruits d'écarts types forts, mais ils dégradent de manière importante les images lorsque l'écart type du bruit diminue. Le filtre gaussien montre de meilleures performances de filtrage par rapport à ces trois filtres sur l'intervalle d'écart type de bruit étudié. Le filtre sigma possède de très bonnes performances pour des bruits de faibles puissances mais sa qualité de filtrage se dégrade rapidement lorsque l'écart type du bruit augmente. D'autre part, le filtre GIWO possède de bonnes qualités de filtrage pour des bruits faibles, équivalentes au filtre sigma et permet une meilleure stabilité de filtrage lorsque la puissance du bruit augmente. Enfin le filtre bilatéral apparaît comme le filtre le plus robuste. Il possède de bonnes propriétés de filtrage pour des écarts types de bruits faibles et une dégradation de la qualité du filtrage beaucoup moins importante que les filtres GIWO et sigma avec l'augmentation de la puissance du bruit.

La figure 6.4, montre les résultats des filtrages sur une image de test. L'image est bruitée avec un bruit blanc gaussien additif d'écart type 15. Ces images montrent que le filtre médian ne réduit pas le bruit efficacement, il crée des tâches dans l'image et introduit du flou. Le filtre SNN introduit du flou et un effet de granularité très important. Nous voyons que le filtre de Nagao crée des zones structurées dans l'image, assimilables à un effet de mosaïque. Le filtre sigma préserve bien les contours mais ne réduit pas efficacement le bruit. Le filtre gaussien rend l'image flou et détruit les contours. D'autre part, le filtre GIWO permet de réduire le bruit tout en conservant les détails de l'image, il introduit du flou et un effet de grain. Finalement, le filtre bilatéral permet un meilleur lissage du bruit que le filtre GIWO sans dégrader plus les contours, il diminue aussi l'effet de granularité. Les résultats de l'appréciation visuelle confirment les résultats obtenus avec les mesures du RMS. De manière générale le filtre bilatéral permet de réduire efficacement de bruit tout en préservant les contours, c'est une propriété essentielle de qualité d'image pour la photographie et la vidéo numérique.

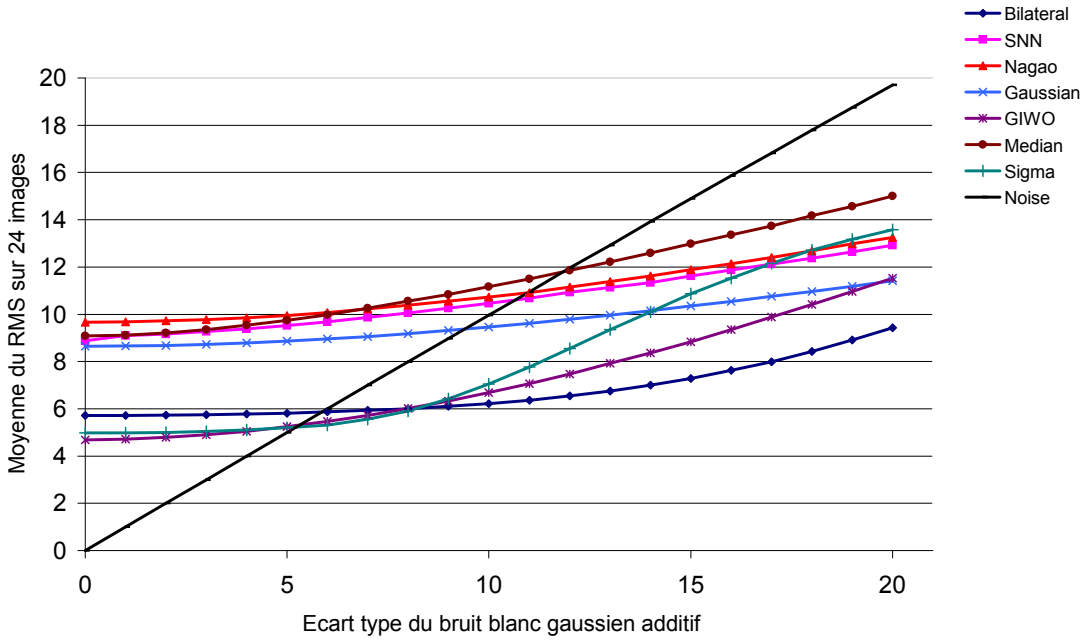


FIGURE 6.3 – Mesure du RMS des filtres à voisinage local étudiés en fonction de l'écart type du bruit blanc gaussien ajouté sur l'image. On rappelle qu'un filtre possède une bonne qualité de filtrage du bruit lorsque sa courbe de RMS est située en dessous de la droite « Noise » représentant le bruit.

6.3.1 Discussion

L'état de l'art dressé dans ce chapitre nous à permis d'aborder les principales solutions de restauration des images bruitées. Pour des raisons de limitations matériels, nous nous sommes principalement intéressé aux méthodes de débruitage à voisinage local. L'étude menée sur les performances des filtres à voisinage local, montre que le filtre bilatéral permet d'obtenir le meilleur compromis entre le filtrage du bruit et la conservation des détails de l'image. De plus, contrairement aux autres filtres étudiés, le filtre bilatéral est paramétrable. Pour une fenêtre de taille fixe, le paramétrage de l'écart type h (voir équation 6.23) de la fonction gaussienne des poids d'intensités permet de régler la similarité photométrique du filtre. Le comportement du filtre peut ainsi être modifié de manière importante, en passant d'un filtre identité, lorsque h tend vers 0 à un filtre gaussien, lorsque h tend vers l'infini. Cette propriété remarquable permet d'envisager d'adapter la puissance du filtrage en fonction de la puissance du bruit dans l'image. C'est un atout très intéressant pour les applications de réduction du bruit en photographie et en vidéo numérique dans lesquelles la puissance du bruit varie constamment en fonction des paramètres de prises de vues (température, luminosité, etc.). On peut supposer que pour chaque puissance de bruit dans l'image, il existe un h optimal que l'on peut calibrer. Pour appliquer ce h optimal, il faut donc être capable d'estimer la puissance du bruit dans l'image. C'est l'objet du chapitre 7.

(a) Originale (b) Bruit gaussien $\sigma = 15$ (c) Médian

(d) SNN (e) Nagao (f) Sigma

(g) Gaussien (h) GIWO (i) Bilateral

FIGURE 6.4 – Comparaison des qualités d'images produites par les différents filtres à voisinage local étudiés.

Chapitre 7

Bruit : filtrage bilatéral adaptatif et application sur la mosaïque de Bayer

Dans les appareils de captures d'images numériques, la quantité de bruit présente dans les images varie constamment et de manière importante en fonction de l'éclairement de la scène. On peut ainsi passer d'images très bruitées à des images très propres. Il est donc important d'avoir à disposition un filtre capable d'adapter la force de son filtrage en fonction de la quantité de bruit dans l'image. La difficulté de discriminer le bruit des détails de l'image est un problème perdurant en traitement d'image. Les filtres adaptatifs comme le filtre bilatéral permettent d'adapter leurs filtrages en fonction des contours et de la structure de ce qu'ils « croîent » percevoir comme des « objets ». Ils ne sont pas capable de s'adapter sans interventions extérieures à la quantité de bruit dans l'image. Ils auront donc le même comportement dans une image bruitée et dans une image sans bruit, pouvant ainsi dégrader des détails existants dans une image propre et ne pas filtrer suffisamment une image contenant des bruits forts. Le filtre bilatéral a la propriété intéressante de posséder un paramètre de similarité photométrique ajustable. En établissant une corrélation entre la quantité de bruit dans l'image et le paramètre de similarité photométrique, il serait possible d'obtenir un comportement de filtre performant. Cela permettrait d'avoir un comportement de filtre idéal, ne filtrant pas ou peu les images propres et filtrant le bruit et les détails de même variance pour optimiser la qualité des images bruitées. Dans ce chapitre nous proposons un filtre bilatéral adaptatif dans le cas d'un bruit de photons, par estimation du nombre de photons ayant percuté le capteur. Nous proposons ensuite une adaptation du filtre bilatéral pour la mosaïque de Bayer, permettant d'appliquer la méthode proposée au filtrage des images en couleurs dans les systèmes à un seul capteur.

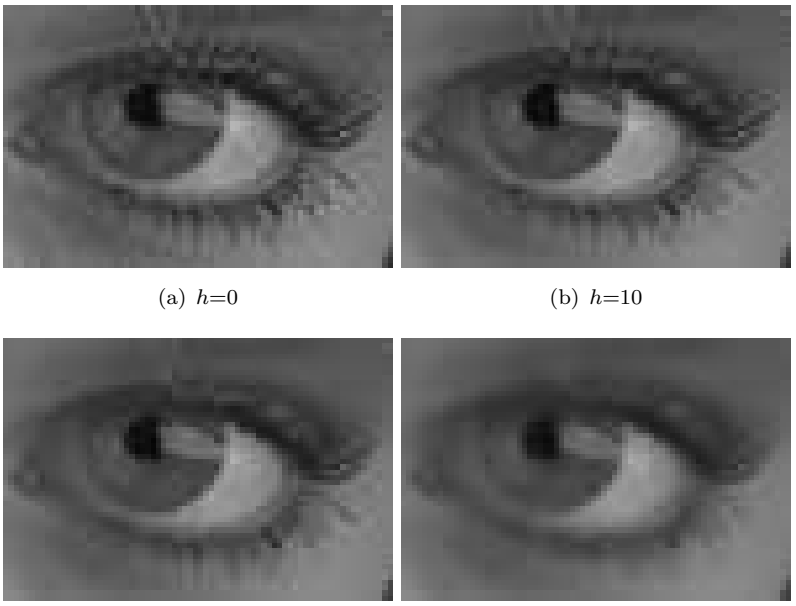
7.1 Rappel des caractéristiques du filtre bilatéral

Le filtre bilatéral [16] possède une place importante dans la littérature de restauration des images bruitées. Il permet de filtrer efficacement le bruit dans les zones d'intensités uniformes, tout en préservant les contours et les structures des objets. Il possède une efficacité démontrée et sa formulation simple contribue à sa popularité [105]. L'intensité d'un pixel filtré est remplacé par la moyenne des combinaisons non-linéaires des intensités des pixels dans un masque carré, centré sur le pixel traité et de taille $(n+1)\times(n+1)$. Les poids utilisés dans le calcul de la moyenne varient en fonction des similarités photométriques et de la position des pixels dans le masque par rapport au pixel courant. La version la plus populaire utilise des poids gaussiens, comme dans l'expression discrétisée suivante :

$$\hat{u}(x) = \frac{1}{C(x)} \sum_{\beta_\rho} \exp -\frac{|x-y|^2}{\rho^2} \exp -\frac{|Y(x)-Y(y)|^2}{h^2} u(y) \tag{7.1}$$

où β_ρ représente la fenêtre de convolution, $C(x)$ est un coefficient de normalisation, y est un ensemble de coordonnées $2D$ dans le masque et x est la position $2D$ d'un pixel dans le plan de l'image. $Y(x)$ représente l'intensité du pixel à la position x dans l'image bruitée. $\hat{u}(x)$ est la valeur estimée du pixel à la position x, ρ et h sont respectivement la déviation standard de la distribution gaussienne des poids géométriques et la déviation standard de la distribution gaussienne des poids d'intensités. Dans la pratique, le paramètre ρ est fixé en fonction de la taille du masque. Le paramètre h permet alors de régler le paramètre de similarité photométrique du filtre. Le réglage de ce paramètre doit prendre en compte le type d'application pour laquelle le filtre est utilisé. Dans le cas particulier de la restauration des images bruitées, ce paramètre doit varier en fonction de la quantité de bruit présente dans l'image, les propriétés intrinsèquement adaptatives du filtre bilatéral lui permettrons ensuite de s'adapter aux détails de l'image. La figure 7.1 illustre le comportement du filtrage bilatéral pour différentes valeurs de h. On peut voir que plus la valeur de h est importante, plus le filtre est passe-bas. Dans les applications de photographie et de vidéos numériques, la quantité de bruit dans les images varie constamment en fonction des paramètres de prises de vues : luminosité, ISO, température etc. Il est donc intéressant de pouvoir ajuster de manière automatique le paramètre h de similarité photométrique du filtre en fonction de la puissance du bruit. Cela permettrait d'optimiser le compromis entre préservation des détails et le filtrage du bruit. Idéalement le filtre devrait filtrer peu ou pas les images faiblement bruitées pour conserver les détails des images. En présence de bruits plus fort, le filtre devrait filtrer le bruit en contrepartie de la perte des structures de la scène possédant des variances

inférieures ou égales à la variance du bruit. Pour ce faire, il faut être capable d'estimer la quantité de bruit dans l'image.



(a) h=0 (b) h=10

(c) h=20 (d) h=40

FIGURE 7.1 – Images obtenues par filtrage bilatéral en utilisant un masque β_ρ carré de taille 5×5 et un paramètre h variable. Plus la valeur de h est importante, plus la similarité photométrique du filtre est faible et plus l'image est floue.

7.2 Modèle de bruit de capteurs

Pour estimer efficacement la quantité de bruit présente dans les images, il est indispensable de disposer d'un modèle de bruit réaliste. On utilise le modèle proposé par Costantini et al. dans [66] (voir section 6.1), on rappelle la formulation de ce modèle :

$$Y_G = G_a.[P_{poiss}(F) + G_{PRNU}.P_{poiss}(u)] + Z \tag{7.2}$$

où u est l'image de la scène, Y_G est l'image bruitée, G_a, F, G_{PRNU} et Z modélisent respectivement la non-uniformité d'amplification du gain, le courant d'obscurité, la non-uniformité de la sensibilité des photorécepteurs et les bruits de lecture. Nous avons vu dans la partie 6.1, que ce modèle peut s'exprimer comme la somme d'un signal bruité dépendant de l'image $Y_{Gsignal}$ et d'un signal bruité dépendant de la température Y_{Gtemp} :

$$Y_{Gsignal} = G_a.G_{PRNU}.P_{poiss}(u) \tag{7.3}$$

et

$$Y_{Gtemp} = G_a.P_{poiss}(F) + Z \tag{7.4}$$

Dans les conditions standards de température et de luminosité, on considère que la composante de bruit Y_{Gtemp} est négligeable par rapport à la composante $Y_{Gsignal}$. Le bruit dans l'image est alors majoritairement poissonnien [72]. De plus, on peut considérer que les amplitudes de variations des gains G_a et G_{PRNU} sont très faibles comparées aux variations des bruits de photons, de telle sorte que G_a et G_{PRNU} peuvent être considérés comme un seul gain uniforme noté G, on en déduit la nouvelle expression de l'image bruitée Y_G, tels que : $Y_G = G.P_{poiss}(u)$, en notant $Y = P_{poiss}(u)$, on obtient la nouvelle expression de l'image Y_G, en fonction du gain et de l'image bruitée par un bruit de photon : $Y_G = G \times Y$.

7.3 Estimation du meilleur paramètre h

Dans cette section, nous proposons une méthode d'estimation de la valeur du paramètre h permettant un filtrage bilatéral adaptatif et optimal du bruit. Pour estimer le bruit, nous utilisons ses propriétés statistiques poissonniennes et les propriétés du processus de contrôle de gain utilisé en photographie et en vidéo numérique.

7.3.1 Le contrôle automatique de gain

Dans les systèmes de captures d'images numériques, un contrôle automatique de gain est nécessaire pour pouvoir photographier des scènes sous différentes intensités d'illuminations. Le contrôle de gain consiste à ajuster l'intensité moyenne de l'image de sortie du capteur, comme cela est illustré sur la figure 7.2. En supposant l'accès disponible au facteur de gain il est possible d'estimer le nombre de photons ayant percuté le capteur. Connaissant la résolution du capteur on peut estimer $d_{h\nu}$ la densité de photons par pixel et par conséquent la puissance du bruit dans l'image. Cependant, l'accès à l'information de gain n'est généralement pas disponible pour les photographes, même avec l'utilisation d'un format brut (format RAW). Dans les sections qui suivent, nous proposons une méthode d'estimation du facteur de gain basée sur les propriétés poissonniennes de distribution du signal de l'image.

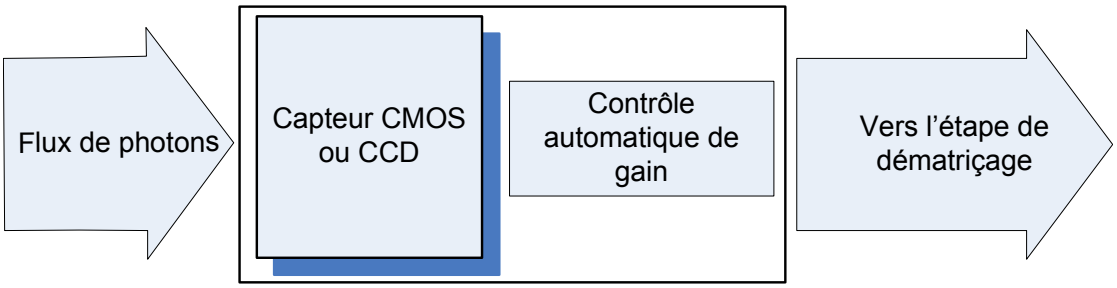


FIGURE 7.2 – Contrôle du gain dans un capteur numérique : le gain est ajusté en fonction de l'éclairement de la scène.

7.3.2 Densité de photons à partir des statistiques de l'image

En présence de manipulation de gain uniquement, on suppose que la variance du bruit est proportionnelle à la moyenne des intensités des pixels dans les régions à valeurs d'intensités constantes ou régions non-structurées. Cela est dû à la statistique de Poisson régissant l'arrivée des photons sur la surface du capteur. Considérons Y l'image délivrée par le capteur et Y_G l'image dont l'intensité moyenne est ajustée par un contrôle de gain. Soit G le facteur de gain, nous pouvons écrire :

$$Y_G = G \times Y \tag{7.5}$$

Considérons maintenant le processus de distribution de Poisson de l'image Y, nous pouvons supposer la relation suivante, dans laquelle $T^-(Y)$ représente une région à valeurs d'intensités constantes :

$$E(T^-(Y)) = Var(T^-(Y)) \tag{7.6}$$

où E est la moyenne et Var est la variance. De la même manière, à partir de l'équation 7.5, on peut écrire :

$$E(T^-(Y_G)) = G \times E(T^-(Y)) \tag{7.7}$$

d'autre part, la variance de l'image Y_G peut s'écrire :

$$Var(T^-(Y_G)) = E(T^-(Y_G)^2) - E(T^-(Y_G))^2 \tag{7.8}$$

en utilisant l'expression de l'équation 7.7 dans l'équation 7.8, on peut écrire la relation suivante :

$$Var(T^{-}(Y_G)) = G^2 \times Var(T^{-}(Y)) \tag{7.9}$$

connaissant les expressions des équations 7.6 et 7.7 on peut écrire :

$$Var(T^{-}(Y)) = G \times E(T^{-}(Y_G)) \tag{7.10}$$

en replaçant cette expression dans l'équation 7.9, on obtient l'expression du gain G, tels que :

$$G = \frac{Var(T^{-}(Y_G))}{E(T^{-}(Y_G))} \tag{7.11}$$

La moyenne de l'image $E(T^{-}(Y_G))$ et sa variance $Var(T^{-}(Y_G))$ peuvent être calculées simultanément pour chaque point dans l'image Y_G en utilisant une fenêtre de convolution efficace [106]. Nous cherchons à établir une corrélation entre tout les $E(Y_G)$ et les $Var(Y_G)$ dans les zones d'intensités constantes T^{-}. Tant que la moyenne et la variance sont supposées être proportionnelles, une corrélation robuste par la méthode des moindres carrés peut être appliquée. Le coefficient de la pente correspondant pourra être interprété comme le facteur de gain transformant le nombre de photons en une intensité de pixel. Pour obtenir une plus grande précision de mesure de la variance du bruit, nous calculons celle-ci sur l'image de bruit estimée $\text{Noise}_{\text{estim}} = Y_G - G_1 \star Y$ ou G_1 est une convolution gaussienne de variance 4. Cette opération permet d'éliminer les variations locales de faibles gradients dans l'image de la scène pouvant perturber l'estimation de la variance du bruit.

7.3.3 Estimation du meilleur h à partir de la densité de photons

Pour pouvoir estimer le gain, nous avons besoin de sélectionner des zones $T^{-}(Y_G)$ à valeurs d'intensités constantes dans l'image bruitée Y_G correspondante. Ces zones peuvent être facilement estimées en utilisant l'image de gradient déterminée de la manière suivante :

$$T^{-}(Y_G) = \varepsilon_w(\nabla(G_\sigma \star Y_G) > \vartheta) \tag{7.12}$$

où Y_G est l'image après le contrôle du gain, G_σ est une convolution gaussienne de variance σ, ε_w est l'érosion morphologique dans une fenêtre de taille $w \times w$, ϑ est un paramètre d'intensité. Dans nos expérimentations le gain peut être précisément estimé en échantillonnant des points $Var(Y)$ et $E(Y)$ sur 10^4 positions de pixels. Dans l'expression de l'équation 7.12 on choisit $\sigma = 2$, $\vartheta = 2$ et $w = 11$.

7.3.4 Calibrage de h en fonction de la densité de photons par pixel $d_{h\nu}$

Comme nous l'avons présenté plus haut, les propriétés de similarités photométriques du filtre bilatéral varient en fonction du paramètre h. Plus la valeur de h est importante plus la similarité photométrique du filtre est faible. Pour obtenir un filtrage bilatéral optimal, il est nécessaire d'adapter la valeur de h en fonction de la puissance du bruit. Nous avons décrit une méthode permettant d'estimer la densité de photons moyenne par pixel $d_{h\nu}$ à partir du gain G appliqué sur l'image. Une calibration de la valeur optimale de h en fonction de $d_{h\nu}$ est nécessaire pour rendre le filtre bilatéral adaptatif. Ceci peut être fait en utilisant un modèle de formation de l'image ou expérimentalement en calibrant les capteurs pour différentes conditions d'illuminations.

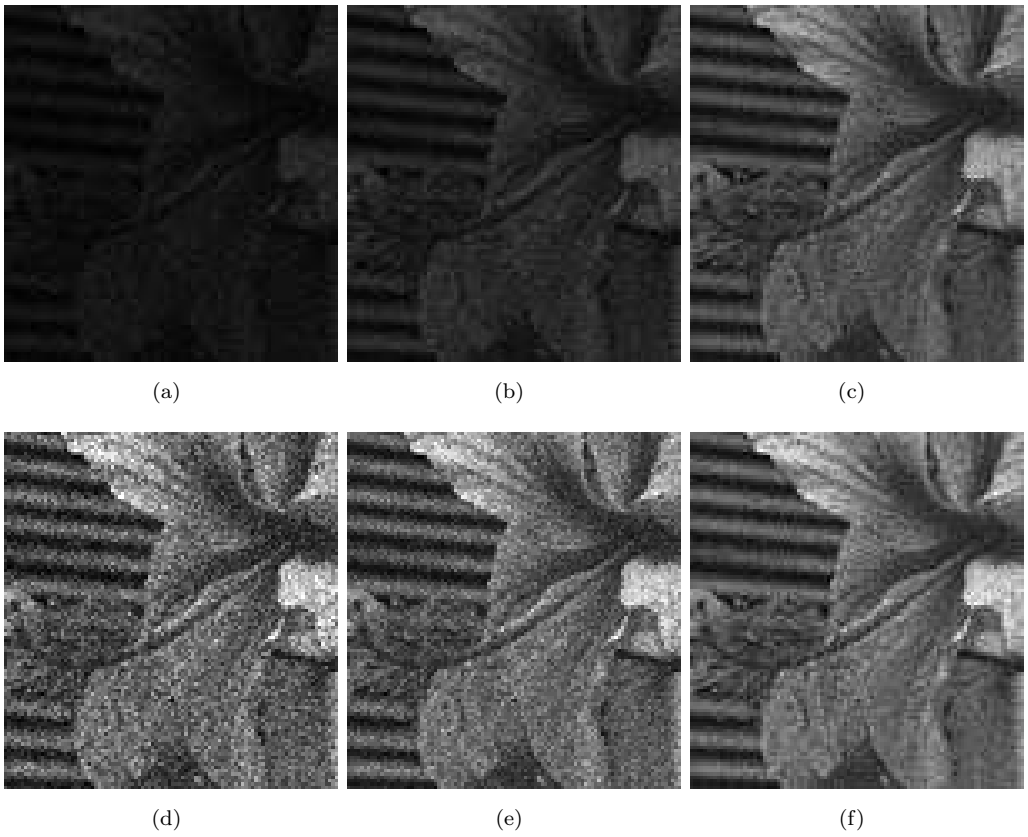
7.4 Expérimentations et résultats

7.4.1 Simulation du bruit de photons

Nous simulons l'acquisition d'images par un capteur numérique en utilisant un processus de Poisson cumulatif par régions [107]. Des images représentant la scène de prise de vue sont utilisées comme fonctions de distributions de probabilités. On simule ainsi l'enregistrement individuel des photons pour chaque positionnement de pixel dans un processus de Monte-Carlo. Comme images de scènes, nous utilisons les images de l'annexe D. Notons qu'il n'y a pas d'alternatives simples à cette simulation. Le bruit de Poisson résultant n'est ni additif ni multiplicatif et il ne peut pas être précisément modélisé avec une distribution plus simple, spécialement pour des bruits forts, dans les zones de lumière basse. La figure 7.3 illustre la simulation du bruit proposée pour des densités de photons par pixel égales à 20, 80 et 180 et les images correspondantes avec une adaptation du gain.

7.4.2 Sélection des zones à valeurs d'intensités constantes

Pour pouvoir estimer le nombre de photons de manière précise, nous avons vu qu'il est nécessaire de sélectionner des zones à valeurs d'intensités constantes $T^{-}(.)$ dans l'image bruitée Y_G. On illustre ici le fonctionnement de la méthode proposée dans la section 7.3.3. La figure 7.4(a), montre l'image bruitée Y_G possédant une densité moyenne de 60 photons par pixel, la figure 7.4(b) montre l'image d'estimation du bruit $\text{Noise}_{\text{estim}}$ et enfin la figure 7.4(c) montre le masque des zones sans textures $T^{-}(Y_G)$ obtenu.



(a) (b) (c)

(d) (e) (f)

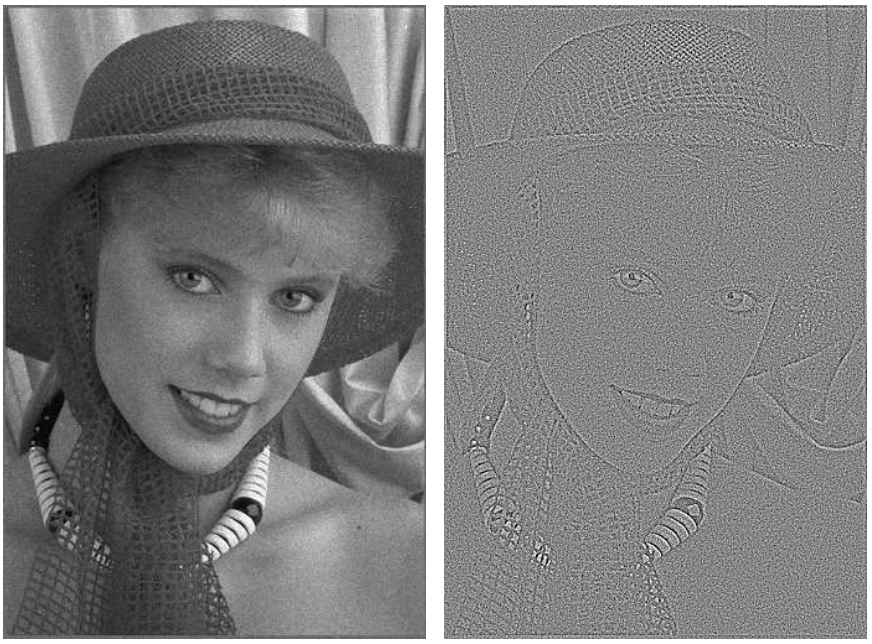
FIGURE 7.3 – Simulation de l'acquisition des photons sur un capteur par un processus de probabilité de Poisson. De (a) à (c), le nombre moyen de photons simulés par pixel est 20, 80, 160. De (d) à (f), on peut voir les images correspondantes avec un contrôle automatique de gain.

7.4.3 Estimation du nombre de photons

Pour évaluer les performances de la méthode d'estimation proposée, nous avons fait des essais pour un nombre de photons variant entre 10^4 et 10^9 en utilisant une image de résolution 512×341. Cela correspond à des densités de photons par pixels variant entre $d_{h\nu} = 0,05$ et $d_{h\nu} = 6000$, respectivement cela correspond à des variations d'une image extrêmement bruitée jusqu'à une image très propre. On estime le nombre de photons $\bar{\Phi}$ percutant le capteur avec la formule suivante :

$$\bar{\Phi}(Y) = \frac{\text{nbpix}(Y)\overline{E(Y)}}{G} = \frac{\sum_{(x,y)\in Y} Y(x,y)}{G}, \tag{7.13}$$

où $\text{nbpix}(Y)$ est le nombre total de pixels dans Y. G est le gain estimé, calculé avec la méthode proposée dans la section 7.3.2. La figure 7.5 montre la corrélation existante entre le nombre de photons estimés et le nombre de photons simulés. On peut voir que la corrélation est excellente. La droite tracée possède une pente strictement égale à 1.



(a) Image bruitée $d_{h\nu} = 60$ (b) Image d'estimation du bruit

(c) Masque des zones sans textures

FIGURE 7.4 – Illustration de la méthode de sélection des zones à valeurs d'intensités constantes.

Les trois derniers points à droite sont en dessous de la droite car on approche le niveau de bruit natif de l'image.

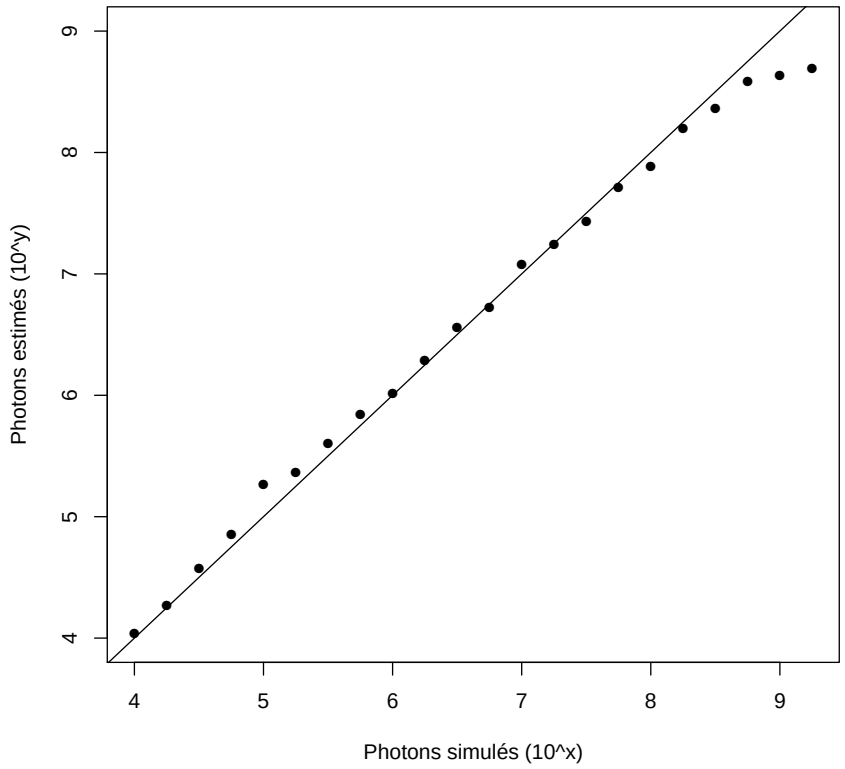


FIGURE 7.5 – Corrélation entre le nombre de photons estimés et le nombre de photons simulés.

7.4.4 Estimation du meilleur h

On utilise le modèle d'acquisition d'images de la section 7.4.1 pour calibrer la meilleure valeur de h en fonction de la densité de photons par pixel $d_{h\nu}$. On utilise un intervalle de densités de photon par pixel variant de 0,5 à 6000. Pour chaque densité de photons, on applique un filtrage bilatéral pour des valeurs de h variants de 0 à 100, par pas de 5. La figure 7.6, montre les courbes des résultats obtenus. La meilleure valeur de h pour une densité de photons $d_{h\nu}$ fixée est celle pour laquelle la valeur de RMS calculée est la plus petite. La calibration du meilleur h en fonction de la densité de photons par pixel $d_{h\nu}$ est donc donnée par l'enveloppe inférieure des courbes de la figure 7.6. Pour une meilleure lisibilité des résultats, la figure 7.7 montre un agrandissement des courbes de la figure 7.6 pour des valeurs de $d_{h\nu}$ variants entre 10 et 6000. En sélectionnant l'enveloppe inférieure de cette série de courbes, nous obtenons une relation donnant le meilleur paramètre de similarité photométrique h en fonction de la densité de photons simulés. On peut voir que pour des densités de photons par pixel inférieures à $d_{h\nu} = 10$, le meilleur paramètre de filtrage est toujours 100, ces bruits extrêmement forts ne sont pas réalistes en photographie numérique et nous ne les prenons pas en compte dans notre étude. La

figure 7.8 montre les résultats de la calibration pour des densités de photons variant entre 10 et 6000. On estime la meilleure courbe passant par les points obtenus en utilisant une équation polynomiale du second degré, tels que $h = 7,2718 \times X^2 - 161,76 \times X + 904,39$ avec $X = \frac{100}{\log 10(d_{h\nu} \times S)}$ ou S est le nombre de pixels dans l'image. Le coefficient de détermination de cette courbe estimée est $R^2 = 0,9911$.

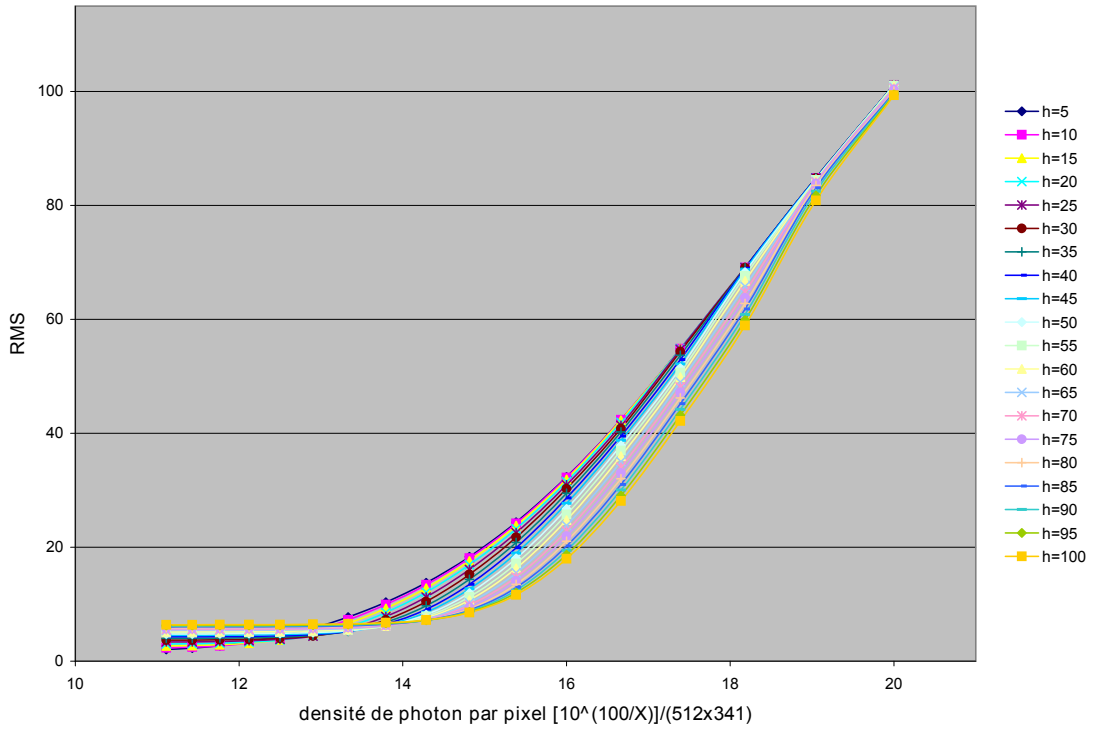


FIGURE 7.6 – Estimation de la meilleure valeur de h pour des densités de photons par pixel $d_{h\nu}$ variant de 0.5 à 6000.

7.4.5 Filtrage des images

Sur la figure 7.9, nous comparons la qualité des images filtrées avec le filtre bilatéral adaptatif proposé par rapport aux images filtrées avec le filtre bilatéral. Pour la comparaison, nous simulons l'acquisition d'images par un capteur numérique avec différentes valeurs de densités de photons $d_{h\nu}$ variant de 30 à 6000 photons par pixel. La première colonne de la figure montre les images bruitées. La deuxième colonne montre les images obtenues avec un filtrage bilatéral possédant un paramètre de similarité photométrique $h = 20$. La troisième colonne montre les images produites par un filtrage bilatéral possédant un paramètre de similarité photométrique $h = 100$. La quatrième colonne montre les images filtrées avec le filtre bilatéral adaptatif proposé. On constate que le filtre bilatéral lorsqu'il possède un paramètre h fixe ne s'adapte pas suffisamment à la puissance du bruit dans les images. En effet, une valeur de h faible permet de préserver les détails

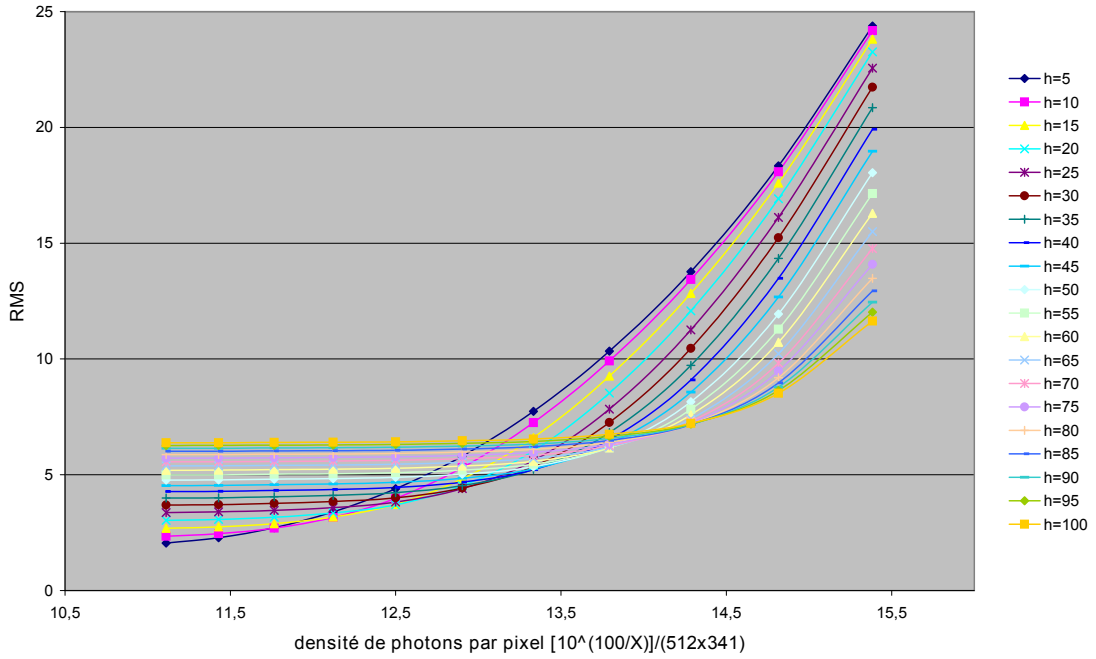


FIGURE 7.7 – Estimation de la meilleure valeur de h pour des densités de photons par pixel variant de 10 à 6000.

de l'image dans les conditions de bruit faible mais ne filtre pas assez le bruit lorsque sa variance augmente. Inversement, une valeur de h élevée permet de filtrer les bruits de variances élevées mais ne conserve pas les détails de l'image lorsque qu'ils ne sont pas dégradés par le bruit. Le filtre proposé adapte parfaitement son paramètre de similarité photométrique en fonction de la puissance du bruit. Il préserve les détails de l'image dans les conditions de faible bruit et filtre le bruit lorsque sa variance augmente permettant d'obtenir le meilleur compromis entre filtrage du bruit et préservation des détails.

7.5 Discussion

Dans ce chapitre, nous avons proposé une méthode pour le filtrage des images digitales dans le cas d'un bruit de photons. Nous avons montré qu'il est important d'adapter le niveau de filtrage au niveau de la puissance du bruit. De plus, nous avons montré comment le filtre bilatéral peut être optimisé si le niveau de bruit est connu. Nous avons proposé un algorithme d'estimation du niveau de bruit qui est simple et efficace. Cet algorithme montre une corrélation excellente entre la puissance de bruit estimée et la puissance de bruit réelle. En appliquant notre méthode de détection du niveau de bruit pour régler le paramètre h du filtre bilatéral, nous obtenons un nouveau filtre bilatéral adaptatif permettant d'obtenir le meilleur filtrage bilatéral pour différents niveaux de bruits de photons.

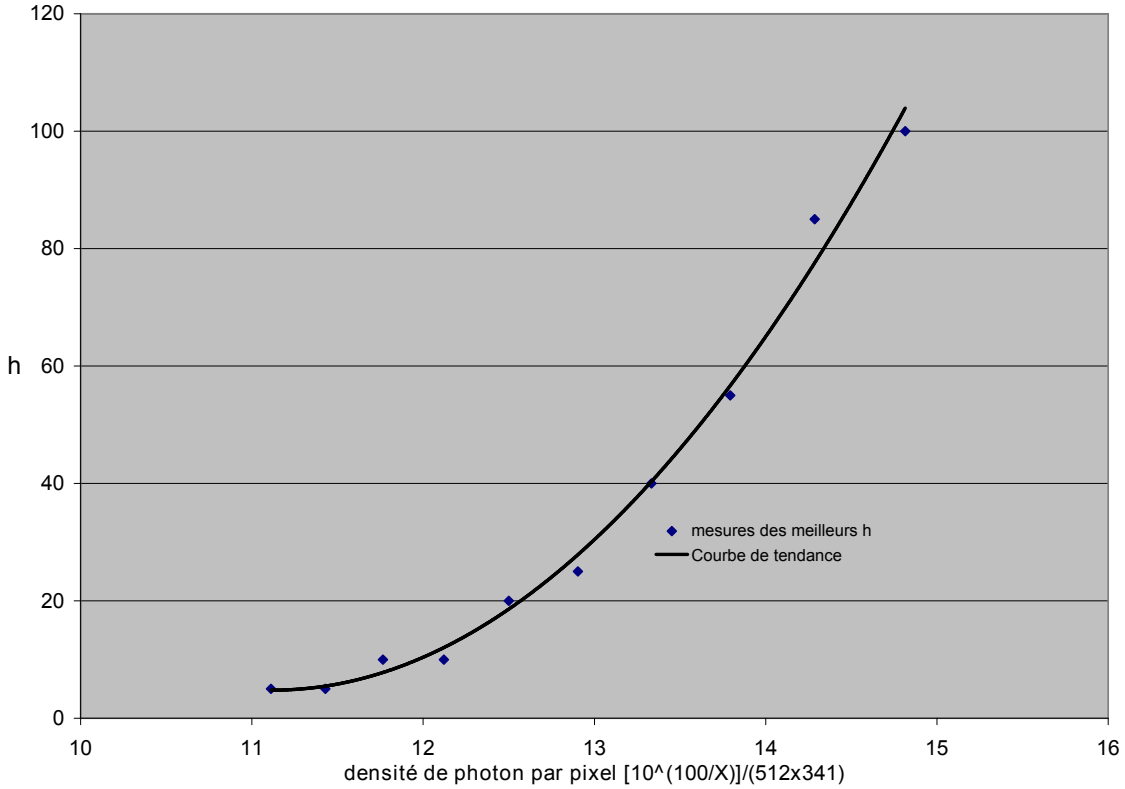
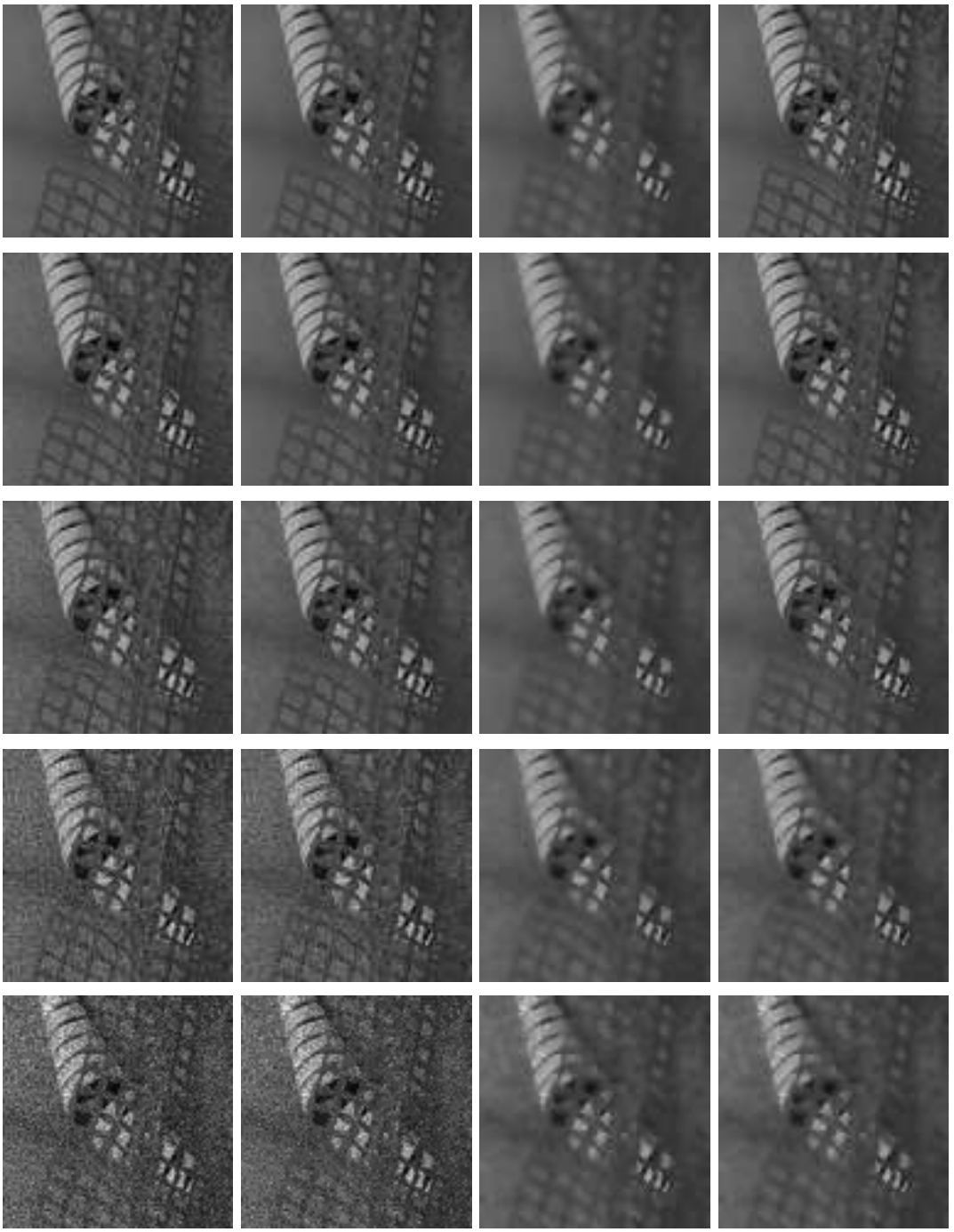


FIGURE 7.8 – Calibrage de la meilleure valeur de h pour des densités de photons par pixel variant de 10 à 6000.

Dans le cadre de l'application de ce filtre à des systèmes de captures d'images numériques en couleurs, il est nécessaire de prendre en compte la mosaïque colorée du capteur. Le filtre bilatéral tels qu'il est formulé n'est pas directement applicable sur la matrice de Bayer. Deux solutions apparaissent possibles, filtrer le bruit après l'étape de dématriçage ou reformuler le filtre bilatéral pour qu'il puisse filtrer la mosaïque de Bayer. On suppose que filtrer l'image avant l'étape de dématriçage permettra de mieux discriminer le signal utile du bruit et favorisera un meilleur filtrage. L'image de mosaïque permet d'avoir accès aux données bruts du capteur avant qu'elles ne soient dispersées à travers les différents canaux de couleurs par l'étape de dématriçage. D'autre part, on suppose que la présence de bruit dans l'image de mosaïque va perturber le fonctionnement de l'algorithme de dématriçage. Pour résumer, deux raisons nous incitent à formuler un filtrage bilatéral pour la matrice de Bayer : une meilleure élimination du bruit et une meilleure qualité de dématriçage. Ces deux étapes sont essentielles pour la qualité des images en photographie et en vidéos numériques.



(a) Images bruitées (b) Filtre bilatéral $h = 20$ (c) Filtre bilatéral $h = 100$ (d) Filtre bilatéral adaptatif

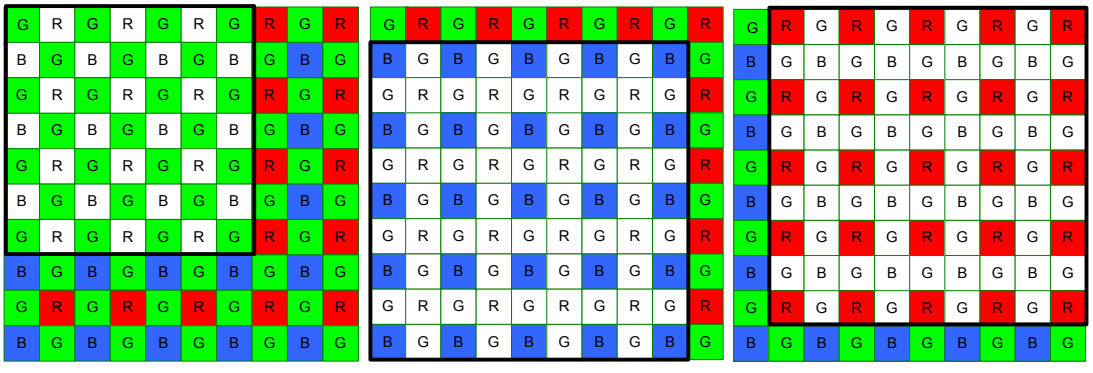
FIGURE 7.9 – Comparaison de la qualité du filtrage entre le filtre bilatéral adaptatif et le filtre bilatéral.

7.6 Formulation du filtre bilatéral pour la mosaïque de Bayer

Dans cette section, nous proposons une nouvelle formulation du filtre bilatéral adaptée pour la mosaïque de Bayer, on obtient un nouveau filtre, le filtre bilatéral Bayer. Nous proposons de filtrer chaque canal de couleur séparément. Étant donné la distribution spatiale des couleurs dans la mosaïque de Bayer, nous proposons d'utiliser des tailles de masques différentes en fonction des canaux traités. Pour le filtrage du canal vert, nous proposons d'utiliser une fenêtre de taille $(2n+1) \times (2n+1)$, soit un nombre $N_V = \frac{M_V \times M_V + 1}{2}$ de pixels verts filtrés, avec $M_V = 2n+1$. Pour les pixels bleus et rouges, nous proposons d'utiliser une fenêtre de taille $(2n+3) \times (2n+3)$, soit un nombre $N_{RB} = \frac{(M_{RB}+1)}{2} \times \frac{(M_{RB}+1)}{2}$ de pixels bleus et rouges filtrés, avec $M_{RB} = 2n+3$. La figure 7.10, illustre ces fenêtres dans le cas particulier où $m = 3$. Le filtre bilatéral Bayer est exprimé de la manière suivante :

$$\widehat{u}(x_c) = \frac{1}{C(x_c)} \sum_{\beta_c} \delta(c - c') exp \frac{-|x_c - y_{c'}|}{\rho^2} exp \frac{-|Y(x_c) - Y(y_{c'})|}{h^2} Y(y_{c'}) \qquad (7.14)$$

où, δ représente la fonction Dirac, c est la caractéristique de couleur du pixel courant, c' est la caractéristique de couleur d'un pixel dans la fenêtre de convolution et β_c est un masque de convolution carré, de taille variable selon la valeur de c, y est un ensemble de positions 2D, x est la position 2D centrale dans le masque de traitement, $Y(x)$ est l'intensité du pixel à la position x, $\widehat{u}(x)$ est l'estimation du pixel à la position x, ρ et h sont respectivement les écarts types des distributions gaussiennes des poids géométriques et des poids d'intensités.



(a) (b) (c)

FIGURE 7.10 – De (a) à (c), on présente les différents masques du filtre bilatéral Bayer pour les canaux vert, bleu et rouge. Dans ce cas particulier où $n = 3$.

Nous proposons de comparer les qualités des images produites par le filtre bilatéral Bayer par rapport au filtre bilatéral appliqué après dématriçage. Nous utilisons un masque de taille 5×5 pour le filtre bilatéral et des masques de tailles 7×7 et 9×9 respectivement pour le filtrage du canal vert et des canaux bleu et rouge pour le filtrage bilatéral Bayer. Nous comparons les résultats des filtres pour des densités de photons par pixel $d_{h\nu}$ variant entre 10 et 6000. Pour chaque densité $d_{h\nu}$ nous faisons varier le paramètre de similarité photométrique h entre 0 et 100 par pas de 5. Nous utilisons les 24 images de l'annexe D. Les graphiques des figures 7.11 et 7.13 montrent respectivement les mesures de RMS pour le filtre bilatéral et pour le filtre bilatéral Bayer. Les figures 7.14 et 7.12 montrent respectivement un agrandissement des courbes des figure 7.11 et 7.13 pour des valeurs de $d_{h\nu}$ variants entre 10 et 6000. Les résultats obtenus sont très similaires et ne permettent pas de faire de distinctions évidentes entre les qualités des images produites par les deux filtres. La figure 7.15 montre les résultats des deux filtrages sur la même image bruitée ($d_{h\nu} = 100$). Les deux images filtrées possèdent le même RMS. L'image filtrée après dématriçage est flou, ses détails sont dégradés et on note la présence de tâches de couleurs de basses fréquences spatiales qui sont très visibles à l'oeil. Ces tâches colorées sont dues à la diffusion du bruit à travers les différents canaux de couleurs. Le filtrage bilatéral Bayer produit une image plus nette avec des détails mieux préservés, l'image est visiblement moins bruitée et ne contient pas de tâches de couleurs.

Il apparaît clairement que filtrer l'image de Bayer permet de produire des images de meilleures qualités en améliorant la réduction du bruit et en diminuant la perturbation de l'étape de dématriçage. Nous proposons maintenant de coupler le filtrage bilatéral Bayer avec la méthode d'estimation du meilleur paramètre de similarité photométrique h en fonction de la puissance du bruit dans l'image.

7.7 Filtrage bilatéral Bayer adaptatif

L'estimation du niveaux de bruit à partir de l'image du capteur proposée au début du chapitre est directement applicable pour le filtre bilatéral Bayer. Nous formulons ainsi un filtre bilatéral adaptatif Bayer. Nous proposons de faire les mesures d'estimation du gain G en calculant la variance $Var(Y_G)$ dans les zones non structurées $T^-(Y_G)$ du canal vert. La figure 7.16 montre les comparaisons des images produites entre le filtrage bilatéral adaptatif Bayer et le filtrage bilatéral appliqué après dématriçage. Nous simulons des images issues de capteurs avec différentes valeurs de $d_{h\nu}$, variant de 20 à 6000, on double à chaque pas le nombre de photons simulés. La première colonne de la figure montre les images bruitées par un bruit de photon puis une étape de dématriçage. La deuxième colonne montre les images obtenues avec un filtrage bilatéral et un paramètre

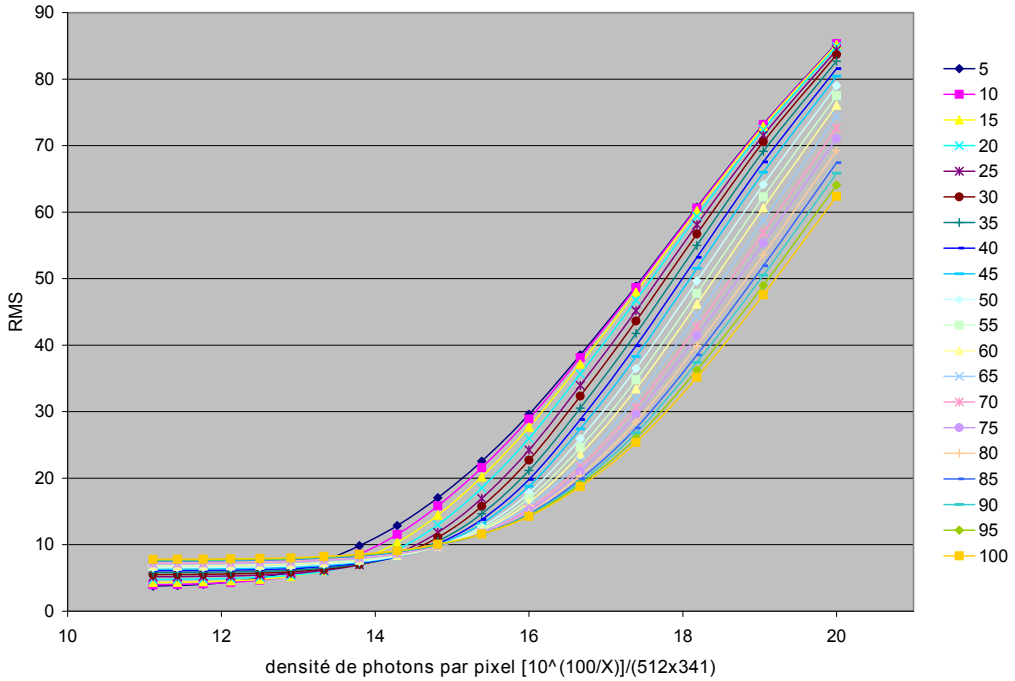


FIGURE 7.11 – Résultats des mesures du RMS pour le filtrage bilatéral appliqué après l'étape de dématriçage. Les mesures sont faites pour des densités de photons par pixel variants de 0.5 à 6000 et pour un paramètre h variant de 0 à 100.

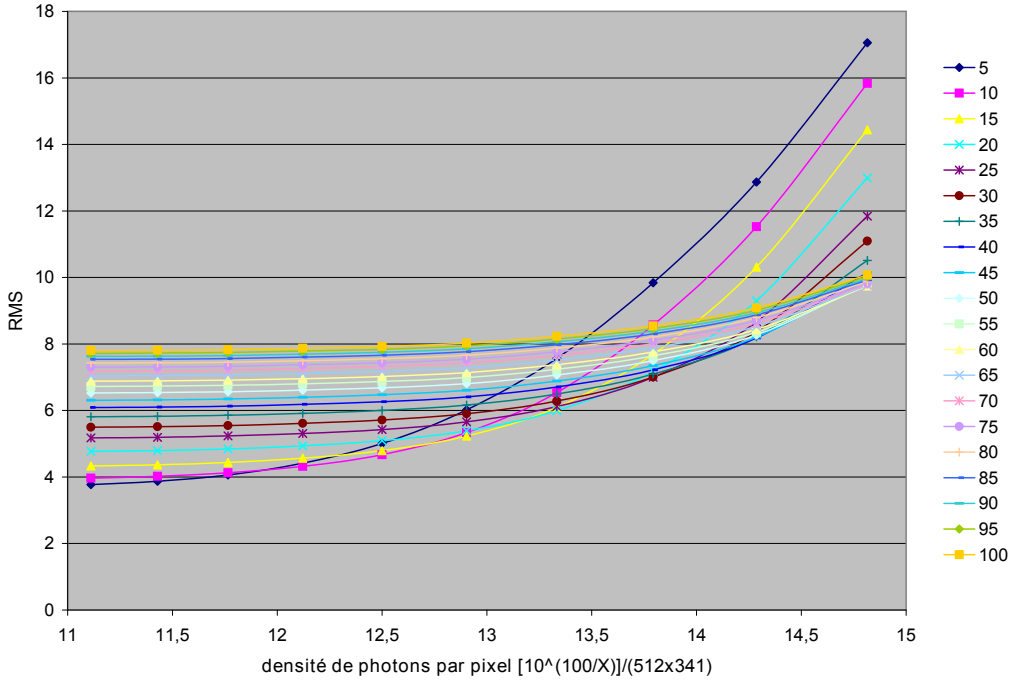


FIGURE 7.12 – Résultats des mesures du RMS pour le filtrage bilatéral appliqué après l'étape de dématriçage. Les mesures sont faites pour des densités de photons par pixel variants de 10 à 6000 et pour un paramètre h variant de 0 à 100.

de similarité photométrique $h = 20$. La troisième colonne montre les images produites par un filtrage bilatéral avec $h = 100$. La quatrième colonne montre le filtrage obtenu avec le filtre bilatéral Bayer adaptatif proposé. On peut voir que le filtre bilatéral avec

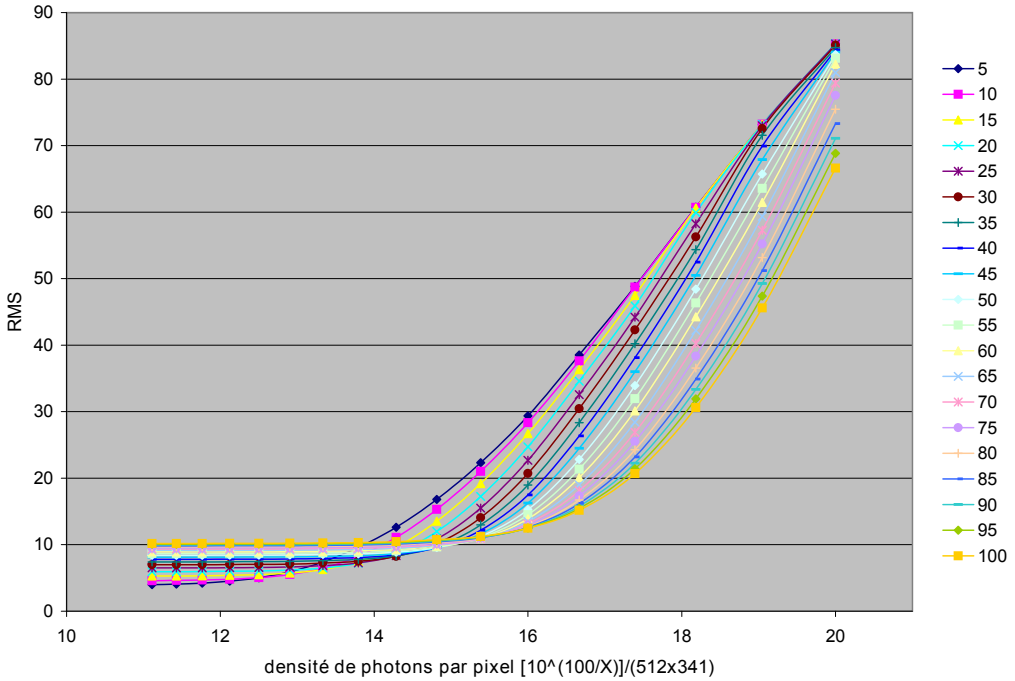


FIGURE 7.13 – Résultats des mesures du RMS pour le filtrage bilatéral appliqué avant l'étape de dématriçage. Les mesures sont faites pour des densités de photons par pixel variants de 0.5 à 6000 et pour un paramètre h variant de 0 à 100.

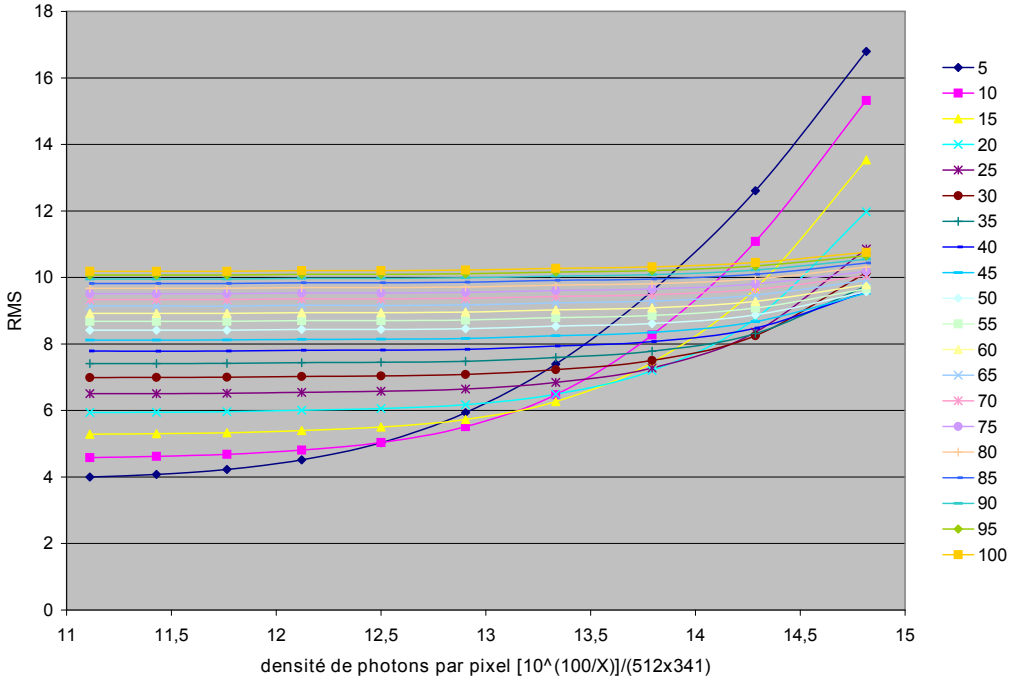
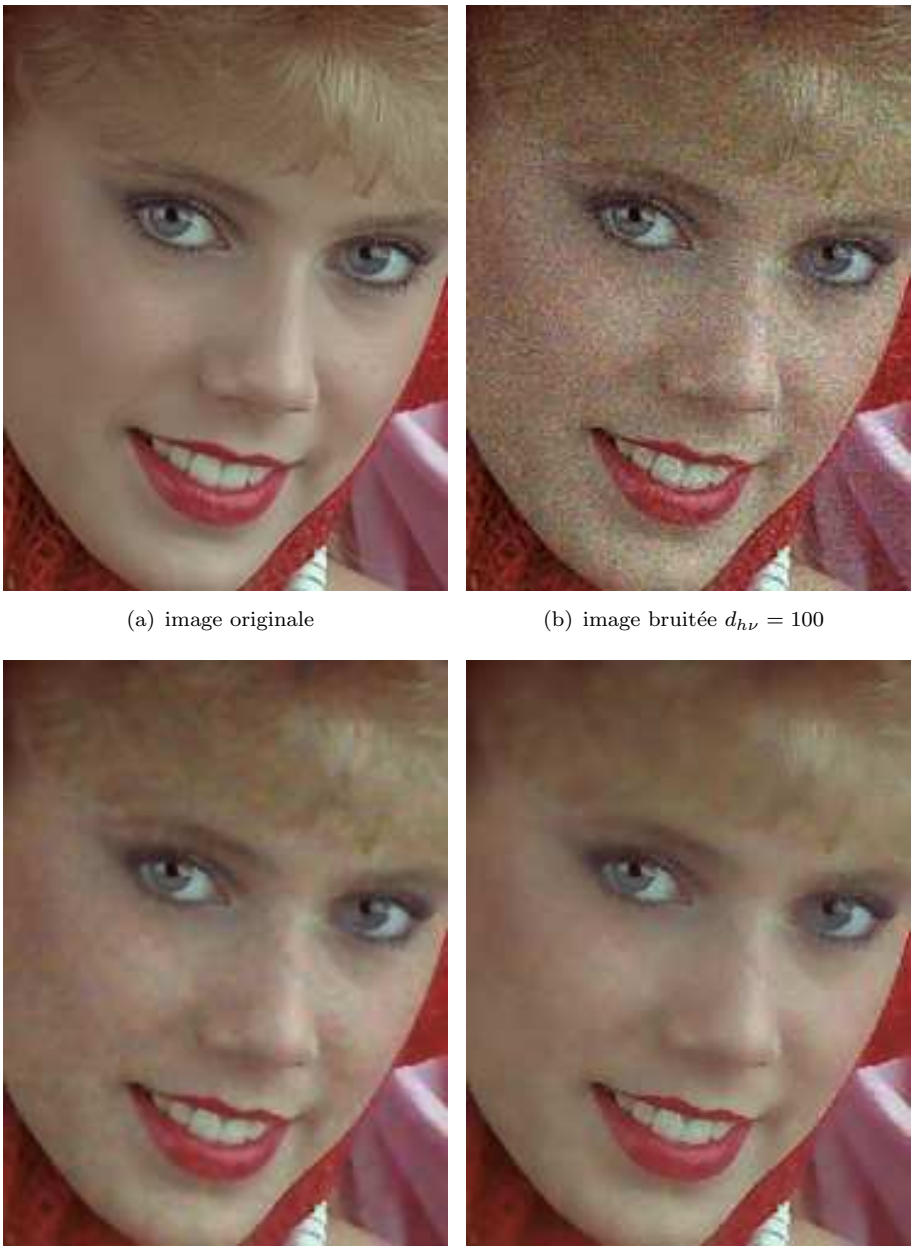


FIGURE 7.14 – Résultats des mesures du RMS pour le filtrage bilatéral appliqué avant l'étape de dématriçage. Les mesures sont faites pour des densités de photons par pixel variants de 10 à 6000 et pour un paramètre h variant de 0 à 100.

un paramètre h fixe ne s'adapte pas suffisamment à la puissance du bruit dans l'image. D'autre part, l'étape de dématriçage diminue l'efficacité du filtre, les images sont rendues floues et le bruit n'est pas bien filtré. Aussi, on peut voir que le filtre bilatéral (pour



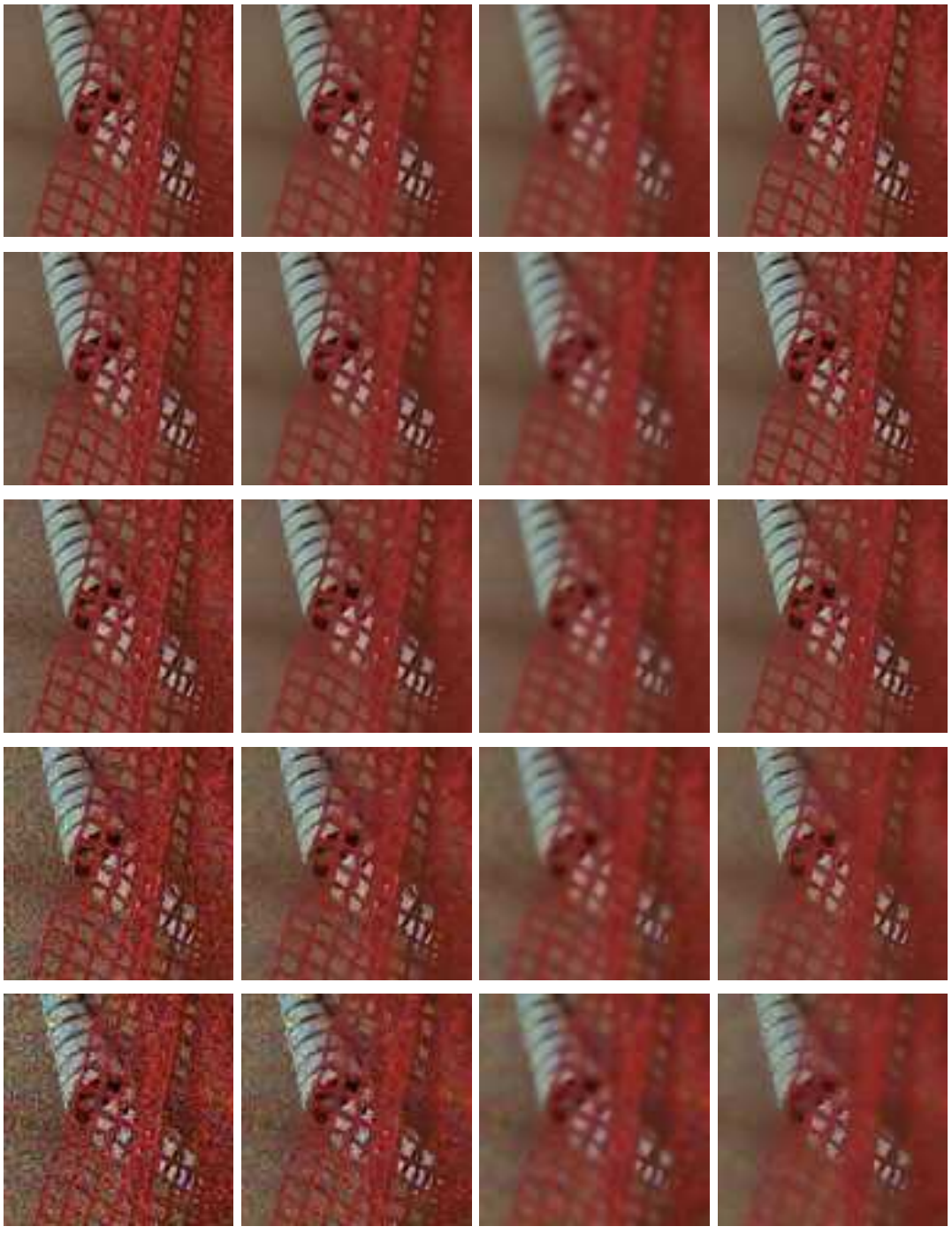
(a) image originale (b) image bruitée $d_{h\nu} = 100$

(c) image filtrée après dématriçage $h = 40$, RMS $= 7,22$ (d) image filtrée avant dématriçage $h = 25$, RMS $= 7,27$

FIGURE 7.15 – Comparaison visuelle entre le filtrage bilatéral appliqué avant et après l'étape de dématriçage.

$h = 100$) introduit des tâches de couleurs basses fréquences lorsque la variance du bruit augmente. Le filtre bilatéral Bayer adaptatif proposé possède les propriétés du filtre bilatéral adaptatif, il adapte efficacement son paramètre de similarité photométrique en fonction de la puissance du bruit, permettant d'obtenir le meilleur compromis entre le filtrage du bruit et la préservation des détails. Le filtre bilatéral adaptatif Bayer possède aussi les propriétés du filtre bilatéral Bayer, les images sont visuellement moins bruitées

et plus nettes, les structures de la scène sont préservées, il permet d'éviter l'apparitions des tâches de couleurs dû à la diffusion du bruit à travers les différents canaux de couleurs.



(a) Images bruitées (b) Filtre bilatéral $h = 20$ (c) Filtre bilatéral $h = 100$ (d) Filtre bilatéral Bayer adaptatif

FIGURE 7.16 – Comparaison de la qualité du filtrage entre le filtre bilatéral Bayer adaptatif et le filtre bilatéral appliqué après dématriçage.

7.8 Conclusion

Dans les appareils de captures d'images numériques, la puissance du bruit varie constamment en fonction des conditions d'éclairement de la scène. Pour permettre un filtrage optimal du bruit il est nécessaire que le filtre soit fortement adaptatif en fonction de cet éclairement. Le filtre bilatéral est un filtre à voisinage local adaptatif en fonction des structures de l'image. Il possède de plus la propriété de pouvoir adapté son paramètre de similarité photométrique. En considérant un modèle de bruit de photons, nous avons proposé une méthode de filtrage permettant d'adapter le paramètre de similarité photométrique du filtre en fonction de l'éclairement de la scène. Cette méthode exploite les propriétés statistiques poissonniennes du signal et le contrôle de gain appliqué sur l'image. L'estimation du bruit proposée montre une corrélation excellente entre le niveau de bruit estimé et le niveau de bruit réel. En appliquant notre méthode de détection du niveau de bruit pour régler le paramètre h du filtre bilatéral, nous obtenons un nouveau filtre bilatéral adaptatif qui permet d'obtenir le meilleur filtrage bilatéral pour différents niveaux de bruits de photons. Dans le cadre de l'application de ce filtre à des systèmes de captures d'images numériques en couleurs nous avons proposé une formulation du filtre bilatéral adaptée pour filtrer la matrice de Bayer, appelé filtre bilatéral Bayer. Nous avons montré que le filtre bilatéral Bayer produit une meilleure qualité visuelle d'image débruitée. Notamment les images obtenues contiennent moins de bruit chromatique, une netteté plus importante et une meilleure préservation des détails. Finalement nous avons proposé de fusionner le filtre bilatéral adaptatif avec le filtre bilatéral Bayer permettant ainsi de formuler un nouveau filtre, le filtre bilatéral Bayer adaptatif. Ce filtre adapte son paramètre de similarité photométrique en fonction de la puissance du bruit et permet d'éviter la dégradation des images introduite par la diffusion du bruit à travers les canaux de couleurs lors du dématriçage. **Ces travaux ont fait l'objet d'une publication à ICSP [17].**

Troisième partie

Implémentation et optimisation sur l'architecture

Chapitre 8

Dématriçage : simulation et optimisation sur processeur TM3270

L'optimisation consiste à modifier une fonction ou un programme pour qu'il fonctionne de manière plus efficace, en réduisant son temps d'exécution, l'espace occupé par les données du programme ou encore sa consommation d'énergie. Fondamentalement, l'optimisation ne doit intervenir que lorsque que le programme à optimiser fonctionne et répond aux fonctionnalités spécifiées [108]. Avant de commencer l'optimisation, il faut savoir mesurer la vitesse d'exécution du code et pouvoir localiser les bloques les plus coûteux en temps d'exécution. On fait ainsi apparaître les *boucles critiques*. Il n'est pas rare que 80% à 90% de l'exécution d'un code soit consacrée à un faible pourcentage du programme. Ces étapes de mesures et de localisations des vitesses d'exécutions peuvent être réalisées à l'aide d'outils spécialisés appelés *profilers*. Ces *profilers*, sont capables de compter le pourcentage de temps d'exécution de chaque fonction du code et le nombre de cycles effectués. Ils permettent ainsi de mesurer les gains de performances obtenus après chaque optimisation. Un programme informatique peut être optimisé à plusieurs niveaux. Au niveau algorithmique, en choisissant pour la même fonctionnalité l'algorithme le moins complexe. On peut aussi optimiser un programme informatique en utilisant les techniques d'optimisations standards telles que l'utilisation de variables locales pour éviter l'aliasing de mémoire ; l'utilisation d'opérations logiques telles que les décalages de bits pour remplacer les opérations de divisions et de multiplications par des puissances de 2, diminuant ainsi les coûts d'appels des fonctions respectives ; l'utilisation du mot clef *inline* pour éviter les appels de fonctions ; la suppression des opérations en virgules flottantes ; l'utilisation de look-up-tables pour remplacer les calculs complexes et enfin l'utilisation du déroulage de boucles pour réduire le nombre de tests et de sauts

de boucles tout en augmentent le parallélisme d'exécution. Finalement, l'optimisation peut être faite directement dans l'écriture du code en utilisant le jeu d'instructions du processeur cible.

Dans ce chapitre, nous présentons l'implémentation et l'optimisation d'algorithmes de dématriçage sur le processeur TriMedia TM3270. Nous traitons les algorithmes de Hamilton [8], Hamilton corrigé par la méthode de LMDC (voir section 5.1.3), Hirakawa [18] et GEDI (voir chapitre 5). Nous utilisons le système de simulation et de compilation TriMedia, décrit dans l'annexe B.

8.1 Versions naïves et pseudos-codes des algorithmes

La version naïve d'un code est sa version fonctionnelle, sans optimisations. Les pseudos-codes des algorithmes étudiés sont présentés dans l'annexe C. La méthode de réduction des artéfacts d'interpolations (IAR, voir section 4.10) est commune à tous les algorithmes étudiés, son pseudo-code est présenté dans l'algorithme 4. Également, tous les algorithmes de dématriçage étudiés dans ce chapitre utilisent la méthode de constance des teintes pour interpoler les canaux de couleurs rouge et bleu à partir du canal vert. Le pseudo-code de sa version naïve est présenté dans l'algorithme 4. Les pseudos-codes de chaque algorithme présentent uniquement la reconstruction du canal vert. Ils sont détaillés dans les algorithmes 5, 6, 7, respectivement pour Hamilton, GEDI et Hamilton corrigé par LMDC. Le pseudo-code de l'algorithme de Hirakawa est divisé en plusieurs parties. L'algorithme 8 présente la conversion de l'espace de couleur RGB vers l'espace de couleur CIELab. L'algorithme 9 montre le calcul des paramètres d'homogénéité. L'algorithme 10 donne le pseudo-code de l'application du critère d'homogénéité et l'algorithme 11 présente le calcul de la fonction d'homogénéité.

8.2 Optimisations algorithmiques de Hamilton, Hirakawa, Hamilton corrigé par LMDC et GEDI

Dans cette section, nous présentons les optimisations algorithmiques que nous avons utilisées. De manière générale, ces optimisations consistent à minimiser mathématiquement le nombre d'opérations nécessaires pour appliquer un algorithme. Elles ne modifient pas la fonctionnalité du code et sont indépendantes de la plateforme d'implémentation.

8.2.1 Convolution séparable

Dans la section 5.1.3, nous proposons de corriger les choix de directions d'interpolations des pixels par les choix majoritaires locaux dans un élément structurant de taille et de forme définie. Le choix de la direction majoritaire se fait à travers le calcul d'une carte de directions traduisant les choix de l'estimateur (dans cette carte on représente par exemple la direction horizontale par un 1 et le choix de la direction verticale par un 0). Nous utilisons un élément structurant M_n carré et de taille $(n+1) \times (n+1)$. Pour calculer les choix majoritaires de directions dans l'image, on applique une convolution de la carte de direction par l'élément M_n unitaire. Cette convolution est linéaire et donc séparable. L'équation 8.1, illustre la séparabilité de cette convolution pour $n = 2$. En utilisant cette technique, on passe du calcul de n^2 additions et $n^2 - 1$ multiplications au calcul de $((n+1)^2 - 1)/2$ additions et $(n+1)^2/2$ multiplications. Dans notre cas, les coefficients de la matrice de convolution étant unitaires et la taille de la matrice étant 3×3, la séparabilité permet de passer du calcul de 8 additions à 4 additions. L'utilisation des propriétés de séparabilité n'introduit aucunes pertes d'informations et n'a donc aucunes répercussions sur la qualité de l'image produite.

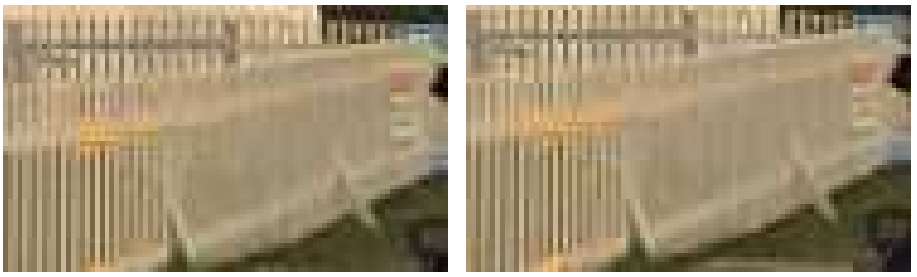
$$m \star \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = m \star \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \star \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \tag{8.1}$$

Dans cette équation, m représente une carte de direction horizontale ou verticale.

8.2.2 Filtre médian approximé

Un filtre médian calculé dans un masque de taille 3×3 est utilisé pour appliquer l'algorithme de réduction des artéfacts de couleurs (voir section 4.10). Le filtre médian est non-linéaire et non-séparable. Pour réduire sa complexité algorithmique il est possible d'en calculer une approximation. Celle-ci est présentée dans l'équation 8.2, où $\widehat{m}(A)$ représente la valeur médiane approximée des données de la matrice A et $m(x, y, z)$ retourne la valeur médiane des éléments x, y et z. Comme pour la convolution séparée, le filtre médian est d'abord appliqué sur les colonnes puis sur les lignes, comme cela est illustré dans l'équation 8.2. En comparaison, le filtre médian classique calculé dans un masque de taille 3×3 nécessite 36 comparaisons, le filtre médian approximé nécessite 12 comparaisons, permettant un gain de plus de 67% de réduction de complexité. Sur la figure 8.1 on peut voir que cette approximation n'altère pas la qualité du filtrage.

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \rightarrow \widehat{m}(A) = m(m(a,d,g), m(b,e,h), m(c,f,i)) \tag{8.2}$$



(a) Filtre médian classique (b) Filtre médian approximé

FIGURE 8.1 – Comparaison de la qualité des images filtrées par la méthode de l'IAR avec le filtre médian classique et le filtre médian approximé.

8.2.3 Filtre médian rapide

Pour optimiser le nombre d'opérations du filtre médian, il est possible d'utiliser la formulation du filtre médian rapide. Considérons la matrice de la formule 8.3, on peut alors calculer la valeur médiane de cette matrice en appliquant les étapes suivantes :

1. ordonner chaque colonne, adg, beh et cfi (9 comparaisons) ;
2. sélectionner la valeur maximale de abc, la valeur médiane de def et la valeur minimale de ghi (7 comparaisons) ;
3. sélectionner la valeur médiane des trois valeurs sélectionnées dans l'étape 2 (3 comparaisons).

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \tag{8.3}$$

Cette formulation permet de paralléliser et de réduire le nombre d'opérations nécessaires à l'application du filtre médian sans pertes de qualité. Le nombre de comparaisons est réduit à 19, permettant ainsi de réduire la complexité algorithmique de 47%.

8.2.4 Élimination du filtrage médian sur le canal vert pour le calcul de l'IAR

La complexité de l'algorithme de réduction des artefacts d'interpolations peut-être réduite en supprimant le filtrage médian du canal vert. En effet, on suppose que le filtrage du canal vert n'est pas indispensable pour éliminer les artéfacts de couleurs, l'adaptation des hautes fréquences des canaux rouge et bleu étant déjà suffisante pour réduire de manière efficace ces artéfacts. En considérant l'équation 8.4, où R, G et B sont respectivement les plans de couleur rouge, vert et bleu et R', G' et B' sont respectivement les plans de couleurs rouge, vert et bleu filtrés. On peut constater que le filtre nécessite le calcul de quatre fonctions médianes sur les différences des plans $R-G$, $B-G$, $G-R'$ et $G-B'$. Le filtrage du plan vert nécessite pour lui seul l'utilisation de deux fonctions médianes. Éliminer le filtrage du plan vert revient donc à réduire la complexité de l'algorithme de 50%. La figure 8.2 montre les résultats de l'algorithme de l'IAR avec et sans filtrage du plan vert. On constate que les qualités d'images sont équivalentes.

$$\begin{aligned} R' &= m(R-G)+G \\ B' &= m(B-G)+G \\ G' &= \frac{[m(G-R')+R']+[m(G-B')+B']}{2} \end{aligned} \tag{8.4}$$

(a) Méthode IAR classique (b) Méthode IAR sans filtrage du plan vert

FIGURE 8.2 – Comparaison de la qualité des images filtrées avec l'algorithme IAR classique et IAR sans filtrage du canal vert.

8.3 Optimisations standards

Dans cette section, nous décrivons les optimisations standards que nous avons utilisées.

8.3.1 Utilisation des décalages de bits

Les multiplications et les divisions sont des opérations mathématiques plus lentes que les additions et les soustractions, elles sont coûteuses en nombre de cycles de calculs. Ces opérations peuvent être remplacées par des décalages de bits lorsque les coefficients sont des puissances de 2. Les décalages de bits sont des opérations très rapides, elles permettent d'utiliser seulement un cycle d'exécution par décalage. Dans l'équation 8.5, on reprend une partie du calcul de l'équation 4.3 utilisée pour l'interpolation du plan vert, soit :

$$m_{j,i} = \frac{2.R_{j,i} + R_{j,i-1} + R_{j-1,i}}{4} \tag{8.5}$$

en remplaçant les opérations de multiplications et de divisions par des décalages de bits, on peut alors écrire :

$$m_{j,i} = (R_{j,i} \ll 1 + R_{j,i-1} + R_{j-1,i}) \gg 2 \tag{8.6}$$

où le signe $\ll n$ représente un décalage à gauche de n bits et correspond à une multiplication par 2^n. Le signe $\gg n$ représente un décalage à droite de n bits et correspond à une division par 2^n. Pour les cas où le coefficient est un entier qui n'est pas une puissance de 2, l'opération est remplacée par une série d'additions ou de combinaisons entre additions et décalages de bits. Un exemple est montré dans les équations 8.7 et 8.8.

$$\text{WidthMul3} = 3 \times \text{Width} \tag{8.7}$$

$$\text{WidthMul3} = (\text{Width} \ll 1) + \text{Width} \tag{8.8}$$

où, WidthMul3 est la multiplication de la largeur Width d'une image par 3. Cette opération peut être remplacée par un décalage à gauche et l'addition de 1 fois la largeur.

8.3.2 Remplacement des opérations en virgule flottante

Le processeur TM3270 est capable de traiter des opérations avec des coefficients codés en virgules flottantes. Cependant ces opérations sont très lentes si on les compare aux opérations utilisant des coefficients codés en entiers. Parmi les algorithmes implémentés,

seule la conversion de l'espace de couleur RGB vers l'espace CIELab nécessite l'utilisation d'opérations en virgules flottantes. Pour réduire la complexité de cette étape, nous avons approximé les calculs flottants par des calculs d'entiers. L'approximation de la composante chromatique $X(.)$ de l'équation 8.9 est présentée dans l'équation 8.10. Dans ces équations, $X(.)$ représente la composante X dans l'espace XYZ, $R(.)$, $G(.)$ et $B(.)$ sont respectivement les trois canaux de couleurs rouge, vert et bleu. Les coefficients codés en virgules flottantes sont approximés par la multiplication d'un coefficient entier et d'une division par décalage de bits.

$$X(.) = 0.39.R(.) + 0.33.G(.) + 0.17.B(.) \tag{8.9}$$

$$X(.) = [25.R(.) + 21.G(.) + 11.B(.)] \gg 6 \tag{8.10}$$

8.3.3 Utilisation des Look-Up Tables

Une look-up table est généralement utilisée pour approximer et remplacer un calcul complexe par une lecture dans la mémoire. Le processeur multimédia TM3270 n'est pas capable de calculer certaines opérations mathématiques comme les calculs de puissances ou les calculs de logarithmes. Lors de la conversion de l'espace de couleur RGB vers l'espace CIELab, le calcul d'une racine cubique est nécessaire pour la conversion de l'espace XYZ vers l'espace LAB. La fonction $f_{Lab}(.)$ utilisée pour cette conversion est présentée dans l'équation 8.11 avec $0 > x_{\text{norm}} > 1$.

$$\begin{array}{llll} si & x \leq 0.008856 & alors & f_{LAB}(x_{\text{norm}}) = 7.787.x_{norm} + 16/116 \\ & & sinon & f_{LAB}(x_{\text{norm}}) = \sqrt[3]{X_{norm}} \end{array} \tag{8.11}$$

Par exemple, considérons le cas du calcul de $f_{Lab}(0.490) = 0,788$. Les résultats pré-calculés de 1024 valeurs de $f_{Lab}(\text{norm})$ sont préalablement mémorisés dans un tableau de type virgules flottante. Le tableau 8.1 montre les valeurs pré-calculées de la fonction f_{Lab} entre les colonnes 501 et 505. La valeur correspondante à $f_{Lab}(0.490)$ peut être directement lue dans le tableau en calculant le numéro de la colonne $C_{f_{Lab}}(x_{\text{norm}})$ correspondante soit $C_{f_{Lab}}(x_{\text{norm}}) = \lfloor \text{norm} \times 1024 \rfloor$, soit $C_{f_{Lab}}(0,490) = 502$.

TABLE 8.1 – LUT de la fonction $f_{Lab}(\text{norm})$ entre les colonnes 501 et 505

501	502	503	504	505
0,788	0,788	0,789	0,790	0,790

TABLE 8.2 – Exemple de déroulage de boucle manuel

Sans déroulage	**Déroulage complet**	**Déroulage partiel**

`for (i=0; i<4; i++) {`
`  A(i);`
`}`
	`A(0);`
	`A(1);`
	`A(2);`
	`A(3);`
		`for (i=0; i<2; i++) {`
		`  A(2*i);`
		`  A(2*i+1);`
		`}`

TABLE 8.3 – Exemple de d'utilisation des *pragmas* pour le déroulage de boucle

Déroulage manuel	**Utilisation des *pragmas***

`for (i=0; i<n; i++) {`
`  A(2*i);`
`  A(2*i+1);`
`}`
`}`
	`#pragma TCS_unroll=2`
	`for (i=0; i<n; i++) {`
	`  A(i);`
	`}`

8.3.4 Déroulage de boucles

Le déroulage de boucles est une technique de transformation des boucles d'itérations dont le but est d'optimiser la vitesse d'exécution d'un programme en contrepartie de la taille de son code. La vitesse d'exécution est améliorée en réduisant le nombre de tests de fin de boucles et en parallélisant les opérations. En effet, les boucles peuvent être réécrites comme une séquence de traitements indépendants, permettant ainsi de réduire le nombre d'itérations. Si les opérations déroulées sont indépendantes les unes des autres, les traitements peuvent être exécutés parallèlement. Le Tableau 8.2 montre un exemple de déroulage de boucles manuel où $A(i)$ est une fonction utilisant i comme paramètre d'entrée. Le compilateur TriMedia est capable de dérouler les boucles de manière automatique en ajoutant des options de compilation ou en précisant au compilateur ce qu'il doit faire avec l'ajout de *pragmas*. L'utilisation des *pragmas* permet de préciser au compilateur de dérouler ou de ne pas dérouler les boucles d'itérations, de laisser le compilateur choisir le nombre d'itérations automatiquement ou d'imposer un nombre d'itérations. L'utilisation des *pragmas* permet d'éviter la réécriture du code. Le tableau 8.3 montre un exemple de leurs utilisations.

8.4 Fonction inline

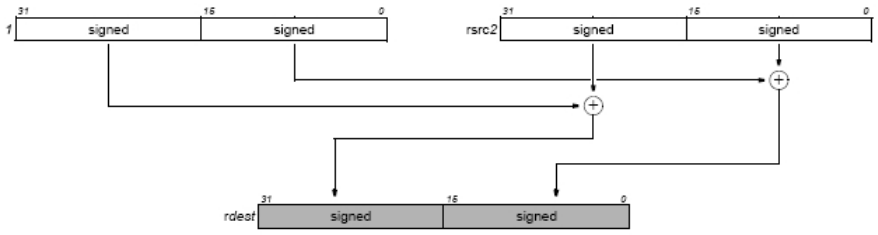
Les appels de fonctions dans le code induisent des dépenses de cycles. L'option de *function inlinning* indique au compilateur de réécrire directement le code de la fonction dans le programme à l'endroit précisé évitant ainsi les ruptures de séquences, en contrepartie de l'augmentation de la taille du codes.

8.5 Optimisations TriMedia

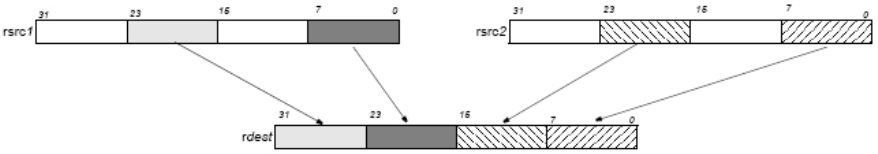
Le système de compilation TriMedia possède des outils d'optimisations du code. Parmi ces outils, on trouve un jeu d'instructions spécialisées pour exploiter de manière optimale les performances de traitement du processeur. Les opérations du jeu d'instruction TriMedia fonctionnent avec des registres de données de tailles 32 bits. Ces registres permettent l'utilisation de données concaténées pour paralléliser les opérations du processeur lorsque celles-ci sont codées sur 8 ou 16 bits. Typiquement, en considérant des données de taille 16 bits, ces opérations permettent au processeur d'exécuter deux opérations simultanées avec le même registre d'entrée, ce qui représente idéalement un gain de rapidité d'exécution de 50%. La figure 8.3 montre deux exemples d'instructions opérant sur des données de taille 16 bits dans des registres de taille 32 bits. La fonction *DUALIADD* prend quatre valeurs de 16 bits en entrée dans 2 registres de 32 bits. Elle calcule deux additions de données 16 bits et retourne les deux résultats dans deux registres de taille 32 bits. La fonction *MERGEDUAL16LSB* concatène les bits de poids faibles de deux paires de données de tailles 16 bits préalablement stockées dans des registres de tailles 32 bits. Les résultats de la concaténation sont stockés dans un registre de taille 32 bits. Le système de compilation TriMedia possède aussi des outils de *profiling* permettant d'optimiser automatiquement le code avec des options de compilation. De manière générale, ces outils fonctionnent avec deux cycles de compilation et d'exécution. Dans un premier temps, le compilateur fait le *profiling* du programme et identifie les parties du code les plus souvent exécutées. Dans un deuxième temps, le compilateur utilise ces informations pour produire un exécutable plus rapide en créant un arbre de décision optimisé.

8.6 Résultats et conclusion

On rappelle que l'implémentation et les simulations des algorithmes sont faites en utilisant le système de simulation TCS. La fréquence du processeur TM3270 est réglée à 350MHz. La méthode de réduction des artéfacts d'interpolations est étudiée séparément



(a) DUALIADD



(b) MERGEDUAL16LSB

FIGURE 8.3 – Exemples d'opérations dédiées au processeur TriMedia utilisant des registres d'entrées de taille 32-bits et permettant de traiter simultanément des données concaténées de taille 16-bits : (a) la fonction *DUALIADD* prend quatre valeurs de 16 bits en entrée dans 2 registres de 32 bits, elle calcule deux additions de données 16 bits et retourne les deux résultats de ces additions dans deux registres de taille 32 bits, (b) fonction *MERGEDUAL16LSB*, concatène les bits de poids faibles de deux paires de données de tailles 16 bits, stockées dans des registres de tailles 32 et stocke les résultats de la concaténation dans un registre de taille 32 bits.

sous ces différentes versions. Les résultats sont présentés dans la section 8.6.1. La section 8.6.2 présente les résultats des algorithmes de Hamilton [8], Hirakawa [18], GEDI et Hamilton corrigé par la méthode de LMDC. On mesure le nombre d'images traitées par seconde en résolution VGA pour le traitement vidéo et le temps de traitement d'une image de 5 méga-pixels pour la photographie. On mesure les optimisations en calculant le nombre de cycles nécessaires au traitement d'un pixel. Pour comparer les performances d'optimisations, nous séparons celles-ci en quatre versions du code :

- **Naive,** c'est la version du code qui ne contient aucunes optimisations ;
- **Algorithmique,** c'est la version du code qui contient seulement des optimisations algorithmiques ;
- **Standard,** c'est la version du code qui contient à la fois les optimisations standards et algorithmiques ;
- **TriMedia,** c'est la version du code qui contient les optimisations algorithmiques, les optimisations standards et les optimisations du jeu d'instructions TriMedia.

8.6.1 Réduction des artéfacts d'interpolation

Le tableau 8.4 montre les résultats des optimisations de l'algorithme de réduction des artéfacts (IAR). La première version du code est la version naïve. Cette version est très complexe, elle utilise 1136,74 cycles d'horloge pour traiter un pixel. Ceci est dû au 36 comparaisons nécessaires au filtrage médian dans un masque de taille 3×3. L'approximation du filtre médian permet d'obtenir une amélioration des performances de 93,5%. Les optimisations standards permettent d'obtenir un gain de 96,6% par rapport à la version naïve en traitant un pixel en 38,49 cycles. Pour les optimisations TriMedia nous avons implémenté trois versions. La première version identifiée par *RGB* utilise l'approximation du filtre médian et filtre les canaux rouge, vert et bleu, elle permet le traitement d'un pixel en 11,45 cycles et une amélioration par rapport à la version naïve de 99%. La deuxième version identifiée par *RB* utilise l'approximation du filtre médian et filtre seulement les canaux rouge et bleu, permettant ainsi une performance de traitement d'un pixel tout les 6,04 cycles et une amélioration de 99,5% par rapport à la version naïve. La troisième version utilise le filtre médian rapide et permet de traiter un pixel en 27,14 cycles et d'obtenir une amélioration de 99,3% par rappport à la version naïve du code.

TABLE 8.4 – Résultats des optimisations pour l'algorithme de réduction des artéfacts

Version du code		**Cycles / pixel**	**Amélioration / à la version naïve (%)**
Naïve		1136,74	-
Algorithmique		73,77	93,5
Standard		38,49	96,6
TriMedia	RGB	11,45	99,0
	RB	6,04	99,5
	Médian rapide	27,14	97,6

8.6.2 Algorithmes de dématriçage

Le tableau 8.5 montre les résultats des optimisations en nombre de cycles par pixel pour chaque algorithme étudié. Les performances d'optimisations sont calculées pour les versions des codes TriMedia par rapport aux versions des codes naïves.

Pour les versions naïves, l'algorithme de Hamilton possède la plus faible complexité avec 66,17 cycles par pixel. L'algorithme de Hamilton corrigé par LMDC est deux fois plus lent avec 126,59 cycles par pixel. GEDI utilise 182,70 cycles pour traiter un pixel, soit 2,7 fois plus que Hamilton. Enfin Hirakawa est 50 fois plus complexe que Hamilton avec 3352,69 cycles par pixel. La différence des performances entre l'algorithme de Hamilton

et sa version corrigée par LMDC s'explique par l'ajout d'une carte d'évaluation de la direction dans un masque carré de taille 3×3. L'algorithme GEDI pour sa part possède deux étapes de reconstruction du canal vert et une carte d'évaluation de la direction. Enfin l'algorithme de Hirakawa nécessite plusieurs étapes de calculs complexes comme la conversion de l'espace de couleur et l'évaluation de l'homogénéité pour deux versions dématriçées de l'image.

Après avoir appliqué toutes les étapes d'optimisations (version TriMedia), les améliorations des algorithmes sont comprises entre 84,9% et 95,7% par rapport à leurs versions naïves. On remarque, que plus un algorithme est complexe, plus le pourcentage d'optimisation est élévé. De manière générale, on remarque que l'amélioration la plus importante est obtenue avec l'étape d'optimisations standards. Particulièrement l'algorithme de Hirakawa passe de 3195,3 à 222,63 cycles, cette optimisation est principalement due à l'utilisation d'une LUT pour la conversion de l'espace des couleurs. Finalement, l'algorithme de Hamilton nécessite 9,21 cycles par pixel, Hamilton corrigé par LMDC nécessite 11,86 cycles par pixel, soit 2,21 cycles de plus pour appliquer la correction par LMDC. L'algorithme GEDI requiert 16,5 cycles pour traiter un pixel soit 7,29% cycles de plus que Hamilton et 5,08 cycles de plus que Hamilton corrigé par LMDC. Enfin, Hirakawa reste toujours très complexe avec 143,06 cycles pour traiter un pixel, soit une complexité 15,5 fois plus importante que Hamilton. La figure 8.4 montre pour chaque algorithme le rapport entre la qualité d'image produite (MSE) et le nombre de cycles nécessaires pour traiter un pixel. On remarque la faible complexité de GEDI par rapport à Hirakawa pour une qualité d'image similaire.

Pour avoir une meilleure lisibilité des résultats, on présente pour chaque algorithme les temps nécessaires pour traiter une image de 5 méga-pixels, correspondant typiquement à la taille d'une image photographique en téléphonie mobile, l'histogramme des résultats est présenté sur la figure 8.5. D'autre part on présente pour chaque algorithme le nombre d'images traitées par seconde en résolution VGA (640×480), typiquement la résolution des images vidéos en téléphonie mobile, l'histogramme des résultats est présenté sur la figure 8.6. On impose un cahier des charges, les algorithmes doivent pouvoir traiter une image de 5 méga-pixels en moins de 0,5 secondes et traiter plus de 25 images par seconde en résolution VGA.

L'algorithme de Hamilton est le plus performant il permet de traiter une image de 5 méga-pixels en 0,13 secondes. L'algorithme de Hamilton corrigé par LMDC, pour une qualité d'image légèrement meilleure, traite une image de 5 méga-pixels en 0,16 secondes, soit 0,03 secondes de plus. GEDI pour une qualité d'image très supérieure à Hamilton permet de traiter une image de 5 méga-pixels en 0,23 secondes soit 0,10 secondes de plus que Hamilton. Ces trois algorithmes respectent le cahier des charges imposé. D'autre

part, Hirakawa qui possède la même qualité d'image que GEDI, traite une image de 5 méga-pixels en 2,04 secondes, soit 16 fois plus lentement que Hamilton et 9 fois plus lentement que GEDI. Hirakawa ne permet pas de respecter le cahier des charges pour les images de résolution 5 méga-pixels.

L'algorithme de Hamilton permet de traiter 74,7 images de résolution VGA par seconde. L'algorithme de Hamilton corrigé par LMDC pour une qualité d'image légèrement meilleure permet de traiter 63,67 images de résolution VGA par seconde, soit environ 11 images de moins. GEDI pour une qualité d'image très supérieure à Hamilton permet de traiter 50,54 images de résolution VGA par seconde, soit environ 24 images de moins que Hamilton. Ces trois algorithmes respectent le cahier des charges imposé. D'autre part, Hirakawa traite seulement 7,64 images de résolution VGA par seconde, soit environ 10 fois moins d'images que Hamilton et 6,6 fois moins que GEDI. Hirakawa ne permet pas de respecter le cahier des charges pour les images de résolution VGA.

TABLE 8.5 – Résultats des optimisations des algorithmes de reconstruction. Les valeurs sont données en nombre de cycles/pixel et les améliorations sont calculées pour la version TriMedia du code.

Version du code	**Hamilton**	**Hamilton + LMCD**	**GEDI**	**Hirakawa**
Naive Cycles/pixel	66,17	126,59	182,70	3352,69
Algorithmique Cycles/-pixel	66,17	99,23	109,02	3195,13
Standard Cycles/pixel	17,66	36,65	41,28	222,63
TriMedia Cycles/pixel	9,21	11,86	16,50	143,06
Amélioration / Version naïve (%)	86,1	90,6	91,0	95,7
images / seconde (VGA)	74,7	63,7	50,5	7,6
temps de traitement en seconde pour une image de 5 Méga pixels	0.13	0.16	0.23	2.04

8.6.3 Conclusion

Dans ce chapitre, nous avons présenté le processeur multimédia TriMedia TM3270 et son système de compilation et de simulation TCS. Nous avons présenté les différentes techniques d'optimisations algorithmiques standards et TriMedia utilisées pour optimiser les simulations d'exécutions des algorithmes de dématriçage sur le TCS pour le processeur TriMedia Tm3270. Nous avons présenté les bonnes performances d'optimisation obtenues, elles sont comprise entre $84,9\%$ et $95,7\%$ par rapport aux versions naïves des algorithmes. Nous avons présenté les performances d'exécutions des algorithmes de Hamilton, Hamilton corrigé par LMDC, GEDI et Hirakawa en nombre de cycles pour

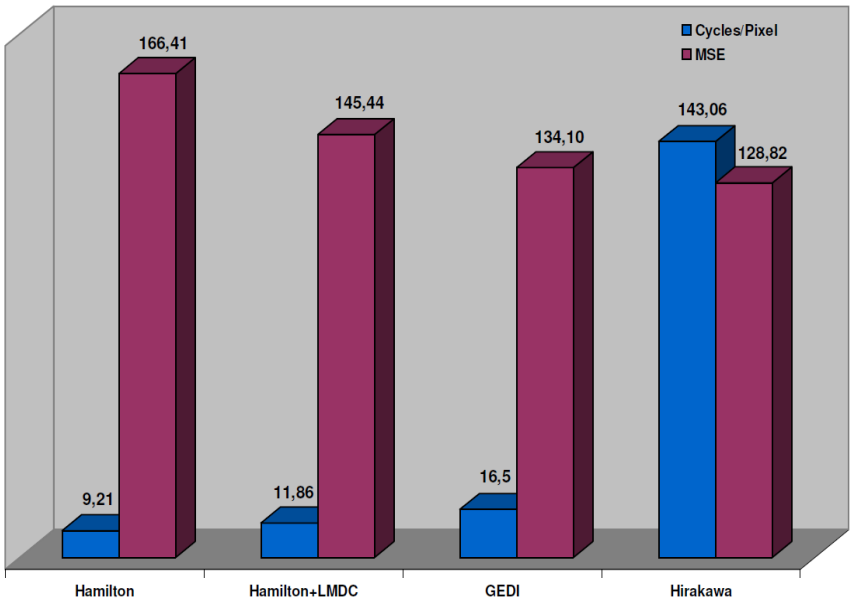


FIGURE 8.4 – Comparaison entre la qualité du MSE et le nombre de cycles de traitement par pixel pour chaque algorithme de dématriçage étudié.

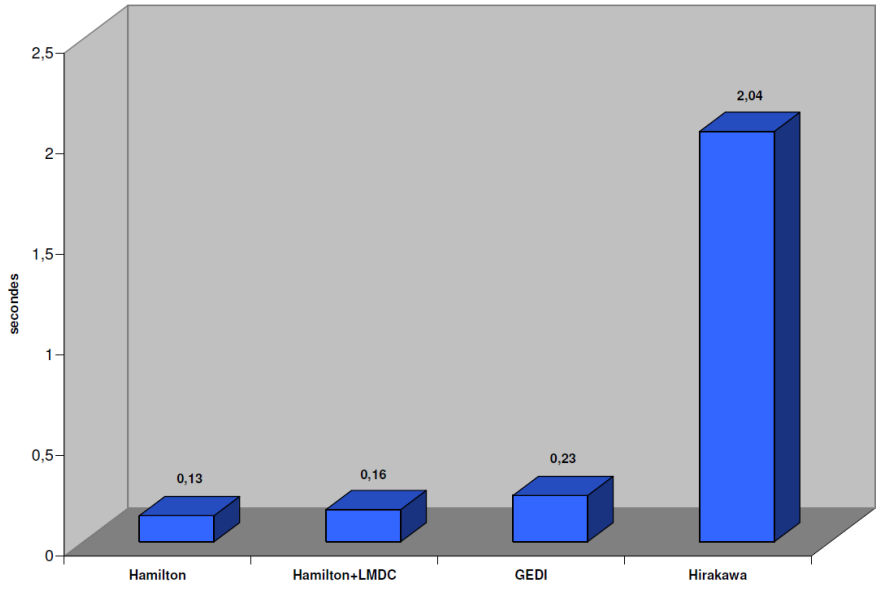


FIGURE 8.5 – Temps en secondes pour traiter une image de taille 5 méga-pixel.

traiter un pixel, en secondes nécessaires pour traiter une image de 5 méga-pixel et en nombre d'images traitées par seconde à la résolution VGA. Nous avons imposé un cahier des charges pour les performances des algorithmes. Ils doivent être capable de traiter une image de résolution 5 méga-pixels en moins de 0.5 secondes et traiter au minimum 25 images de résoluation VGA par seconde. Nous avons vu qu'en utilisant toutes les optimisations, il est possible de respecter le cahier des charges imposé pour les algorithme de Hamilton, Hamilton corrigé par LMDC et GEDI. Nous avons vu que pour une meilleure

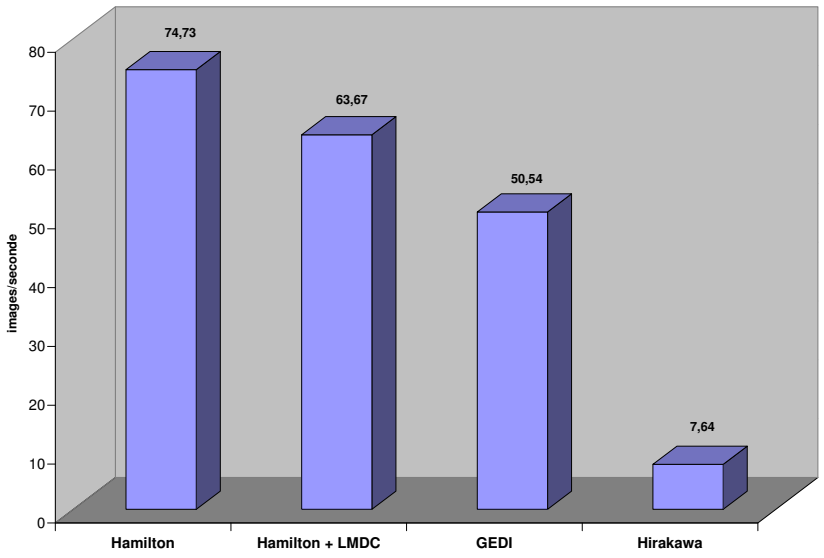


FIGURE 8.6 – Calcul du nombre d'images de résolution VGA traitées par seconde.

qualité d'image reconstruite l'algorithme de Hamilton corrigé par la méthode LMDC est seulement 1,28 fois plus complexe que l'algorithme de Hamilton. Pour une qualité d'image nettement supérieure, l'algorithme GEDI est seulement 1,8 fois plus complexe que l'algorithme de Hamilton. D'autre part, nous avons vu que l'algorithme de Hirakawa qui possède la même qualité d'image que GEDI est 15,5 fois plus complexe que Hamilton et 9 fois plus complexe que GEDI, l'algorithme de Hirakawa ne permet pas de respecter le cahier des charges imposé. Finalement, complémentairement à l'étude de qualité d'image réalisée dans le chapitre 5 on montre dans ce chapitre que l'algorithme GEDI possède à la fois de bonnes performances d'exécutions sur un processeur multimédia standard utilisé en téléphonie mobile associé à une très bonne qualité d'image produite. **Les travaux menés dans ce chapitre ont fait l'objet d'une publication [17] à DASIP2007 (Design and Architecture for Signal and Image Processing).**

Chapitre 9

Bruit : Simulation et optimisation sur processeur TM3270

Dans ce chapitre, nous présentons l'implémentation et l'optimisation du filtre bilatéral Bayer proposé dans le chapitre 7 sur le processeur multimédia TriMedia TM3270 de NXP Semiconductors. On utilise le système de compilation et de simulation TCS avec une fréquence de processeur de 350 MHz. Comme pour les algorithmes de dématriçage étudiés dans le chapitre 8, nous utilisons les techniques d'optimisations algorithmiques, standards et TriMedia. Nous imposons le même cahier des charges, traiter une image photographique de taille 5 méga-pixel en moins de 0,5 secondes et traiter aux moins 25 images de résolution VGA par seconde. Nous présentons d'abord l'optimisation du code par les techniques standards : utilisation de LUTs, déroulage de boucles, utilisation de types de données appropriés, etc. Nous présentons ensuite les optimisations obtenus par l'utilisation des fonctions du jeu d'instructions TriMedia. Nous présentons ensuite les résultats d'optimisations obtenus avec une nouvelle formulation du filtre bilatéral Bayer mieux adaptée pour l'architecture et pour le jeux d'instruction du processeur TriMedia TM3270.

9.1 Version naïve et pseudo-code de l'algorithme

La version naïve du code est la version sans optimisations. Cette version utilise des conditions « si » et « sinon », des opérations mathématiques complexes, comme la fonction exponentielle et aucunes optimisations de boucles. Le pseudo-code du filtre est présenté dans l'algorithme 12.

9.2 Optimisations standards

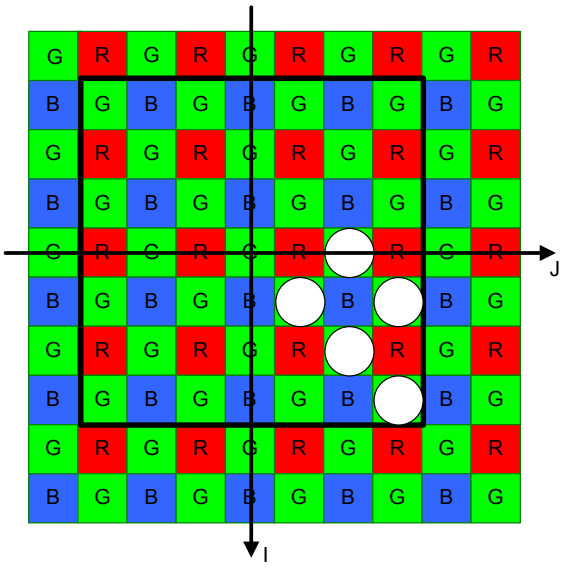
9.2.1 Utilisation de Look Up Table

Dans sa version gaussienne, le filtre bilatéral utilise deux appels de la fonction exponentielle pour déterminer les poids des pixels dans le calcul de la moyenne pondérée. Cette fonction de poids est dépendante de l'intensité du pixel et de sa position. Il est possible de remplacer l'appel de la fonction exponentielle par la lecture des valeurs pré-calculées de cette même fonction dans une LUT. L'équation 9.1 rappelle la formulation de la fonction de calcul des poids.

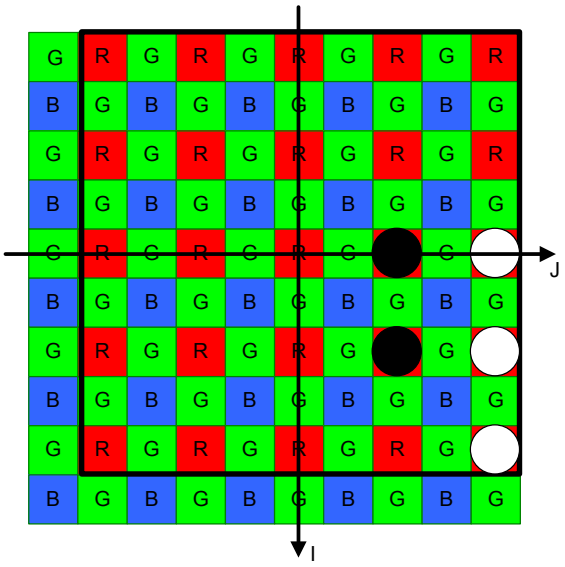
$$P(x,y) = exp\frac{-|x-y|}{\rho^2} exp\frac{-|u(x)-u(y)|}{h^2} \tag{9.1}$$

Dans cette équation, $P(x,y)$ montre le calcul du poids d'un pixel à la position de coordonnées 2D y pour le calcul de la nouvelle valeur du pixel à la position de coordonnées 2D x. Les données de l'image étant de type *unsigned char*, la LUT des poids d'intensités contient 256 valeurs d'exponentielles pré-calculées, correspondant aux 256 valeurs de différences possibles. Nous utilisons des poids de type *float*. La taille de la LUT des poids d'intensités est donc de $256 \times 4 = 1024$ octets. Pour construire la LUT des poids de positions, nous utilisons comme exemple la version du filtre utilisant la moyenne pondérées de 25 pixels pour chaque canal de couleurs, ce qui correspond à $n = 3$ dans la formulation du filtre (voir section 7.6). Cela correspond à l'utilisation d'un masque de taille 7×7 pour le filtrage des pixels verts et un masque de taille 9×9 pour le filtrage des pixels bleus et rouges. Dans ces masques, nous comptons 40 positions de points possibles, 24 positions de pixels verts dans le masque de taille 7×7, dans lesquelles sont déjà inclues les postions centrales des pixels rouges et bleus, et 16 positions périphériques pour les pixels rouges et bleus dans les masques de taille 9×9. On utilise donc une LUT de 40 valeurs de type *float* pour les poids de positions. Le compte du nombre de positions est illustré sur la figure 9.1.

Afin de réduire la taille de la LUT, nous pouvons utiliser les propriétés de symétries de la matrice de Bayer. Notons, que la position d'un pixel par rapport au pixel central est calculé en utilisant une distance euclidienne 2D, cela exclut toutes dépendances de direction dans le calcul du poids. Considérons maintenant une fenêtre de taille 7×7 centrée sur un point vert dans la matrice de Bayer. Considérons ce point vert comme l'origine d'un repère orthonormé à 2 dimensions (I, J), comme cela est illustré sur la figure 9.2). L'unité des axes I et J est normée à 1 entre deux centres de pixels. Un pixel vert peut donc prendre les coordonnées : 1, 2 ou 3 sur les axes I et J dans une fenêtre de taille 7×7. En utilisant les propriétés des calculs combinatoires, le nombre



(a) Compte de la position des pixels pour le cas vert

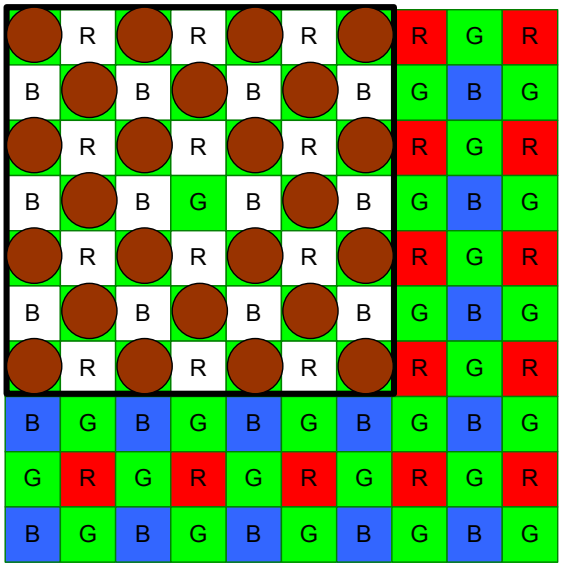


(b) Compte de la position des pixels pour les cas rouge et bleu

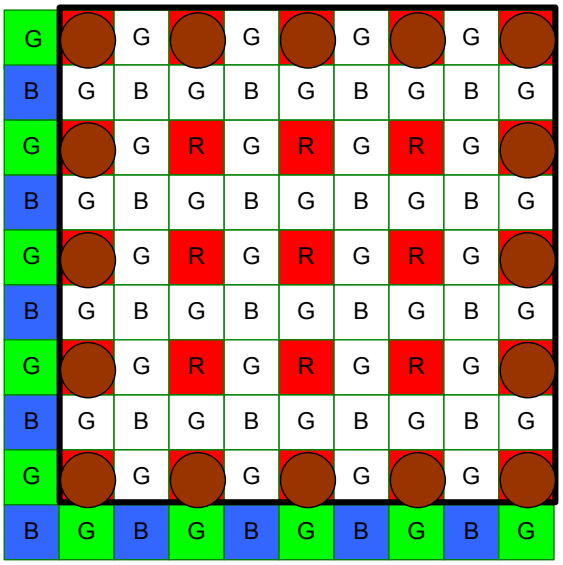
FIGURE 9.1 – Compte du nombre de positions dans les fenêtres du filtre bilatérale Bayer : (a) les positions des pixels verts comptés sont marqués par des disques marrons, (b) les positions des pixels rouges comptés sont marqués par des disques marrons, les positions déjà prises en compte pour le cas vert ne sont pas recomptées, l'échantillonnage des pixels bleus étant le même que celui des pixels rouge, ce cas prend aussi en compte les positions des pixels bleus ; on compte au totale 40 positions possibles.

de permutations sans répétitions de coordonnées 2D est : $(3)_2 = 6$. En excluant la coordonnée $(0,0)$, nous trouvons 5 distances euclidiennes uniques pour les poids de positions des pixels verts dans une fenêtre de taille 7×7. Ces pixels sont mis en évidence par des disques blancs sur la figure 9.2(a). En considérant maintenant la fenêtre de taille 9×9 utilisée pour la convolution des pixels rouges et bleus. En utilisant le même raisonnement, nous trouvons aussi 5 distances euclidiennes à 2 dimensions uniques. Nous

pouvons exclure de ce compte les cas déjà considérés pour le filtrage des pixels verts (cas de la fenêtre 7×7). Cela nous amène au compte de 3 distances euclidiennes uniques. Ces positions sont mises en évidence par des disques blancs sur la figure 9.2(b). Les distances communes avec le cas vert sont mises en évidence par des disques noirs. Finalement, toutes les distances euclidiennes possibles peuvent être stockées dans une LUT de 8 valeurs de type *float*, soit 8×4 octets.



(a) Compte de la position des pixels pour le cas vert



(b) Compte de la position des pixels pour les cas rouge et bleu

FIGURE 9.2 – Compte du nombre de positions des pixels dans les fenêtres du filtre bilatéral Bayer : (a) montre avec des disques blancs les positions élémentaires pour le cas de l'interpolation des points verts, (b) montre avec des points blancs les positions élémentaires utilisées pour l'interpolation des pixels rouges et bleus. Les disques noirs mettent en évidence les positions élémentaires communes entre les positions d'interpolations des pixels verts, rouges et bleus.

9.2.2 Fusion des LUTs

Pour réduire la complexité du calcul des poids, nous pouvons fusionner la LUT des poids d'intensités avec la LUT des poids de positions. On obtient ainsi 8 LUTs de 256 valeurs de type *float*. Soit une taille de LUTs de $256 \times 8 \times 4$ octets. En effet, l'utilisation séparée de ces deux LUTs conduit à deux lectures de valeurs et une multiplication pour le calcul d'un poids. Avec la fusion des LUTs, le calcul de la fonction de poids est réduit à la lecture d'une valeur dans une LUT.
Les tableaux 9.3 et 9.1 montrent les nombres d'opérations nécessaires pour traiter un pixel avec la version du filtre bilatéral Bayer naïve et avec l'utilisation des LUTs. L'utilisation des LUTs permet un nombre important de simplifications. On notera que les calculs des 50 exponentielles sont remplacés par 25 lectures dans une LUT. On note aussi un gain de 75 multiplications, 50 additions, et 50 divisions. Ces gains sont directement dûs aux pré-calculs des poids. Le résultat de l'exécution de ce code est présenté dans le tableau 9.4.

TABLE 9.1 – Complexité du filtre bilatéral Bayer en nombre d'opérations par pixel

Operations	Exponentielle	Multiplications	Additions	Divisions	Valeurs absolues	Racines carrés
poids de position	25	75	25	25	0	25
poids d'intensité	25	25	25	25	25	0
moyenne et normalisation	0	50	24	1	0	0
totale	50	150	74	51	25	25

TABLE 9.2 – Complexité du filtre bilatéral Bayer après l'utilisation des LUTs

Operations	Exponentielle	Multiplications	Additions	Divisions	Valeurs absolues	Lectures
poids de position	0	0	0	0	0	0
poids d'intensité	0	0	0	0	25	25
moyenne et normalisation	0	25	24	1	0	0
totale	0	25	24	1	25	25

9.2.3 Déroulage de boucles

La formulation du filtre bilatéral Bayer (équation 7.6) montre que le filtre possède une fenêtre de traitement dont la taille varie selon la couleur du pixel courant. Cela induit l'utilisation de structures de contrôles. La technique du déroulage de boucles peut partiellement éliminer les choix conditionnels introduits par l'arrangement des couleurs dans la matrice de Bayer. L'algorithme 1 illustre la boucle d'itération avant déroulage. L'algorithme 2 illustre la boucle d'itération après déroulage. On voit que les structures

de contrôles apparaissent à chaque ligne dans la version déroulée, alors qu'elles apparaissent à chaque pixel dans la version naïve. Le résultat de l'exécution de ce code est donné dans le tableau 9.4.

Algorithm 1 Version sans déroulage de boucles

width est la largeur de l'image
heigh est la hauteur de l'image
$fRed()$ est la fonction d'interpolation des pixels rouges
$fBlue()$ est la fonction d'interpolation des pixels bleus
$fGreen()$ est la fonction d'interpolation des pixels verts
actual_pixel_color est la couleur du pixel courant
pour $j = 0 ; j < heigh - 1 ; j++$ **faire**
  pour $i = 0 ; i < width - 1 ; i++$ **faire**
    si $actual\_pixel\_color == Red$ **alors**
      $Img[j * width + 1] = fRed();$
    sinon si $actual\_pixel\_color == Green$ **alors**
      $Img[j * n + 1] = fGreen();$
    sinon
      $Img[j * width + 1] = fBlue();$
    fin si
  fin pour
fin pour

9.3 Représentation des données appropriée et utilisation du jeu d'instructions TriMedia

La représentation du type des données a un impact important sur l'optimisation d'un code. Il a une influence sur le nombre de cycles par opération, l'utilisation de la mémoire cache et le stockage. Pour améliorer les performances d'exécution du code il est important d'utiliser la représentation des données la plus petite possible. Par exemple, travailler avec des valeurs de type *char* permet dans notre cas, de réduire la mémoire requise pour le stockage des LUTs en passant de 8192 octets de mémoire (8 LUTs de 256 valeurs de type virgule flottante) à 2048 octets(8 LUTs de 256 valeurs de type char). Le résultat de l'optimisation de représentation des données est montré dans le tableau 9.4.

D'autre part, il est préférable de travailler avec des valeurs entières pour exploiter au maximum les opérations dédiées du processeur TriMedia. En effet, celles-ci utilisent des registres d'opérandes de taille 4 octets, il est donc possible de représenter 1, 2 ou 4 pixels dans chaque registre suivant la représentation des données utilisée (8, 16, 32 octets). Ces instructions permettent au processeur de calculer plus de quatre opérations parallèlement, réduisant ainsi efficacement le nombre de cycles effectués. La figure 9.3, montre un exemple d'opération dédiée avec l'utilisation de la fonction *dspuquadbssub*.

Algorithm 2 Version avec déroulage de boucles

```
width est la largeur de l'image
heigh est la hauteur de l'image
fRed() est la fonction d'interpolation des pixels rouges
fBlue() est la fonction d'interpolation des pixels bleus
fGreen() est la fonction d'interpolation des pixels verts
actual_line est la ligne en
pour j = 0 ;j < heigh − 1 ;j + + faire
  si actual_line == Red_Green_line alors
    pour i = 0 ;i < width − 1 ;i = i + 4 faire
      Img[j ∗ width + i] = fRed() ;
      Img[j ∗ width + i + 1] = fGreen() ;
      Img[j ∗ width + i + 2] = fRed() ;
      Img[j ∗ width + i + 3] = fGreen() ;
    fin pour
  sinon
    pour i = 0 ;i < width − 1 ;i = i + 4 faire
      Img[j ∗ width + i] = fGreen() ;
      Img[j ∗ width + i + 1] = fBlue() ;
      Img[j ∗ width + i + 2] = fGreen() ;
      Img[j ∗ width + i + 3] = fBlue() ;
    fin pour
  fin si
fin pour
```

Nous utilisons cette fonction pour calculer la place du poids d'un pixel dans les 8 LUTs en fonction de sa position et de son intensité. Cette fonction reçoit comme entrée deux registres de taille 4 octets qui contiennent chacun 4 mots (pixels) de taille 1 octet, elle retourne le résultat de quatre valeurs absolues de différences dans un registre de taille 4 octets. D'autre part, nous utilisons la fonction *super_ld*32 pour charger des données consécutives de taille 8 octets dans deux registres de destination de taille 4 octets. Le tableau 9.3 illustre le nombre de cycles nécessaires au processeur pour effectuer le traitement de 4 pixels et les gains obtenus. Les résultats de l'utilisation de fonctions dédiées du processeur sont montrés dans le tableau 9.4.

9.4 Résultats expérimentaux

Dans cette section, nous présentons les résultats obtenus avec les optimisations présentées. Chaque étape d'optimisation est ajoutée à l'étape qui la précède. Pour la simulation et la génération des rapports de performances, nous utilisons le système de simulation TriMedia TCS. Les exécutions des codes sont simulés avec les paramètres du processeur TM3270 pour une fréquence de 350 MHz. Pour mesurer les optimisations apportées par chaque étape, nous mesurons : le nombre de cycles d'horloge nécessaire pour traiter un

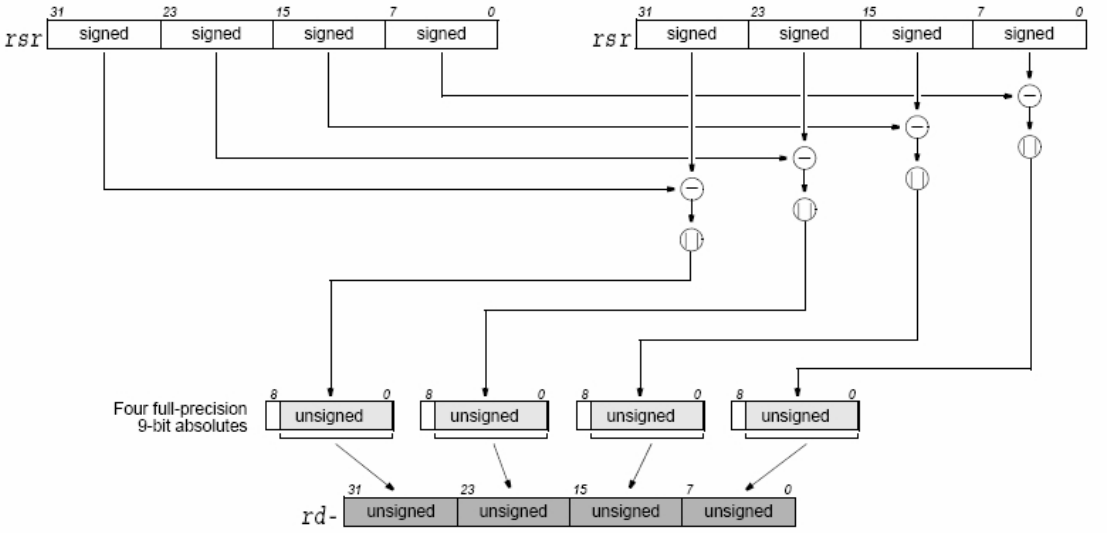


FIGURE 9.3 – Fonctionnement de la fonction *dspuquadbssub* : cette fonction reçoit deux registres de taille 4 octets qui contiennent chacun 4 pixels de taille 1 octet, elle retourne le résultat de quatre valeurs absolues de différences dans un registre de taille 4-octets. Nous utilisons la fonction *super_ld*32 pour charger des données consécutives de taille 8 octets dans deux registres de destination de taille 4 octets.

TABLE 9.3 – Gain de cycles obtenus avec l'utilisation des opérations TriMedia

			Nombre de cycles pour appliquer la fonction	Nombre de cycles pour traiter 4 pixels	Total
Chargement des données	Opération classique		5	20	20
	Fonction super_ld32()		5	10	10
Calcul de l'index dans la LUT	Opération classique	Soustraction	1	4	12
		Valeur absolue	2	8	
	Fonction dspquadabsub()			3	3

pixel, le pourcentage d'optimisation par rapport à la version naïve, le nombre d'images traitées par seconde avec une résolution VGA(640×480) et le temps nécessaire pour traiter une image de taille 5 mégas-pixels. Les résultats des optimisations sont montrés dans le tableau 9.4.

La première ligne du tableau 9.4 présente les résultats obtenus avec l'exécution de la version naïve. La première colonne montre les performance en nombre de cycles par pixel après chaque étape d'optimisation. La version naïve nécessite $6026,1$ cycles pour traiter un pixel, la version utilisant des Look-Up-table nécessite $445,7$ cycles, soit une optimisation de 92.6%. Le déroulage de boucle permet d'obtenir le traitement d'un pixel en 219.2 cycles, soit une optimisation de $96,36\%$ par rapport à la version naïve. L'utilisation de

TABLE 9.4 – Résultats des performances des étapes d'optimisation

	Cycles/pixel	Gain/Version naive %	Images/seconde (640 × 480)@350MHz	Temps pour traiter une image de taille 5 mégas-pixels
Version naive	60326, 1		0, 19	172, 36
Look Up Table	445, 7	92, 6	2, 56	6, 36
Déroulage de boucle	219, 2	96, 36	5, 2	3, 13
Type entier	155, 5	97, 42	7, 33	2, 22
Fonctions TriMedia	64, 1	98, 94	17, 78	0, 92

types de données entier permet de traiter un pixel en 155, 5 cycles, soit une amélioration de 97, 42% par rapport à la version naïve. Finalement, l'utilisation du jeux d'instruction TriMedia permet de traiter un pixel en 64, 1 cycles, soit une optimisation par rapport à la version naïve de 98, 94%, permettant de traiter 17.78 images par seconde avec une résolution VGA et de traiter une image de taille 5 mégas-pixels en 0, 92 secondes. Ces résultats nous montrent qu'en utilisant toutes les optimisations proposées, le cahier des charges n'est pas respecté. On cherche donc à trouver une nouvelle formulation du filtre mieux adaptée aux capacités de traitement du processeur TM3270. Analysons d'abord les problèmes d'adéquation entre l'algorithme et les fonctions proposées par la librairie TriMedia :

– Le premier problème est la taille du masque de traitement. En effet, charger des lignes de 7 et 9 pixels n'est pas adapté pour une utilisation efficace de la fonction *super_ld*32. Cette fonction permet de charger 8 octets, soit 8 pixels dans notre cas. Charger une ligne de 9 pixels nécessite donc l'utilisation de deux fonctions *super_ld*32 pour seulement un pixel supplémentaire. Il serait donc préférable d'utiliser un masque de traitement contenant 8 pixels par ligne.
– Le second problème est la non-efficacité d'utilisation des valeurs chargées. Traiter les pixels un par un réduit considérablement l'utilisation des données chargées lors de l'utilisation de la fonction *super_ld*32. En effet, à cause de l'arrangement des couleurs dans la matrice de Bayer, 50% des valeurs chargées ne sont pas utilisées. Il faut donc trouver une méthode permettant d'utiliser toutes les valeurs de pixels chargées, en prenant en compte l'arrangement des couleurs dans la matrice de Bayer.

Dans la partie suivante, nous proposons une nouvelle formulation du filtre bilatéral Bayer. Cette nouvelle formulation prend en compte les remarques précédentes pour établir une meilleure adéquation entre l'algorithme et le jeu d'instructions du processeur TriMedia. Cette adéquation permettra d'augmenter l'optimisation des performances d'exécution de l'algorithme.

9.5 Nouvelle formulation du filtre bilatéral Bayer dédiée au processeur TM3270

On prenant en compte les remarques de la section précédente, nous énonçons une nouvelle formulation du filtre bilatéral Bayer :

- Afin d'optimiser les appels de la fonction de chargement *super_ld*32, nous imposons un masque de convolution de taille 8×8. Cette configuration permet le chargement d'une ligne complète de la fenêtre avec l'utilisation d'une seule instruction de chargement. De cette manière, toute la fenêtre peut être chargée avec l'utilisation de 8 fonctions *super_ld*32. Nous utilisons deux registres de taille 32 octets pour stocker les pixels de type *unsigned char* (4 valeurs de pixel sont stockées dans chaque registre). Cela permet l'utilisation d'instructions à 4 opérandes de la bibliothèque TriMedia (la fonction *dspuquadabssub()* par exemple).
- Pour optimiser l'utilisation des valeurs chargées nous imposons d'interpoler un groupe de pixels rouge, vert, bleu simultanément. Pour ce faire, nous pouvons tirer avantage de l'arrangement des couleurs dans la matrice de Bayer. Comme on peut le voir sur la figure 9.4, la matrice de Bayer est composée de la répétition d'un élément E de taille 2×2, cet élément inclut 1 pixel rouge, 1 pixel bleu et 2 pixels verts. Nous proposons d'interpoler tout les points de cet élément à chaque itération. Ce fonctionnement permet à la fois d'utiliser toutes les valeurs chargées de la fenêtre et permet une utilisation optimisée de la fonction *super_ld*32. Cette méthode permet aussi une utilisation efficace des opérations TriMedia en éliminant les opérations de tri des données.

La nouvelle fenêtre de convolution de taille 8×8 et son noyau de convolution sont montrés sur la figure 9.4.

La figure 9.6 illustre le calcul de la différence de la valeur absolue des pixels utiles pour la lecture du poids de pondération dans la LUT. $RgRgXX$ et $GBGBXX$ représentent un couple de données chargées de 4 octets. $RgRg0$ et $GBGB0$ représentent le noyau de la fenêtre de convolution. Pour le calcul des poids des pixels, les valeurs du noyaux E sont chargées, dupliquées et concaténées dans les deux registres $RgRg0$ et $GBGB0$. Les calculs des valeurs absolues des différences d'intensités des pixels utilisés pour le calcul des poids dans la moyenne pondérée du filtre bilatéral sont fait directement en utilisant la fonction *dspuquadabssub()*. En utilisant ce fonctionnement, toutes les données chargées sont utilisées permettant ainsi une optimisation maximale de l'utilisation de la fonction de chargement des données *super_ld*32 et de la fonction *dspuquadabssub()* sur les données chargées.

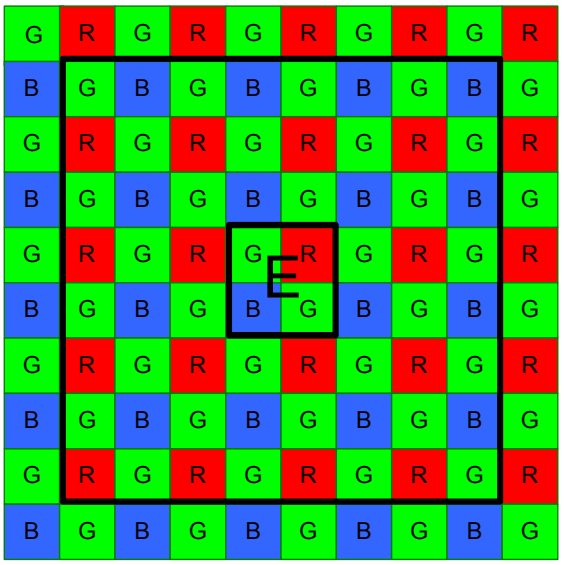


FIGURE 9.4 – Présentation de la nouvelle fenêtre de convolution du filtre bilatéral Bayer et son noyau E.

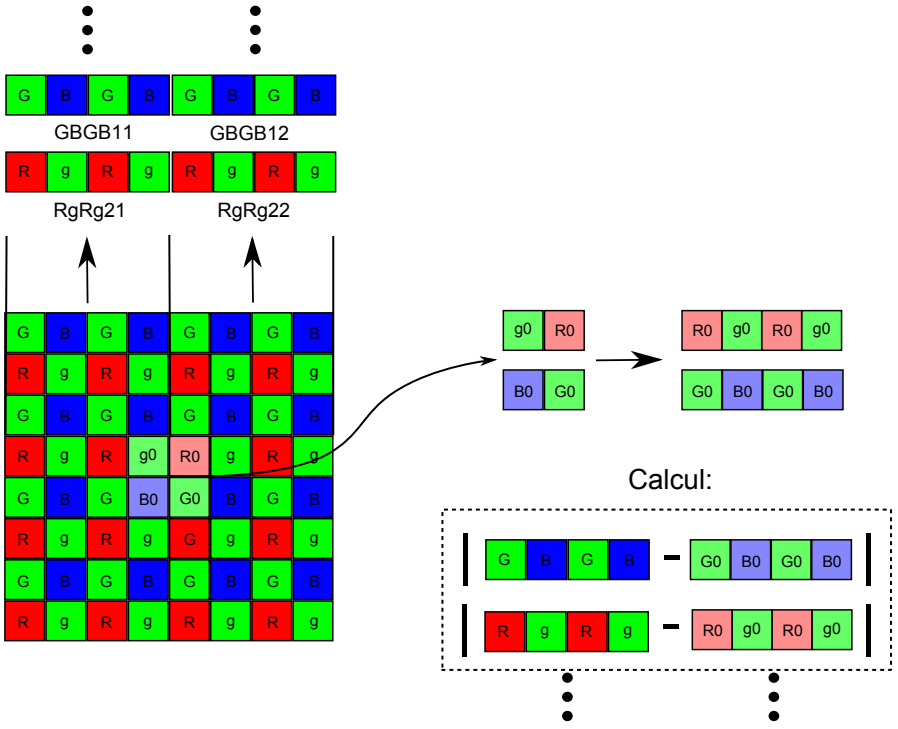


FIGURE 9.5 – Schéma de la nouvelle formulation du filtre bilateral Bayer

9.5.1 Résultats de la simulation

Dans cette partie, nous présentons les performances d'optimisations obtenues avec la nouvelle formulation du filtre bilatéral Bayer. La simulation est faite avec l'environnement TCS en utilisant les caractéristiques du processeur TM3270. Comme mesure d'optimisation nous utilisons le compte du nombre de cycles d'horloge, le % d'amélioration

comparé à la version naïve et le nombre d'images traitées par secondes à la résolution VGA, la fréquence du processeur est fixée à 350MHz. Pour le filtrage, nous utilisons un masque de taille 8×8 comme cela est expliqué dans la section précédente. Le déroulage de boucle est implicitement introduit par l'utilisation d'un noyau de taille 2×2 et explicitement par le traitement de 2 noyaux à chaque itération. Nous utilisons le système de LUT présenté dans la section 9.2.2 et nous travaillons avec des données de type *unsigned char*. Nous avons aussi expérimenté une autre version de cette formulation en utilisant un masque de traitement de taille 6×6. Les résultats des optimisations sont présentés dans le tableau 9.5 et sur la figure 9.6. On peut voir que la nouvelle formulation permet d'améliorer la rapidité d'exécution de l'algorithme de 49,61% comparé à la première version optimisée, permettant ainsi de traiter 35,27 images de résolution VGA par seconde et de traiter une image de résolution 5 mégas-pixels en 0,46 secondes. En utilisant un masque de taille 6×6 nous arrivons à obtenir une amélioration de 61,93% par rapport à la première version opimisée, permettant ainsi d'atteindre un traitement de 46,9 images par seconde en résolution VGA et de traiter une image de résolution 5 mégas-pixels en 0,35 secondes.

TABLE 9.5 – Résultats des performances des étapes d'optimisation de la nouvelle formulation du filtre bilatéral Bayer

	Cycles/pixel	Gain/Version naive %	Gain/Version avec LUT	Images/seconde (640 × 480)@350MHz	Temps pour traiter une image de taille 5 mégas-pixels
Version naive	60326, 1			0, 19	172, 36
Look Up Table	445, 7	92, 6		2, 56	6, 36
Déroulage de boucle	219, 2	96, 36	50, 82	5, 2	3, 13
Type entier	155, 5	97, 42	65, 12	7, 33	2, 22
Fonctions TriMedia	64, 1	98, 94	85, 62	17, 78	0, 92
Nouvelle formulation avec une fenêtre de taille 8×8	32, 3	99, 46%	92, 75%	35, 27	0, 46
Nouvelle formulation avec une fenêtre de taille 6×6	24, 3	99, 6%	94, 55%	46, 9	0, 35

9.6 Conclusion

Dans ce chapitre, nous avons simulé et optimisé l'exécution du filtre bilatéral Bayer sur le processeur TM3270 en utilisant le système de compilation TriMedia avec un cahier des charges imposant de traiter une image de résolution 5 méga-pixels en moins de 0, 5

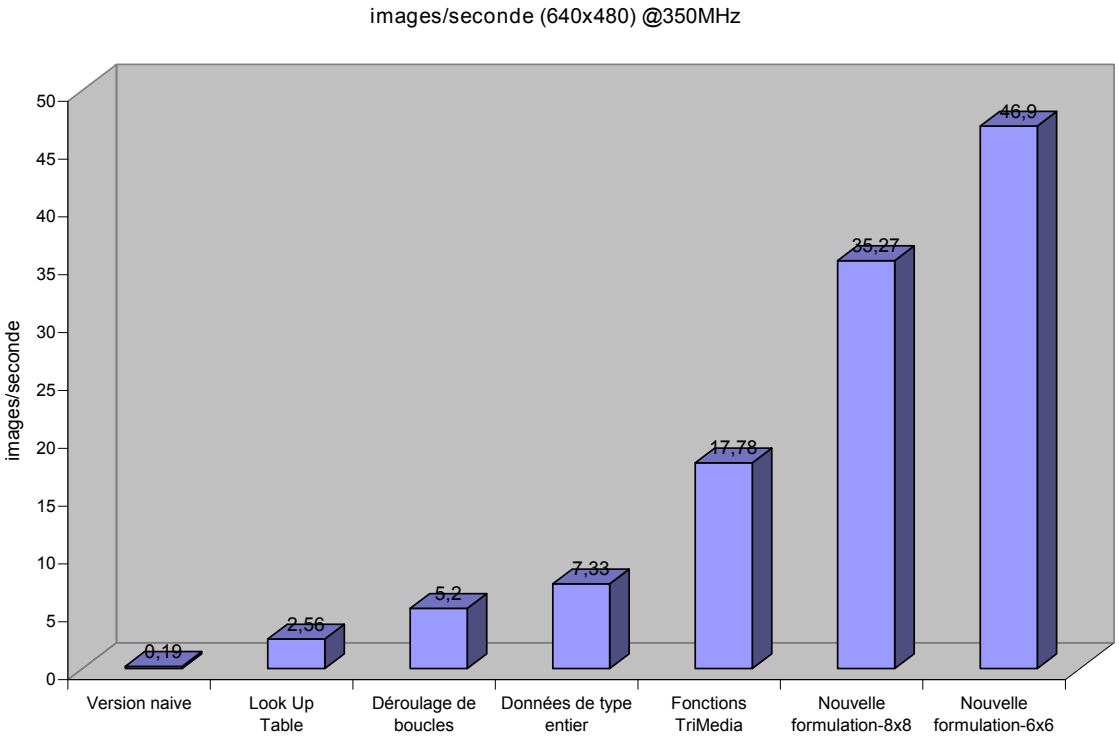


FIGURE 9.6 – Histogramme du nombre d'images de résolutions VGA traitées par seconde pour les différentes optimisation et formulations du filtre utilisées.

secondes et de traiter au minimum 25 images de résolution VGA par seconde. Nous avons utilisé des techniques d'optimisations générales comme l'utilisation de LUTs, le déroulage de boucles, la représentation adaptée du type des données. Nous avons aussi utilisé les fonctions du jeu d'instructions TriMedia. Malgré l'utilisation de toutes ces optimisations, il est impossible de respecter le cahier des charges en utilisant la formulation originelle de l'algorithme, avec un nombre d'images traitées par seconde de 17.78 et une image de taille 5 méga-pixels traitée en $0,92$ secondes. En analysant le code de l'algorithme et les propriétés des fonctions de la bibliothèque Trimedia, nous avons constaté que la formulation originelle du filtre bilatéral pour la matrice de Bayer ne permet pas une utilisation optimisée de la fonction *super_ld*32. Nous avons aussi constater qu'elle ne permet pas une utilisation efficace des données chargées (seulement 50% des données chargées sont utilisées). Pour résoudre ces déffisciences, nous avons proposé une nouvelle formulation du filtre bilatéral Bayer dédiée au processeur TM3270. Cette formulation utilise une fenêtre de convolution de taille 8×8 permettant une utilisation optimale de la fonction *super_ld*32. On utilise un noyau central carré de taille 2×2, l'utilisation de ce noyau permet une utilisation optimale des valeurs chargées. En implémentant cette nouvelle formulation du filtre bilatéral Bayer, nous atteignons un traitement en temps réel pour des images de résolutions VGA, avec 35,27 images traitées par seconde et une image de résolution 5 méga-pixels traitée en 0,46 secondes, pour la version utilisant un masque de taille 8×8. On obtient un traitement de 46,9 images par seconde et une

image de résolution 5 méga-pixels traitée en 0,35 secondes pour la version utilisant un masque de taille 6×6. Ces formulations apportent respectivement des gains de 99,46% et 99,6% par rapport à la version naive du filtre et permettent de respecter le cahier des charges. **Les travaux de ce chapitre ont fait l'objet d'une publication [19] à IST/SPIE2009**.

Chapitre 10

Implémentation d'une architecture dédiée pour l'algorithme GEDI

Dans le chapitre 5, nous avons proposé un nouvel algorithme de dématriçage pour la matrice de Bayer : GEDI. Cet algorithme allie une qualité d'image produite sans artéfacts de reconstruction avec une faible complexité algorithmique. Dans le chapitre 8, nous avons implémenté, simulé et optimisé GEDI sur le processeur TriMedia TM3270. Nous avons obtenu des temps d'exécutions de $50,5$ images par seconde pour une résolution VGA et $0,23$ secondes pour traiter une image de résolution 5 méga-pixels.

Les architectures de traitements présentes dans les systèmes mobiles de captures d'images ne possèdent pas toujours des processeurs multimédias de la puissance du processeur TriMedia TM3270. Aussi pour des raisons de rapidité d'exécution, pour minimiser la consommation d'énergie et pour une utilisation standardisée, nous proposons de développer un circuit intégré de type ASIC (Application Specific Integrated Circuit) dédié à l'algorithme GEDI. Le développement est réalisé en langage VHDL (VHSIC(Very High Speed Integrated Circuit) Hardware Description Language) en utilisant le logiciel de développement ISE WebPACK de Xilinx.

Pour des raisons de coût et d'espace (miniaturisation), on rappelle que la surface de silicium utilisée pour l'implémentation doit être la plus petite possible. La quantité de mémoire utilisée sur le circuit doit donc être minimale. Cette contrainte d'espace mémoire est directement opposée à la contrainte de rapidité de traitement des données. En effet, plus on augmente la capacité à traiter un volume d'informations important dans le circuit, plus la rapidité d'exécution de l'algorithme augmente, mais plus la surface de

silicium utilisée augmente. Il est donc important de trouver le bon compromis entre ces deux aspects. On se place dans le cadre de la photographie et de la vidéo numérique, on impose un traitement des images de résolution 5 méga-pixels en moins de $0,5$ secondes et un traitement d'au moins 25 images de résolution VGA par seconde.

10.1 Vue d'ensemble de l'architecture proposée

On rappel le fonctionnement de l'algorithme de dématriçage GEDI. Le pseudo-code de l'algorithme est présenté dans l'algorithme 6. L'algorithme est appliqué en 5 étapes :

1. interpoler le plan vert verticalement et le plan vert horizontalement
2. utiliser l'estimateur présenté dans la section 5.1 pour estimer si la direction d'interpolation est verticale ou horizontale ;
3. corriger les erreurs de choix d'interpolation par la méthode de LMDC (voir section 5.1.3) ;
4. interpoler le plan vert avec les directions d'interpolations choisies ;
5. interpoler les plans rouge et bleu par la méthode de constance des teintes (voir section 4.2) ;

Nous proposons une architecture divisée en quatre fonctions. L'enchaînement des fonctions est illustré sur la figure 10.1. Dans notre cas, les données issues du capteur sont de taille 8-bits :

– La première fonction, appelée « H&V green interpolation », a pour rôle de calculer l'interpolation verticale et l'interpolation horizontale des pixels verts manquants. Le flux d'entrée de cette fonction arrive directement du capteur : c'est le flux de pixels de la matrice de Bayer, on l'appelle $Bayer$. Cette fonction délivre trois flux de pixels : $Bayer$, G_h le flux de pixels verts interpolés horizontalement et G_v le flux de pixels verts interpolés verticalement.
– La deuxième fonction, appelée « Interpolation direction », calcule l'estimation de la direction d'interpolation pour chaque pixel manquant dans le plan vert. Cette estimation utilise l'estimateur proposé dans l'équation 5.1. La fonction reçoit trois flux de pixels : $Bayer$, G_h et G_v et elle délivre trois flux de pixels, $Bayer$, G_h, G_v et un flux de choix de directions d'interpolations D.
– La troisième fonction, appelée « Correction direction », corrige la direction d'interpolation estimée dans la fonction « interpolation direction » en utilisant la méthode de correction par LMDC proposée dans la section 5.1.3. Le choix de direction d'interpolation d'un pixel vert est remplacé par le choix de direction d'interpolation majoritaire

calculé dans un masque de taille $n \times n$ et centré sur le pixel à traiter. Dans notre cas, nous choisissons : $n = 3$. Cette fonction reçoit en entrée, les quatre flux de pixels sortant de la fonction « Interpolation direction » et fournit deux flux de pixels *Bayer* et *Green* le flux des pixels verts interpolés.

– La quatrième fonction est appelée « R&B interpolation », elle a pour but de reconstruire les pixels rouges et bleus manquants à partir du plan vert interpolé. Elle utilise la méthode d'interpolation par constance des teintes présentée dans la section 4.2. Cette fonction reçoit en entrée les flux de pixels *Bayer* et *Green* et délivre en sortie les flux de pixels *Red*, *Green* et *Blue*, respectivement les flux de pixels rouges, verts et bleus, formant les vecteurs couleurs des pixels de l'image reconstruite.

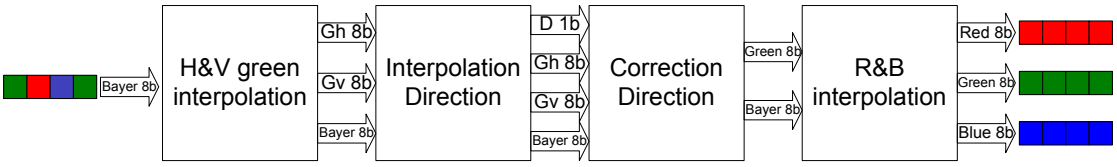


FIGURE 10.1 – Architecture générale de l'algorithme GEDI, elle est composée des quatre fonctions : H&V interpolation, Interpolation direction, Correction direction et R&B interpolation.

Dans les sections suivantes, nous détaillons l'architecture et le fonctionnement de ces différentes fonctions.

10.2 Fonction « H&V green interpolation » : interpolation horizontale et verticale du plan vert

Le rôle de cette fonction est d'interpoler les pixels verts manquants dans les directions horizontale et verticale. Le schéma de cette fonction est présenté sur la figure 10.2. Elle est divisée en deux composants : un composant de chargement du masque de traitement « 5×5 Window » et un composant de calculs d'interpolations « H&V computation ». L'algorithme GEDI s'applique dans un masque de taille 5×5, centré sur le pixel à traiter. On rappelle, que les données du capteur sont délivrées ligne par ligne. Il est donc nécessaire de construire un masque de taille (5×5) à partir du flux de données du capteur. C'est le rôle du premier composant, dont l'architecture et le fonctionnement sont détaillés dans la section 10.2.1. Tous les pixels de la fenêtre sont ensuite envoyés vers le composant « H&V computation » pour calculer les interpolations verticales et horizontales des pixels verts manquants. L'architecture et le fonctionnement de ce composant sont décrits dans la section 10.2.2.

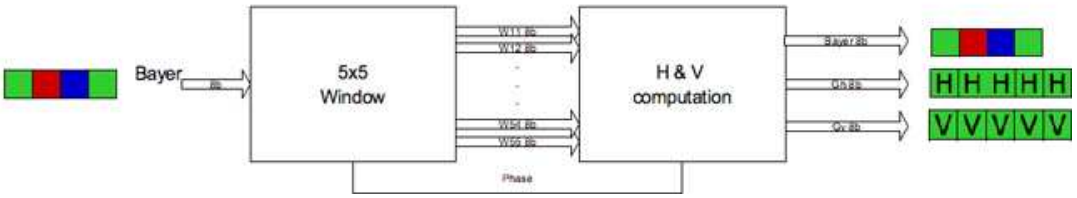


FIGURE 10.2 – Architecture de la fonction « H&V green interpolation ».

10.2.1 Composant « 5×5 Window » : chargement du masque de traitement

L'architecture de ce composant est présenté sur la figure 10.3. Le masque de taille 5×5 est composé de 25 registres permettant de stocker les 25 pixels du masque. La taille des registres est déterminée par la taille des données délivrées par le capteur. Dans notre cas, les pixels sont codés sur 8bits. Quatre FIFO (First In, First Out) sont utilisées pour sauvegarder et séquentialiser les données restantes des lignes de l'image. La taille t_F d'une FIFO est définie par la largeur M de l'image à laquelle on soustrait la largeur m du masque, multiplié par la taille t_D des données, soit : $t_F = (M - n) \times t_D$ bits. Le fonctionnement de cette architecture est intuitif. Les données issues du capteur (flux *Bayer*) arrivent dans le premier registre W_{55}. Les pixels se décalent ensuite à chaque coup d'horloge à travers les FIFO et les registres, jusqu'au remplissage total des registres (jusqu'à la position W_{11}). Le temps de latence du chargement de ce masque est donc égal au nombre de pixels chargés pour remplir tous les registres du masque. On calcule $n - 1$ lignes multipliées par la largeur de l'image M, auxquelles on ajoute n la largeur du masque, pour charger les n derniers pixels, soit : $L_{11} = M \times (n - 1) + n$ cycles. Ici le masque est de taille 5×5, soit $L_{11} = M \times 4 + 5$ cycles. Une fois que les registres sont pleins, le masque est ensuite déplacé vers la droite dans l'image à chaque coup d'horloge, un temps de latence de n (largeur du masque) cycles est nécessaire lorsque le masque arrive en fin le ligne, pour recadrer le masque sur le $(n - 1)/2 + 1$ ième pixel de la ligne suivante. La mémoire utilisée dans les FIFO est $A_1 = 4 \times (M - 5) \times 8$ bits. Le nombre d'éléments séquentiels utilisés dans ce composant est $E_{11} = 5 \times 5 \times 8 = 200$.

10.2.2 Composant « H&V computation » : interpolations verticales et horizontales des pixels verts

L'architecture du composant « H&V interpolation » est présentée sur la figure 10.4. Les calculs effectués pour les interpolations horizontales et verticales des pixels verts sont les mêmes, seules les données d'entrées changent (voir section 4.3). On utilise les données $W_{35}, W_{34}, W_{33}, W_{32}, W_{31}$ pour le calcul de l'interpolation horizontale et les

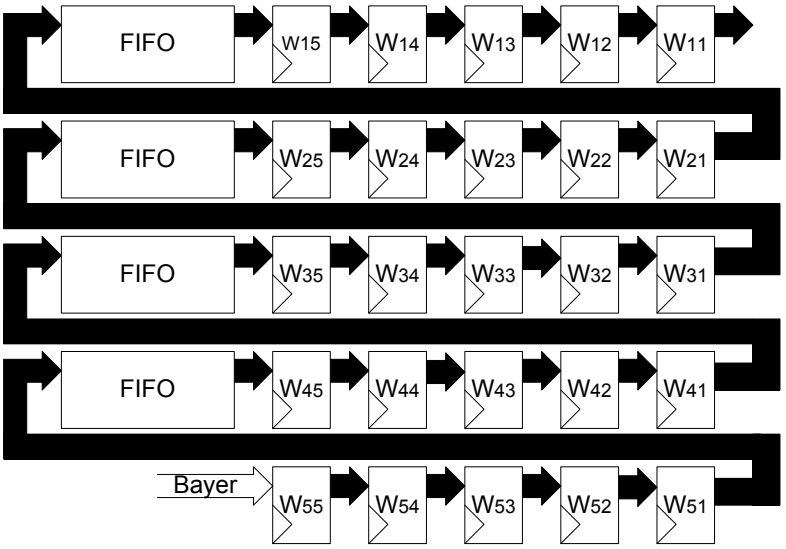


FIGURE 10.3 – Construction du masque de traitement : les pixels issus du capteur sont accumulés jusqu'à la dernière position du masque. Ils se décalent à chaque coup d'horloge vers la position suivante à travers les FIFO et les registres, jusqu'au remplissage total des registres. Une fois que ceux-ci sont pleins, le masque est ensuite déplacé vers la droite dans l'image à chaque coup d'horloge.

données $W_{53}, W_{43}, W_{33}, W_{23}, W_{13}$ pour le calcul de l'interpolation verticale. Le même composant « Green computation » est alors utilisé pour calculer les deux flux d'interpolations G_h et G_v. Cette opération est illustrée sur la figure 10.4. A partir des 25 pixels de la fenêtre, une ligne verticale est envoyée vers un premier composant pour calculer l'interpolation verticale G_v et une ligne horizontale est envoyée vers un second composant pour calculer parallèlement l'interpolation horizontale G_h.

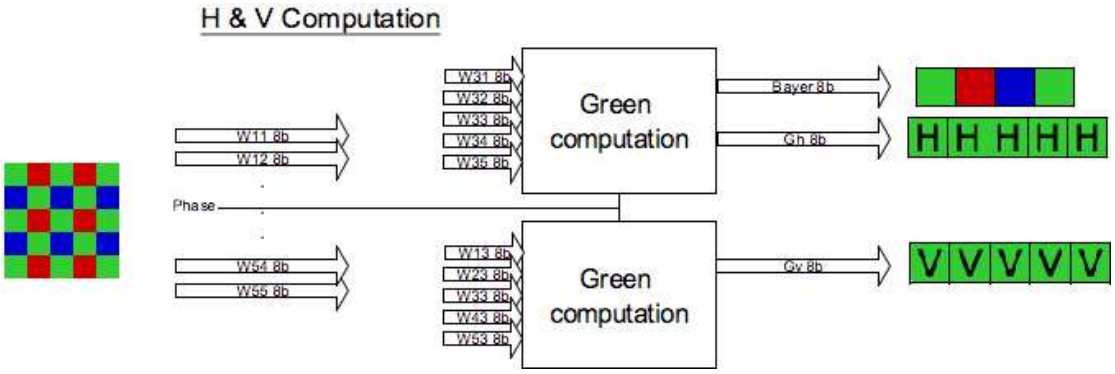


FIGURE 10.4 – Architecture du calcul des interpolations verticales et horizontales

Pour l'interpolation des pixels verts, on utilise l'équation 4.8. Le composant « Green computation » est illustré sur la figure 10.5. Il reçoit 5 flux de pixels en entrée, correspondant à la ligne où la colonne centrale dans le masque. Considérons les données w_1, w_2, w_3, w_4, w_5 représentant les valeurs de la ligne ou la colonne chargée pour interpoler le pixel vert à la position w_3. Si w_3 est déjà un pixel vert, alors l'interpolation n'est pas nécessaire et la valeur du pixel est synchronisée à travers le composant. Si w_3 est

un pixel rouge ou bleu, alors il faut calculer l'interpolation de la composante verte. Le calcul est séquentialisé en trois étapes. Considérons le cas suivant où le pixel central w_3 est rouge, les valeurs d'entrées sont donc R_1, G_2, R_3, G_4, R_5. Dans la première étape, on calcule $2G_2 - R_1$ et $2G_4 - R_5$. Les deux valeurs G_2 et G_4 sont multipliées par 2 par un décalage de bits à gauche, on leurs soustrait respectivement les valeurs R_2 et R_4 en utilisant un soustracteur. Les deux valeurs résultantes sont stockées dans deux registres r_1 et r_2, tels que $r_1 = 2G_2 - R_1$ et $r_2 = 2G_4 - R_5$. Parallèlement la valeur R_3 est séquentialisée et stockée dans un registre $r_3 = R_3$. La deuxième étape consiste à additionner les valeurs des registres r_1 et r_2 à travers un additionneur, le résultat est stocké dans un registre r_4, tels que $r_4 = 2G_2 + 2G_4 - R_1 - R_5$. La valeur de R_3 est séquentialisée et stockée dans un registre $r_5 = R_3$. Dans la troisième étape, la valeur de r_5 est multipliée par 2 en utilisant un décalage de bits à gauche, la valeur de r_5 est aussi pipelinée vers la sortie. Les valeurs des registres r_4 et $2 \times r_5$ sont additionnées dans un registre de taille 10 bits puis divisées par 4 en utilisant deux décalages de bits à droite, on obtient la valeur du pixel vert interpolé G_3 tels que $G_3 = (G_1 + G_2)/2 + (2R_3 - R_1 - R_5)/4$. Finalement, un multiplexeur fait de choix de transmettre la valeur interpolée ou la valeur de w_3 stockée dans le registre r_5 vers le registre r_6 en fonction du signal de phase, respectivement si w_3 n'est pas un pixel vert ou est un pixel vert. Dans tout les cas, le flux de la matrice de Bayer est synchronisé et stocké dans le registre r_7 pour être transmis vers la prochaine fonction. Finalement ce composant est séquentialisé en trois étapes, son temps de latence est donc $L_{12} = 3$. On compte la taille utilisée par les éléments séquentiels : on utilise 4 registres de taille 8-bits, 2 registres de taille 10-bits pour les soustractions, 1 registre de taille 11-bits pour l'addition et 2 registres de tailles 1-bit pour synchroniser la phase, E_{12} la taille totale des registres de ce composant est $E_{12} = 4 \times 8 + 2 \times 10 + 11 + 2 = 65$ bits.

10.2.3 Résumé

Le temps de latence total L_1 de cette fonction est égale à la somme des temps de latences de ses deux composants, soit $L_1 = L_{11} + L_{12} = M \times 4 + 5 + 3$ cycles. La mémoire RAM utilisée A_1 est égale à $A_1 = 4 \times (M - 5) \times 8$ bits. Le nombre d'éléments séquentiels utilisés est $E_1 = E_{11} + 2_{12} = 330$.

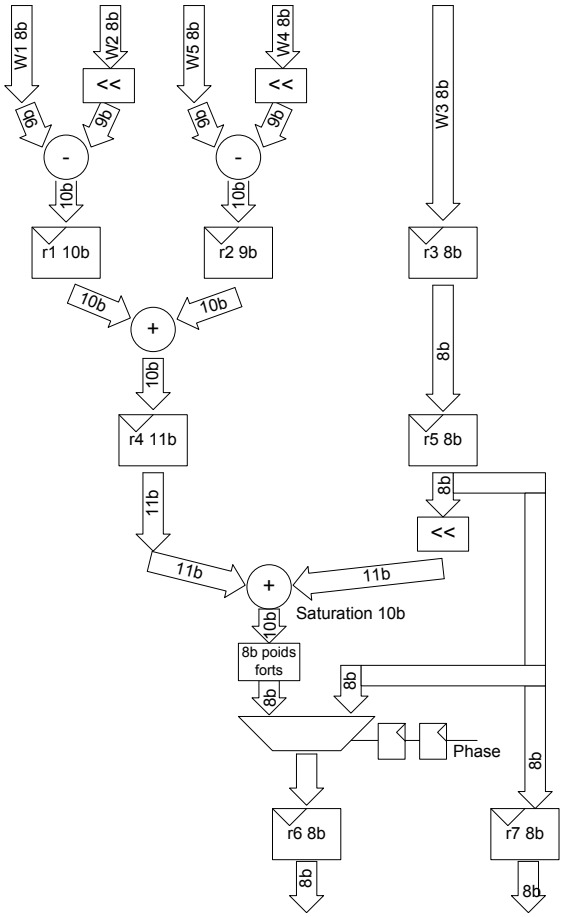


FIGURE 10.5 – Architecture du composant de calcul de l'interpolation des pixels verts manquants.

10.3 Fonction « Interpolation direction » : estimation de la direction d'interpolation

La fonction « Interpolation direction » a pour but d'estimer la direction d'interpolation des pixels verts manquants. Pour le calcul, on utilise l'équation 5.1 de l'algorithme GEDI. Cette fonction reçoit trois flux de pixel en entrée, $Bayer$, G_h et G_v. Elle délivre trois flux de pixels en sortie, $Bayer$, G_h et G_v et un flux de décision D. Son architecture est présentée sur la figure 10.6. Elle est divisée en trois composants : « 3×3 Window », « Delta computation » et « décision direction ».

10.3.1 Composant « 3×3 Window » : chargement du masque de traitement

Ce composant a pour but de charger une fenêtre de taille 3×3. Son architecture est illustrée sur la figure 10.9. Les trois flux G_v, G_h, et $Bayer$, sont d'abord concaténés dans un registre de taille 24-bits. Un masque de taille 3×3 est ensuite chargé à partir de ces

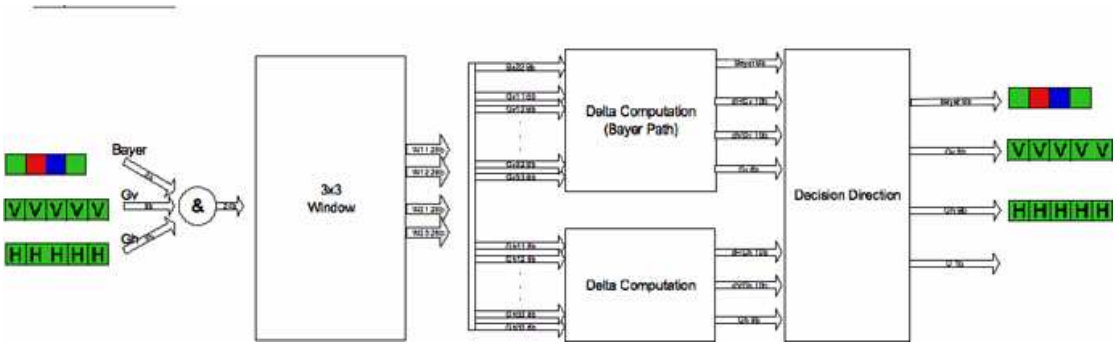


FIGURE 10.6 – Architecture de la fonction de direction d'interpolation et ses différents composants.

données pour disposer de toutes les informations nécessaires à l'estimation de la direction d'interpolation. Le masque est chargé de la même manière que le masque de la fonction « 5×5 Window » dans la section 10.2.1. Sa latence est donc $L_{21} = M \times (n-1) + n$, ici $n = 3$, soit $L_{21} = M \times 2 + 3$.

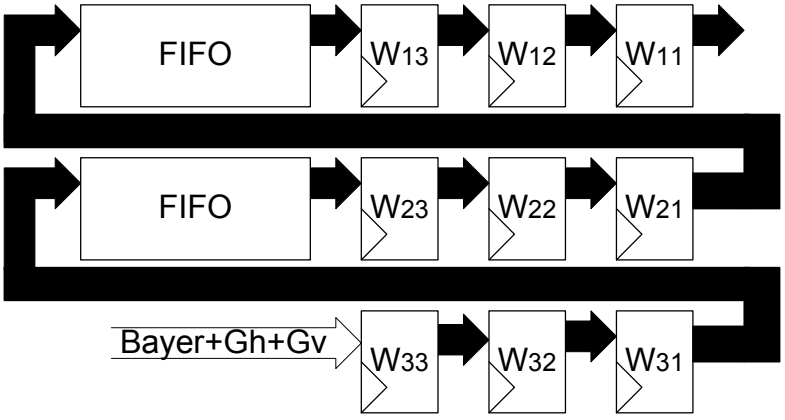


FIGURE 10.7 – Architecture du composant « 3×3 Window ».

10.3.2 Composant « Delta computation » : calcul des gradients

Le composant de calcul des gradients est nécessaire à l'estimation de la direction d'interpolation. Le calcul des gradients est décris dans l'équation 5.1. L'architecture du composant est illustré sur la figure 10.9. Il calcule parallèlement l'interpolation verticale et l'interpolation horizontale dans une fenêtre de taille 3×3. On utilise les notations du masque de la figure 10.9. Le composant reçoit en entrée les flux de pixels du masque et délivre en sortie les flux de gradient verticaux et horizontaux, respectivement dH et dV. Pour le calcul d'estimation, deux composants « Delta computation » sont utilisés parallèlement pour calculer les gradients du plan vert interpolé verticalement et les gradients du plan vert interpolé horizontalement. Ils délivrent ainsi les flux de données $dHGh$, $dVGh$, $dHGv$ et $dVGv$ respectivement les flux des gradients horizontaux et verticaux du plan vert interpolé horizontalement et les flux des gradients horizontaux et

verticaux du plan vert interpolé verticalement. Le composant « Delta computation » est séquentialisé en deux étapes, son temps de latence est donc $L_{22} = 2$.

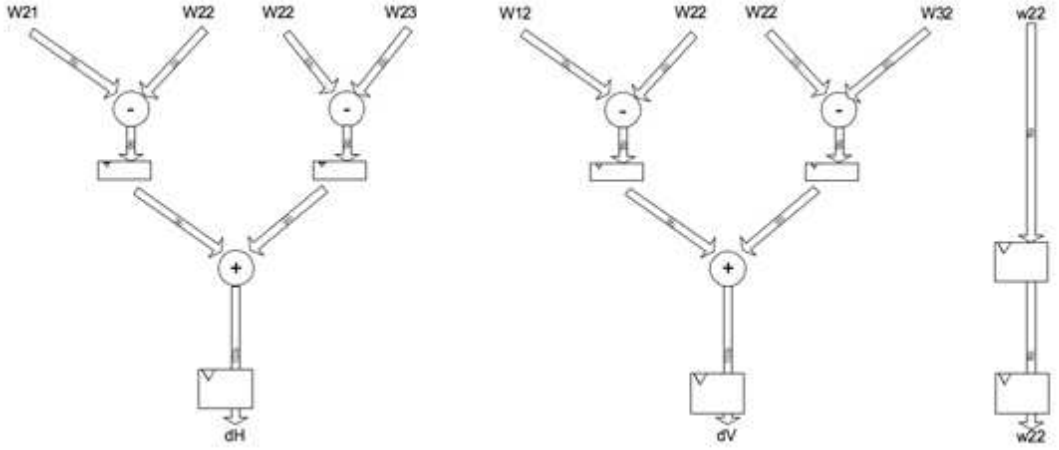


FIGURE 10.8 – Architecture du composant « Delta computation »

.

10.3.3 Composant « Decision Direction » : Choix de la direction d'interpolation

Ce composant estime les directions d'interpolations des pixels verts manquants. Son architecture est illustrée sur la figure 10.9. Il reçoit les flux des deux paires de gradients calculés dans l'étape précédente : $dHGh$, $dVGh$, $dHGv$ et $dVGv$. Les gradients verticaux et horizontaux sont additionnés puis comparés entre eux. La direction d'interpolation choisie est celle dans laquelle le résultat du calcul du gradient est le plus faible. Le flux de décisions créé D et les flux $Bayer$, G_v et G_h synchronisés sont transmis vers la fonction suivante. Ce composant est séquentialisé en deux étape, son temps de latence est donc $L_{23} = 2$.

10.3.4 Résumé

Le temps de latence de cette fonction est égale à la somme des temps de latences de ses différents composants soit $L_2 = L_{21} + L_{22} + L_{23} = M \times 2 + 3 + 2 + 2$ cycles. La taille de la mémoire RAM utilisée est $A_2 = 2 \times (M - 3) \times 8$ bits. La taille des éléments séquentiels $E_2 = 444$ bits.

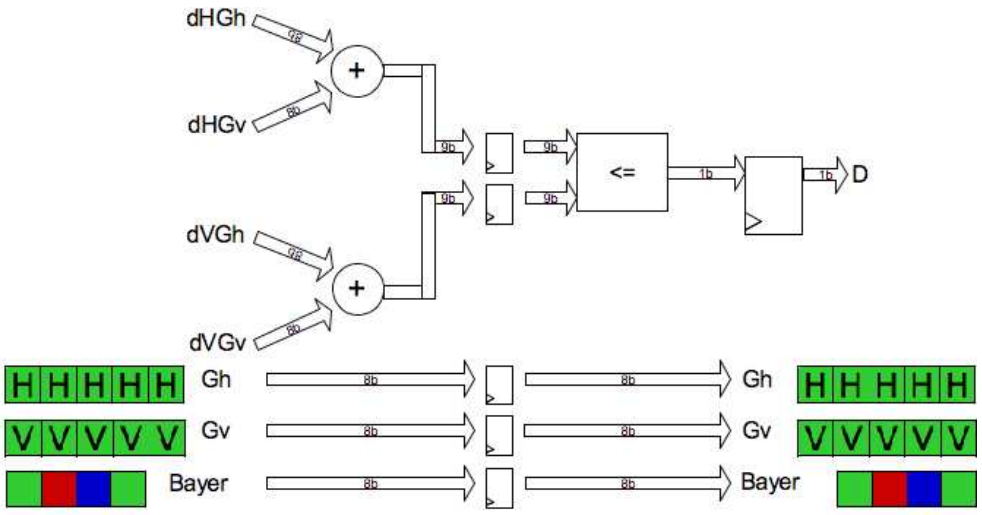


FIGURE 10.9 – Architecture du composant « Decision Direction ».

10.4 Fonction « Correction direction » : correction par LMDC

Maintenant que la direction d'interpolation est estimée pour chaque pixel du plan vert, nous allons corriger les erreurs d'interpolations en utilisant la méthode de LMDC (voir section 5.1.3). C'est le rôle de la fonction « Correction direction ». L'architecture du composant est illustrée sur la figure 10.10. Il reçoit trois flux de pixels G_h, G_v, et *Bayer* et le flux de directions D. Ces flux sont d'abord concaténés en un seul flux de taille 25-bits. Pour faire le calcul, on doit d'abord charger une fenêtre de taille 3×3, soit une latence $L_{31} = M \times 2 + 3$ cycles.

Le calcul d'homogénéité est effectué en comptant la somme S des valeurs des flux D dans le masque 3×3 ($D = 0$ ou $D = 1$ suivant la direction estimée). Suivant la valeur de S (supérieure ou inférieure ou égale à $S = (n - 1/2)$, soit $S = 12$ dans notre cas), la direction d'interpolation G_h ou G_v est choisie en utilisant un multiplexeur. Le résultat est transmis à travers le flux de pixel G. Le flux de *Bayer* est synchronisé à travers ce composant. Le temps de latence de ce calcul est de $L_{32} = 1$ cycle.

10.5 Résumé

Le temps de latence de cette fonction est égal au temps de latence du chargement du masque de taille 3×3 additionné au temps de latence de l'étape de calcul d'homogénéité, soit un temps de latence total $L_3 = M \times 2 + 3 + 1$ cycles. La taille de la mémoire RAM utilisée est $A_3 = 3 \times 2 \times (M - 3) \times (8 + 1)$ bits. Le nombre des éléments séquentiels utilisés est $E_3 = 266$.

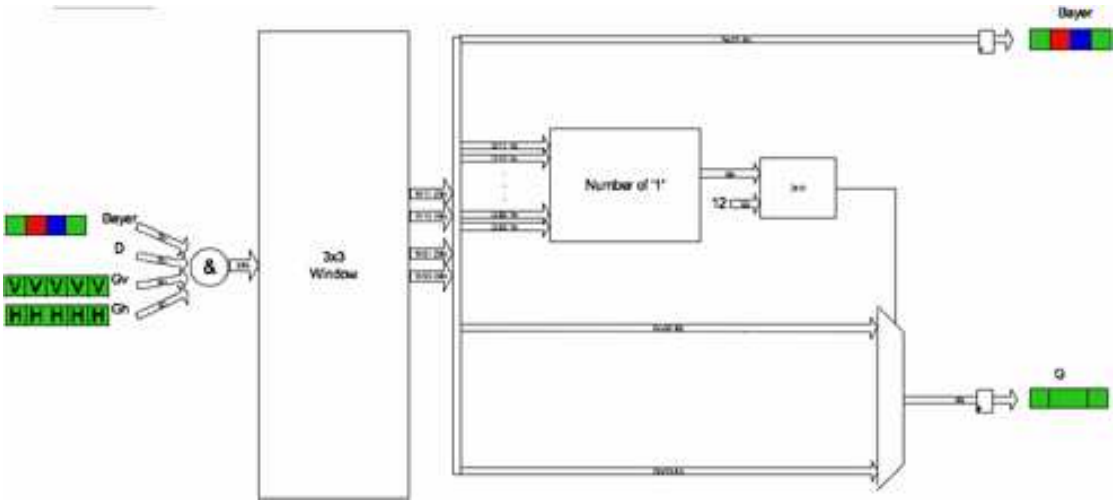


FIGURE 10.10 – Architecture du composant de la correction de la direction d'interpolation par LMDC (voir section 5.1.3).

10.6 Fonction « R&B interpolation » : interpolation des pixels rouges et bleus

Maintenant que le plan vert est entièrement reconstruit, la dernière étape consiste à interpoler le plan rouge et le plan bleu. C'est le rôle de la quatrième et dernière fonction « R&B interpolation », dont l'architecture est illustrée sur la figure 10.12. Cette fonction est constituée de deux composants. Le premier composant construit un masque de taille 3×3. Le deuxième composant calcul l'interpolation des pixels de couleurs rouges et bleus manquants. Le composant de construction de la fenêtre à une latence $L_{41} = M \times 2 + 3$.

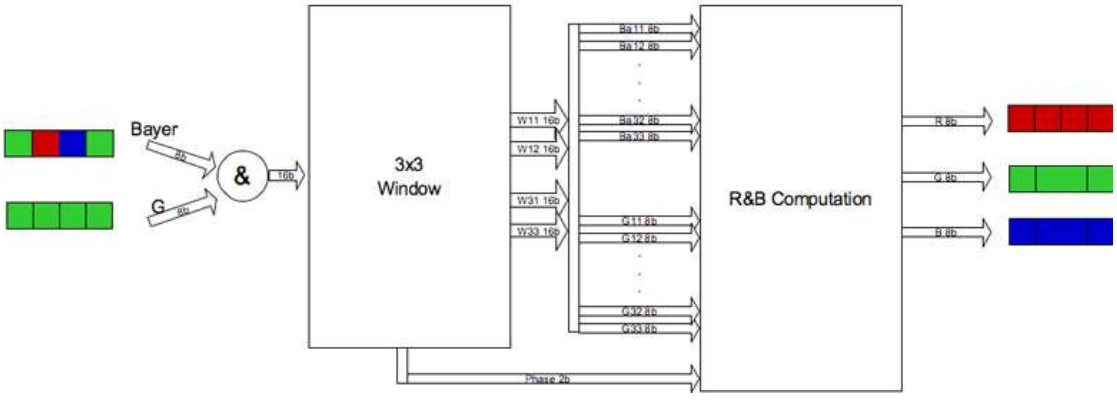


FIGURE 10.11 – Architecture de la fonction « R&B interpolation ».

10.6.1 Composant « R&B interpolation » : interpolations des pixels rouges et bleus

Pour cette étape, on utilise l'algorithme décrit dans la section 4.2. Ce composant est le plus complexe en termes de calculs, mais son fonctionnement est simple. Les interpolations sont calculées en fonction au signal de phase qui indique les différents cas d'interpolations possibles. Le pixel vert est retardé par deux niveaux de registres pour garder la synchronisation avec les sorties des pixels bleus et rouges. Ce fonctionnement est illustré sur la figure 10.12. Ce composant est séquentialisé en deux étapes, sa latence est donc $L_{42} = 2$.

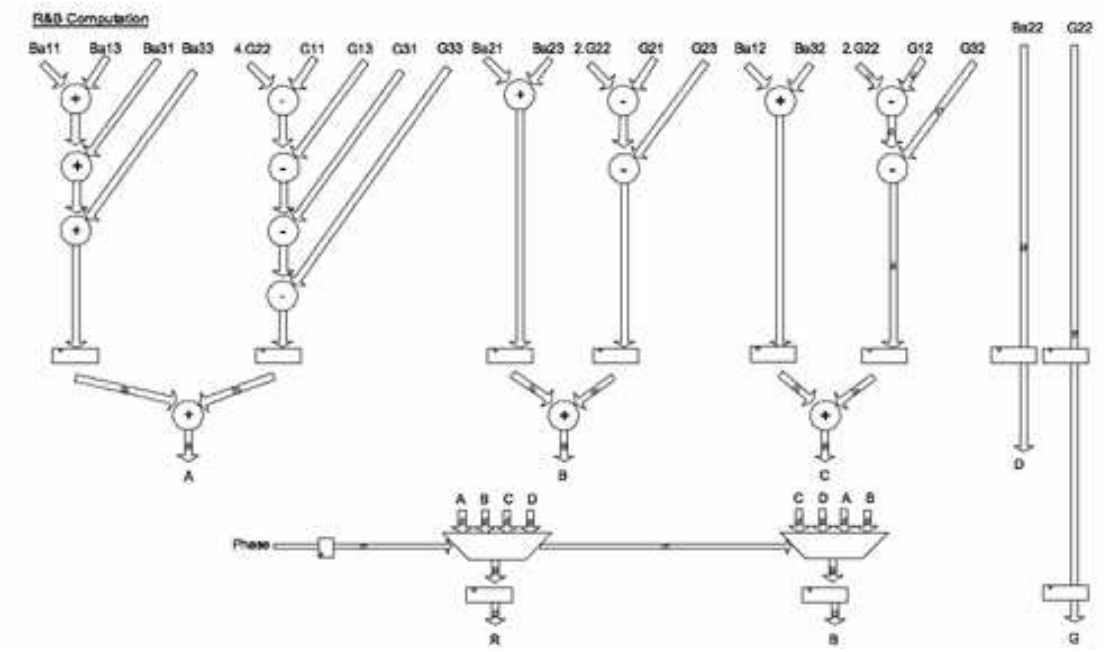


FIGURE 10.12 – Architecture du composants de calculs des pixels rouges et bleus

10.6.2 Résumé

Le temps de latence de cette fonction est égal au temps de latence de chargement du masque de taille 3×3 additionné au temps de latence de la fonction « R&B interpolation », soit $L_4 = L_{41} + L_{42} = M \times 2 + 3 + 2$. La taille de la mémoire RAM utilisée est $A_4 = 2 \times 2 \times (M - 3) \times 8$ bits. Le nombre des éléments séquentiels utilisés est $E_4 = 218$ bits.

10.7 Résultats et conclusions

Les caractéristiques de l'architecture proposée en terme de temps de latence, de taille mémoire RAM et de nombre d'éléments séquentiels utilisés sont détaillés dans le tableau 10.1 pour les quatre fonctions de l'architecture. Dans ce tableau, M représente la taille d'une ligne de l'image traitée. Les diagrammes circulaires des figures 10.13, 10.14 et 10.15, représentent respectivement les pourcentages de latence, de mémoire RAM et d'éléments séquentiels utilisés dans chaque fonction par rapport à l'architecture totale. On peut voir que c'est la fonction « Interpolation direction » qui possède la plus grande latence, ceci est dû au chargement du masque de traitement 5×5. On peut voir que c'est la fonction « Correction direction » qui utilise le plus de mémoire RAM. Ceci est dû au fait que pour garder la synchronisation entre les trois flux de pixels rouge, vert et bleu et le flux de direction, une fenêtre de taille 3×3 est utilisée. Il en est de même pour la fonction « Interpolation Direction ». On peut voir que c'est la fonction Correction direction qui utilise le plus grand nombre d'éléments séquentiels, ceci est dû au chargement des flux de pixels G_h, G_v et $Bayer$ et au composant de calcul « Delta Computation ». La latence L de l'ensemble de l'architecture pour la sortie du premier vecteur couleur est égale à la somme des latences de chaque fonction, soit $L = L_1 + L_2 + L_3 + L_4$, soit $L = 10 \times M + 24$. Un pixel est ensuite généré à chaque coup d'horloge. On rappelle qu'une latence de 5 cycles est nécessaire à la fin de chaque ligne pour recadrer le masque sur la ligne suivante. Considérons une image de taille $M \times N$, le nombre de cycles C pour produire une image est donc $C = (L-1) + M \times N + (N-1) \times 5 = MN + 10M + 5N + 18$. Considérons une image de taille 5 méga-pixels de résolution 2582×1936. Le nombre de cycles pour traiter une image est donc $C = 2582 \times 1936 + 10 \times 2582 + 5 \times 1936 + 18 = 5034270$ cycles. En considérant une fréquence d'horloge de 150 MHz, une image de résolution 2582×1936 est traitée en 0.0336 secondes, soit 6,8 fois plus rapidement que l'exécution sur le processeur TriMedia TM3270. Considérons maintenant une image de résolution VGA (640×480). Le nombre de cycles pour traiter une image est $C = 316018$ cycles. En considérant une fréquence d'horloge de 150 MHz, une image est traitée en $0,0021$ secondes, soit 474.6565 images par seconde, soit 10 fois plus rapidement que l'exécution sur le processeur TriMedia TM3270. Pour finir, les résultats de la synthèse sur la cible FPGA Xilinx Virtex 5 et Altera Stratix IV sont présentés respectivement dans les tableaux 10.2 et 10.3. Quelques informations manquent, en raison d'un problème de compatibilité des outils utilisés. Cependant, le design proposé semble fonctionner plus rapidement sur la cible Xilinx Virtex 5 et utilise moins de ressources que sur la cible Altera Stratix IV.

TABLE 10.1 – Latence, mémoire et éléments séquentiels utilisés par les différentes fonctions de l'architecture

Fonctions	Latence (cycles)	RAM (bits)	Éléments séquentiels (bits)
H&V interpolation	$4 \times M + 8$	$4 \times (M - 5) \times 8$	330
Interpolation Direction	$2 \times M + 7$	$3 \times 2 \times (M - 3) \times 8$	444
Correction Direction	$2 \times M + 4$	$3 \times 2 \times (M - 3) \times (8 + 1)$	266
R&B interpolation	$2 \times M + 5$	$2 \times 2 \times (M - 3) \times 8$	218
Total	$10 \times M + 24$	$166 \times M - 562$	1258

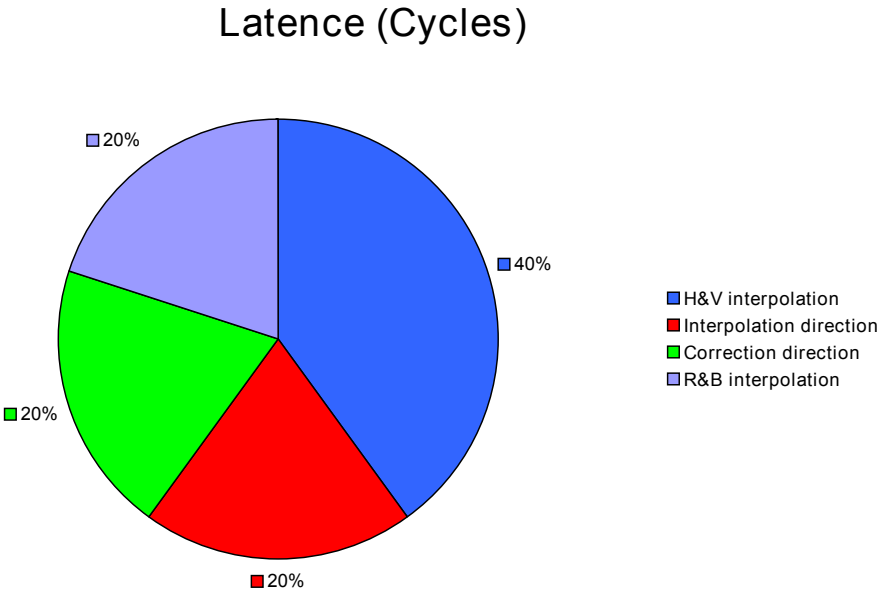


FIGURE 10.13 – Diagramme circulaire montrant le pourcentage des cycles de latence de chaque fonction.

TABLE 10.2 – Résultats de la synthèse FPGA pour la cible Xilinx Virtex 5 xc5vlx30ff324-2

Fréquence estimée	237,2 MHz	
Période estimée	4,217 ns	
Registres	2510 blocs	13% d'utilisation des ressources totales
Dual Port Rams (RAM32X1D)	288 blocs	
LUTs	3580 blocs	18% d'utilisation des ressources totales

TABLE 10.3 – Résultats de la synthèse FPGA pour la cible Altera Stratix IV EP4SGX70B

Fréquence estimée	176,6 MHz
Période estimée	5,664 ns
Registres	3942 blocs
M9Ks (Altesynram)	54 blocs
LUTs	4740 blocs

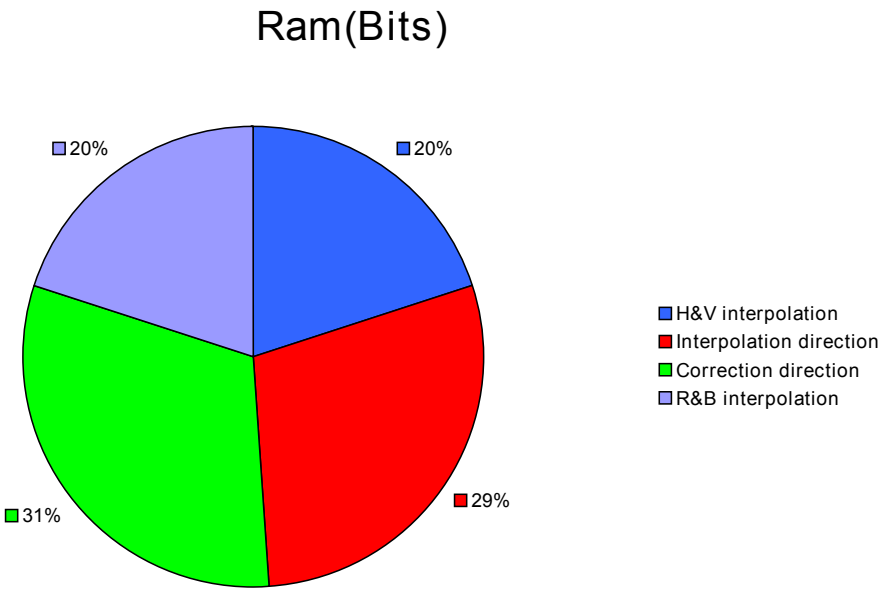


FIGURE 10.14 – Diagramme circulaire montrant le pourcentage de RAM utilisée par chaque fonction.

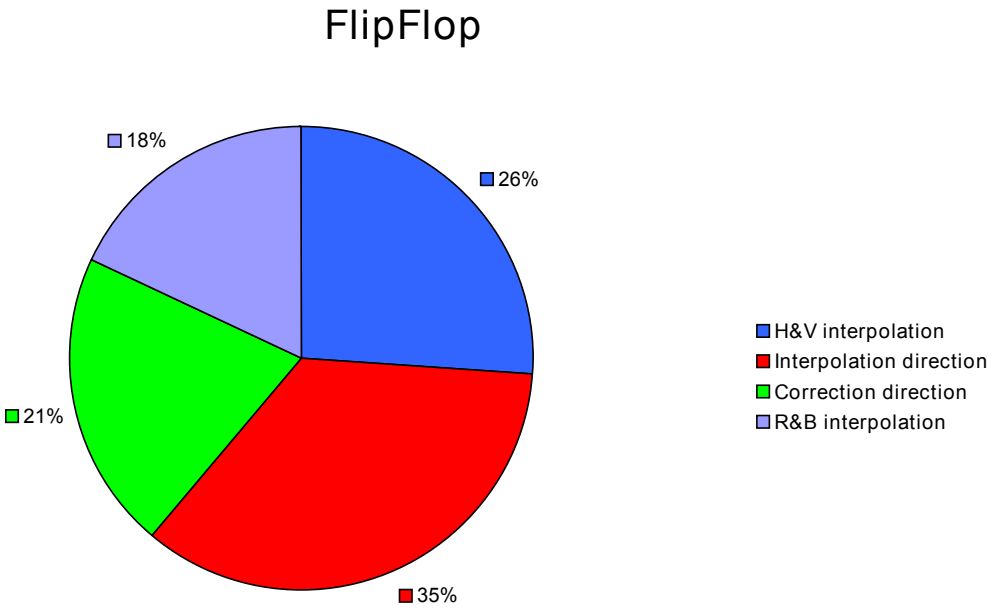


FIGURE 10.15 – Diagramme circulaire montrant le pourcentage de bascules D utilisées par chaque fonction.

Quatrième partie

Conclusion Générale

Chapitre 11

Conclusion et perspectives

11.1 Synthèse

Au cours de cette thèse, nous nous sommes intéressé à la qualité des images produites par les systèmes de captures d'images numériques grand public de type : appareil-photos numériques compacts, téléphones mobiles, assistants personnels numériques, etc. Nous avons étudié les dégradations introduites par la chaîne de capture d'images et plus particulièrement les perturbations générées par les capteurs. Deux caractéristiques fondamentales de ces capteurs sont à l'origine de dégradations dans les images. La non-sensibilité à la couleur qui implique un multiplexage chromatique spatial (filtre de Bayer) et la production de bruit. Pour corriger ces problèmes, nous avons cherché à mettre au point des algorithmes de dématriçage et de réduction du bruit à la fois performants en terme de qualité d'image produite et compatibles avec les architectures de traitements des systèmes employés en photographie numérique.

En raison de l'utilisation généralisée de l'arrangement de couleur proposé par Bayer [11] pour le multiplexage chromatique, le dématriçage de cette mosaïque a été largement étudié et de nombreux algorithmes ont été proposés. Cependant, il n'existe pas de méthodes simples permettant de délivrer une image sans artéfacts. Nous avons entre autres étudié les fonctionnements et les qualités des images produites par les algorithmes proposés par Hamilton et al. dans [8] et Hirakawa et al. dans [18]. Ces deux algorithmes utilisent le principe de dématriçage par interpolations directionnelles. Nous avons constaté que c'est l'utilisation d'un classifieur de direction d'interpolation qui détermine à la fois la qualité des images produites et la complexité de calcul de ces algorithmes. En s'appuyant sur cette observation, nous avons cherché un classifieur peu complexe permettant de produire des images sans artéfacts.

Nous avons développé un nouveau classifieur (GED, Green Edge Direction) basé sur des calculs de gradients dans le canal vert et une homogénéisation du choix de direction local (LMDC, Local Majority Direction Choice). Nous avons généralisé l'utilisation de la méthode LMDC à tous les algorithmes fonctionnant par interpolations directionnelles. Nous avons montré que cette méthode permet d'améliorer de manière importante la qualité des images produites. Le nouveau classifieur GED, utilisé avec le principe de l'interpolation directionnelle permet de formuler un nouvel algorithme de dématriçage de la mosaïque de Bayer (GEDI, Green Edge Directed Interpolation). Cet algorithme possède des propriétés de reconstruction des images excellentes. Il minimise la présence d'artéfacts de dématriçage tels que l'apparition de fausses couleurs, l'introduction de flou, la création d'effets de moiré, la création de structures en formes de labyrinthes et l'effet de grille. Complémentairement à ses bonnes propriétés de reconstruction, GEDI possède une faible complexité de calculs en comparaison des méthodes de la littérature permettant de produire des qualités d'images du même ordre (algorithme de Hirakawa [18] et Gunturk [55]).

Nous avons implémenté, optimisé et simulé l'exécution des algorithmes GEDI, Hamilton [8], Hamilton corrigé par la méthode de LMDC et Hirakawa [18] sur le processeur TriMedia TM3270 de NXP Semiconductors. Le processeur TM3270 est un processeur de référence et de puissance moyenne utilisé dans de nombreuses puces dédiées aux traitements multimédias de la téléphonie mobile (puce PNX4103). L'implémentation de GEDI sur ce processeur permet de traiter une image de résolution 5 méga-pixels en 0,23 secondes et de traiter des images de résolution VGA à une cadence de 50,5 images par seconde. Soit une exécution 9 fois plus rapide que l'algorithme de Hirakawa et seulement 1,43 fois moins rapide que l'algorithme de Hamilton. D'autre part, l'exécution de l'algorithme de Hamilton corrigé par LMDC est seulement 1,23 fois moins rapide que l'algorithme de Hamilton. Le calcul d'homogénéité apporte 0,03 cycles d'exécution par pixel.

Dans le but de standardiser l'utilisation de GEDI sur différentes plateformes de traitements, nous avons proposé une architecture dédiée pour cet algorithme. L'utilisation d'un circuit intégré permet entre autres de minimiser la consommation d'énergie et d'augmenter la rapidité du traitement. Considérons M la largeur de l'image traitée, l'architecture proposée utilise une mémoire de $166 \times M - 562$ bits et 1258 éléments séquentiels, elle possède une latence de $10M + 24$ cycles. Cette architecture permet de traiter une image de résolution 5 méga-pixels en 0.0336 secondes et des images de résolution VGA à la cadence de 474,65 images par seconde, permettant une exécution de traitement multipliée par 10 par rapport à l'exécution sur le processeur TriMedia TM3270.

Dans les systèmes de captures d'images numériques photographiques et vidéos, la quantité de bruit dans l'image varie constamment en fonction de l'intensité d'illumination de la scène. Nous avons montré qu'il est important d'adapter le niveau de filtrage au niveau du bruit. Cette adaptation permet de préserver les détails de la scène lorsque ceux-ci ne sont pas dégradés par la présence de bruit et de filtrer le bruit en contrepartie de la perte des détails lorsque le niveau de bruit dépasse l'information de structure. Nous avons montré comment le filtre bilatéral peut-être optimisé si le niveau de bruit est connu. Nous avons développé un filtre de restauration d'images dans le cas d'un bruit de photon. Ce filtre permet d'adapter le paramètre de similarité photométrique du filtre bilatéral en fonction de la quantité de bruit estimée dans l'image. Nous avons présenté une méthode d'estimation du niveau de bruit qui est simple et efficace, en exploitant les propriétés statistiques poissonniennes du signal et le processus d'adaptation du gain utilisé dans les systèmes de captures d'images numériques. Les tests que nous avons réalisés montrent que cet algorithme permet d'obtenir une corrélation excellente entre le niveau de bruit estimé et le niveau de bruit réel. En appliquant notre méthode de détection de puissance de bruit pour adapter le paramètre de similarité photométrique du filtre bilatéral, nous avons obtenu un nouveau filtre bilatéral adaptatif permettant d'obtenir le meilleur filtrage bilatéral pour différents niveaux de bruits de photon.

Dans le cadre de la réduction du bruit dans les capteurs d'images numériques couleurs, nous avons développé une formulation du filtre bilatéral adaptée pour la matrice de Bayer, le filtre bilatéral Bayer. Nous avons montré qu'il est important de filtrer l'image avant l'étape de dématriçage pour deux raisons fondamentales. La première raison est que la discrimination du bruit par rapport au signal utile est plus efficace lorsqu'elle est effectuée avant le mélange des canaux de couleurs introduit par l'étape de dématriçage. La deuxième raison est que la présence de bruit dans l'image de mosaïque perturbe fortement le fonctionnement de l'étape de dématriçage. Les tests effectués montrent que le filtrage appliqué avant l'étape de dématriçage permet de produire une meilleure qualité d'image. Notamment cela permet d'éliminer la création des tâches de couleurs induites par la dispersion du bruit à travers les canaux de couleurs. D'autre part, les images produites sont généralement moins bruitées, moins floues et possèdent des contours mieux préservés.

Finalement, nous avons utilisé l'algorithme d'estimation du bruit proposé pour adapter le paramètre de similarité photométrique du filtre bilatéral Bayer. On formule ainsi un nouveau filtre, le filtre bilatéral adaptatif Bayer. Ce filtre possède à la fois les propriétés d'adaptation du paramètre de similarité photométrique en fonction de la puissance du bruit dans l'image et les qualités du filtre bilatéral Bayer citées si-dessus.

Nous avons implémenté, optimisé et simulé l'exécution du filtre bilatéral Bayer sur le processeur TriMedia TM3270. Nous avons adapté sa formulation en fonction des ressources du processeur pour permettre une exécution optimisée du filtre. Finalement, la formulation proposée permet de traiter une image de résolution 5 méga-pixels en $0,35$ secondes et de traiter des images de résolution VGA à une cadence de $46,9$ images par seconde.

11.2 Perspectives

Dans cette thèse, nous nous sommes principalement intéressé aux algorithmes de reconstruction de la mosaïque de Bayer en raison de sa présence généralisée dans les appareils de captures d'images numériques. Cependant, il existe de nombreux filtres de couleurs dans la littérature, notamment la société Kodak à récemment proposé un filtre « panchromatique » [10], dans lequel un pixel « panchromatique », c'est dire sensible à l'ensemble des couleurs du spectre visible est ajouté dans la mosaïque (voir figure 11.1). L'ajout de ce pixel permet une meilleure sensibilité à la lumière. Les images obtenues sont ainsi moins bruitées. Le temps d'obturation peut-être diminué, ce qui permet d'obtenir des images plus nettes. Enfin, la résolution des capteurs peut être augmentée en diminuant la surface photosensible des photosites. Les images de la figure 11.2 montre la différence de présence de bruits obtenue entre l'utilisation du nouveau filtre « panchromatic » proposé par Kodak et le filtre de Bayer superposés sur le même capteur. Dans nos futures travaux, nous envisageons d'adapter l'algorithme GEDI pour le dématriçage du filtre de couleurs « panchromatique ».

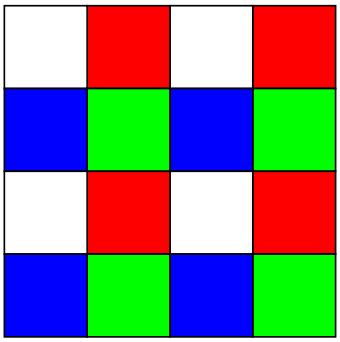


FIGURE 11.1 – Filtre « panchromatique » proposé par la société Kodak [10].

D'autre part, nous avons vu que la puissance du bruit dans un capteur d'image numérique varie constamment en fonction de la luminosité de la scène. Pour permettre un filtrage performant du bruit, nous avons montré qu'il faut adapter la puissance du filtrage à la puissance du bruit. Il serait intéressant d'utiliser les méthodes de stabilisation de la variance proposées par Anscombe [109] pour minimiser les variations de puissance du

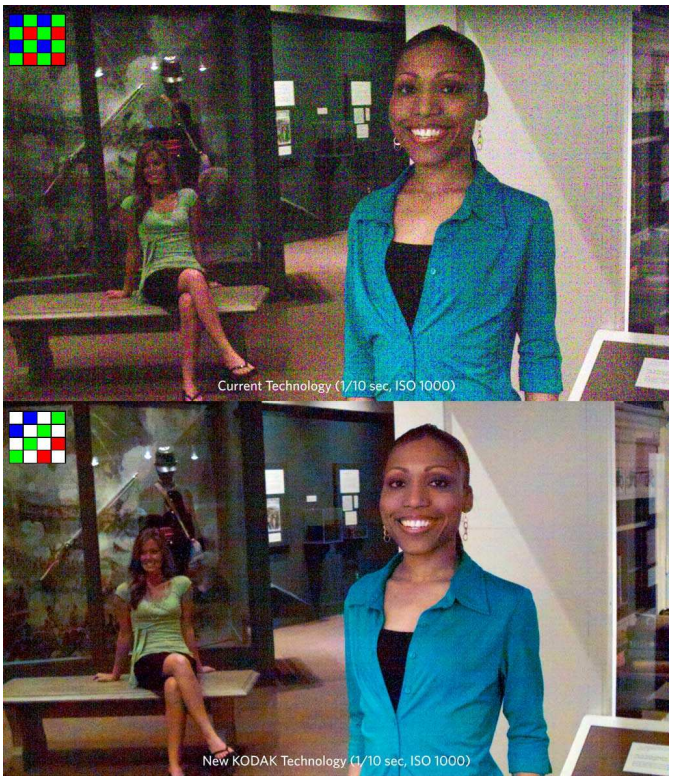


FIGURE 11.2 – Comparaison de sensibilté au bruit entre le filtre « panchromatic » et le filtre de Bayer.

bruit, on pourrait immédiatement réutiliser ces méthodes dans notre travail d'estimation du nombre de photons dans ce contexte. Cela permettrait d'éviter l'adaptation du paramètre de similarité photométrique du filtre utilisé en fonction de la puissance du bruit. Enfin, nous envisageons d'étudier l'intégration de l'architecture du circuit intégré proposé pour l'algorithme de GEDI dans une chaîne de traitement réelle.

Cinquième partie

Annexes

Annexe A

Propriétés des systèmes optiques

La figure A.1 montre le schéma d'une chaîne d'éléments de capture optique formée d'un système de lentilles. La formation de l'image $A'B'$ d'un objet AB à travers un système de lentilles est présenté sur la figure A.2. En optique géométrique, un système optique est défini par son centre optique O, son foyer image F', son foyer objet F et son axe optique passant par les points F, F' et O. On rappelle que tous les points situés à l'infini, se propagent parallèlement à l'axe optique et focalisent sur le foyer image F'. On rappelle aussi, que tous les rayons issus du foyer objet F se propagent parallèlement à l'axe optique après avoir traversé la lentille.

La caractéristique fondamentale d'un système optique est sa distance focale OF'. Celle-ci détermine le grossissement et le champ de vision observé au travers de l'objectif. Plus la focale est courte, plus le champ de vision est large. Inversement plus la focale est longue, plus le champs de vision est restreint. La focale variable d'un objectif sert à agrandir ou à diminuer la taille des objets sur le capteur. Une illustration de l'utilisation de la variation de la focale pour le réglage du zoom est montrée sur la figure A.3. Le réglage du focus qui contrôle la netteté de l'image, peut être fait en modifiant la distance AO entre l'objet et le centre optique, comme cela est illustré sur la figure A.4.

La focale normale, celle qui permet de se rapprocher le plus de la vision humaine, est déterminée par la diagonale du négatif utilisé : 43mm pour un film de taille 24×36 mm par exemple. Dans la pratique, on parle généralement d'un 50mm. Les appareils ayant une distance focale inférieure sont appelés grands angles, les autres sont des téléobjectifs. Ces distances diffèrent totalement entre un appareil-photo argentique et un appareil-photo numérique puisque la taille du capteur de l'appareil numérique est relativement restreinte. Les lentilles sont alors très rapprochées du capteur. C'est pour cela qu'on trouve habituellement la correspondance avec un appareil argentique classique 35 mm.

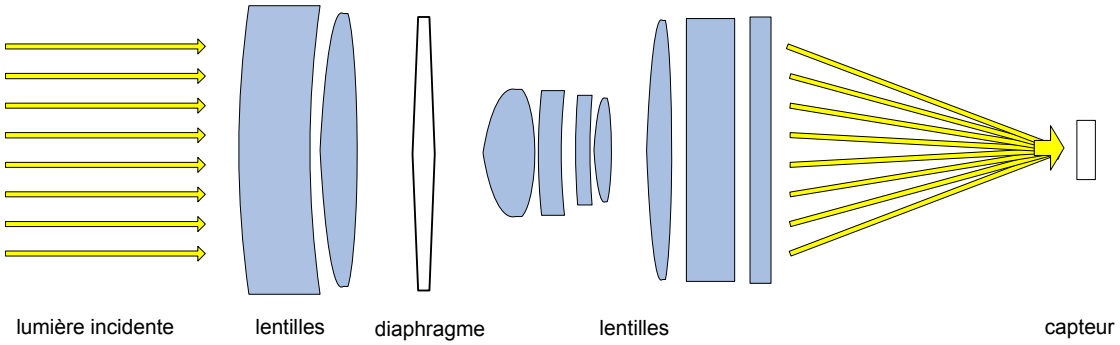


FIGURE A.1 – Système de lentilles permettant de projeter l'image de la scène sur le capteur numérique.

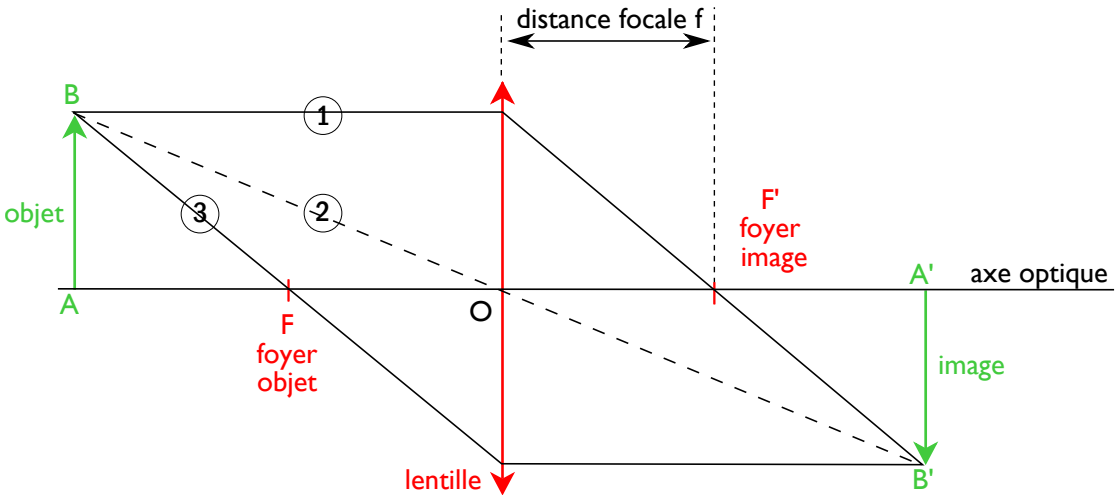


FIGURE A.2 – Formation d'une image à travers une lentille convergente : l'objet AB est projeté en une image $A'B'$ renversée, le rayon (1) parallèle à l'axe optique sort de la lentille en passant par le foyer F', le rayon (2) passant par le centre optique n'est pas dévié, le point B' se trouve à l'intersection des rayon (1) et (2), le rayon (3) passe par le foyer objet F' et ressort parallèle à l'axe optique.

Le diaphragme est un mécanisme présent sur l'objectif, dont le fonctionnement est semblable à celui de l'iris de l'œil. Composé de fines lamelles qui se chevauchent, il permet d'ajuster la quantité de lumière traversant l'objectif. Sa valeur est appelée ouverture. Symbolisée par la lettre f, elle correspond au rapport du diamètre utile de l'objectif à sa distance focale (f/D). Les valeurs les plus courantes sont : $1-1,4-2-2,8-4-5,6-8-11-16$, etc. La quantité de lumière est divisée par deux à chaque graduation (voir figure A.5), donc plus la valeur d'ouverture est grande, plus le diaphragme est fermé.

L'obturateur peut être situé au centre de l'objectif ou juste devant le film. Il est composé de lamelles métalliques qui se recouvrent mutuellement, dont le but est de laisser passer la lumière ou non. Lors de l'appui sur le déclencheur, l'obturateur s'ouvre puis se referme. La durée pendant laquelle il reste ouvert, appelée temps de pose ou vitesse d'obturation, peut prendre les valeurs suivantes, exprimées en seconde : $1-1/2-1/4-1/8-1/15-$

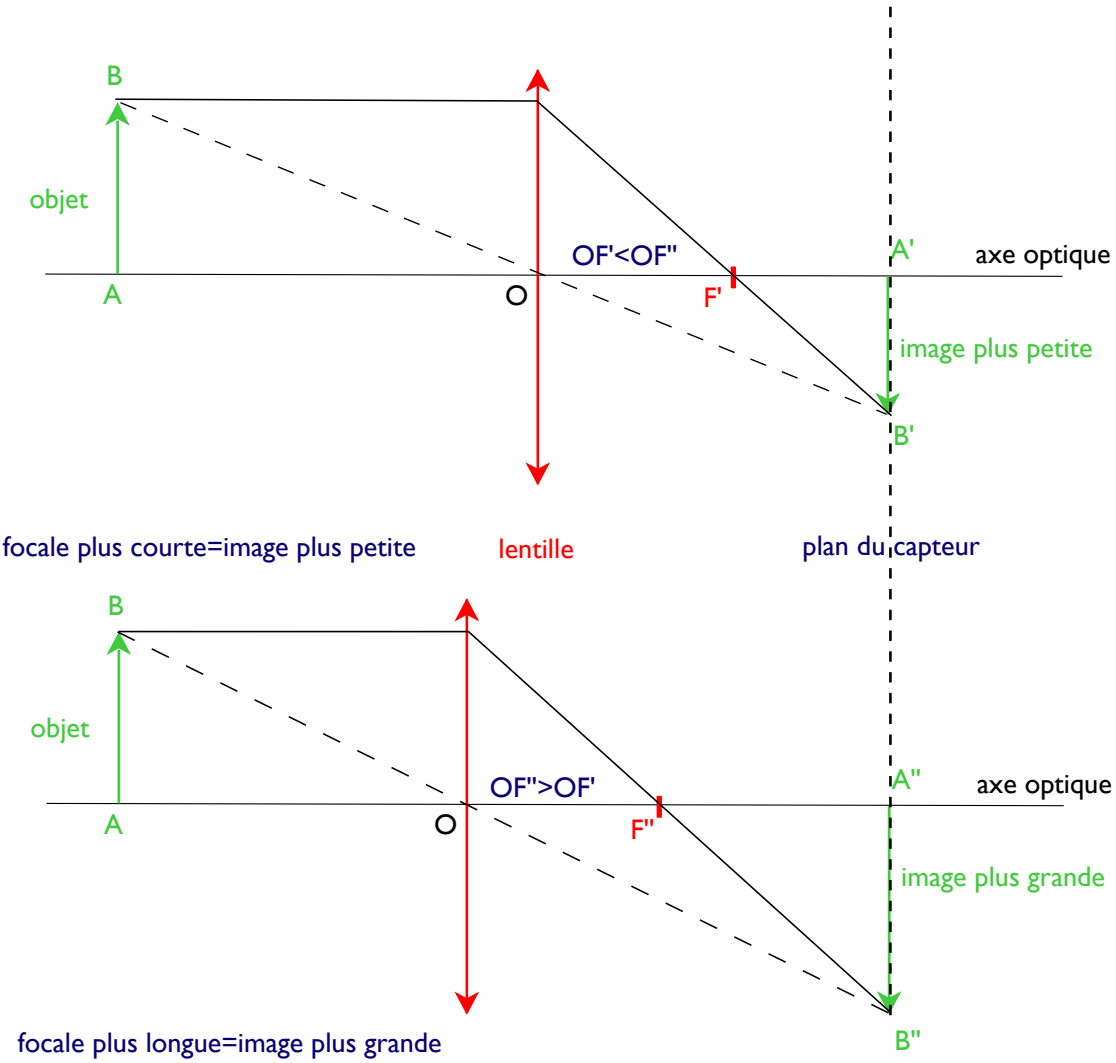


FIGURE A.3 – Ajustement de la taille de l'image par variation de la focale : en changeant la distance focale et en déplaçant la lentille pour refaire la mise au point, la taille de l'image varie et permet d'utiliser au mieux la surface du capteur.

$1/30-1/60-1/125-1/250-1/500$, etc. Comme pour l'ouverture, la quantité de lumière reçue à chaque graduation est divisée par deux.

Le couple diaphragme/obturateur permet donc de régler l'exposition, c'est à dire la quantité de lumière reçue par le film ou le capteur. On parle alors de couple ouverture/vitesse. Les appareil-photos numériques proposent en général quatre modes de réglage de l'exposition :

- Automatique, où le couple diaphragme/obturateur est réglé par l'appareil ;
- Semi-automatique, où l'utilisateur choisi un mode prédéfini, par exemple : sport, intérieur, nuit, portrait, paysage etc.
- Priorité à l'ouverture, où l'utilisateur peut régler l'ouverture du diaphragme, l'appareil se charge d'adapter la vitesse d'obturation ;

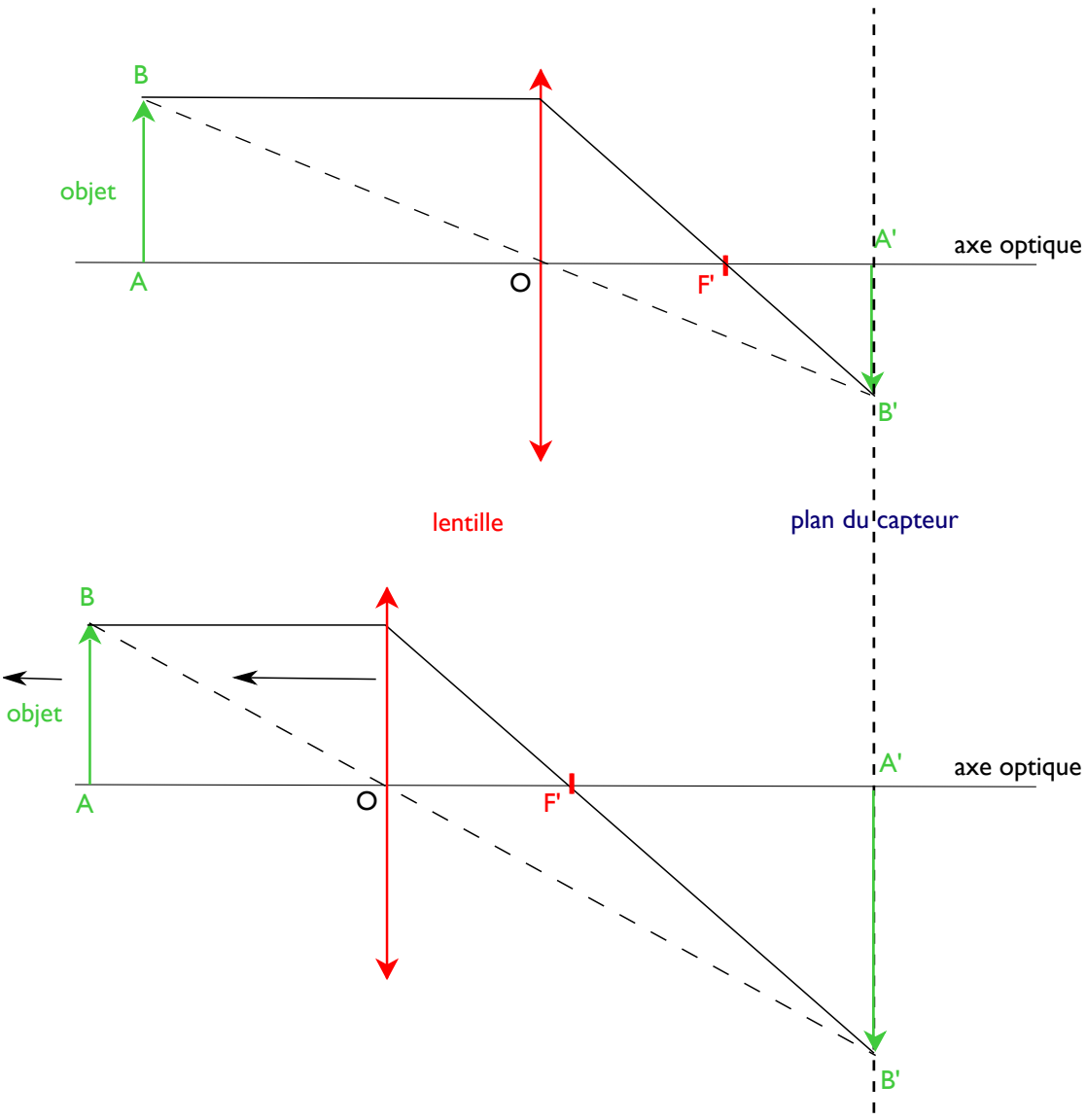


FIGURE A.4 – Principe de mise au point ou focus : pour que l'image $A'B'$ soit sur le capteur, il faut faire varier la position de la lentille par rapport au capteur, soit OA'.

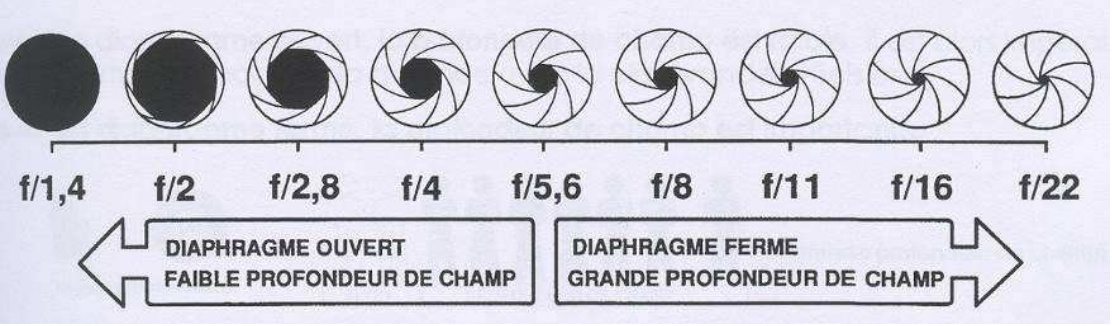


FIGURE A.5 – Différentes ouvertures de diaphragme : plus l'ouverture du diaphragme est grande, plus la profondeur de champs est faible, inversement, plus l'ouverture du diaphragme est faible, plus la profondeur de champs est grande.

– Priorité à la vitesse, l'utilisateur choisit la vitesse d'obturation et l'appareil règle l'ouverture du diaphragme ;

La profondeur de champ est la distance entre le point le plus rapproché et le point le plus éloigné dont l'appareil fournit une image nette. Elle n'est pas également répartie de part et d'autre de la distance de mise au point (figure A.6 et figure A.7) : elle est deux fois plus étendue à l'arrière qu'à l'avant du sujet. La profondeur de champ dépend de :

– l'ouverture du diaphragme : plus le diaphragme est fermé et plus la profondeur de champ est grande
– la focale de l'objectif : plus la focale est courte et plus la profondeur de champ est grande
– la distance appareil/sujet : plus le sujet est éloigné plus la profondeur de champ est grande

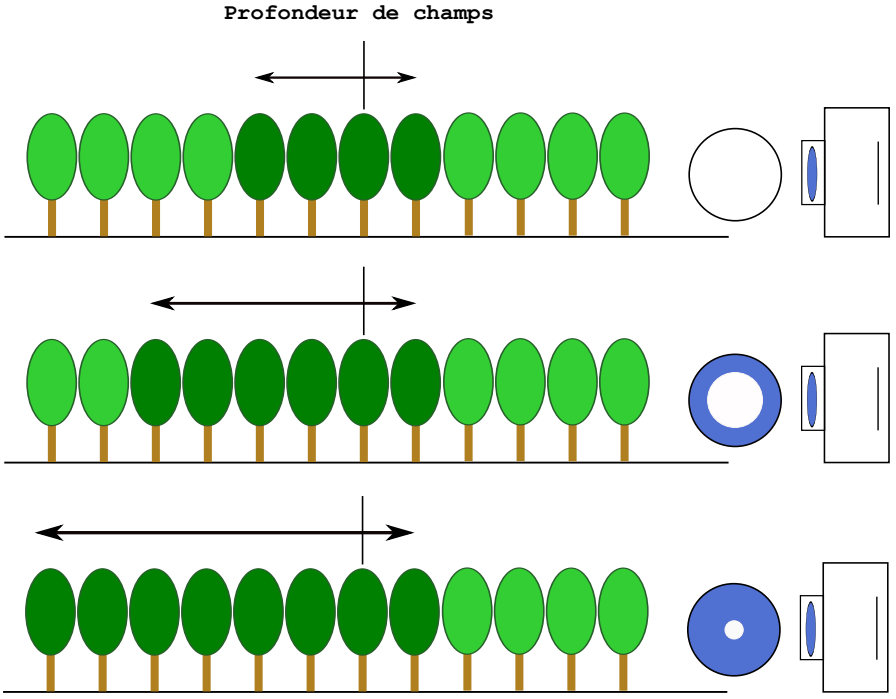
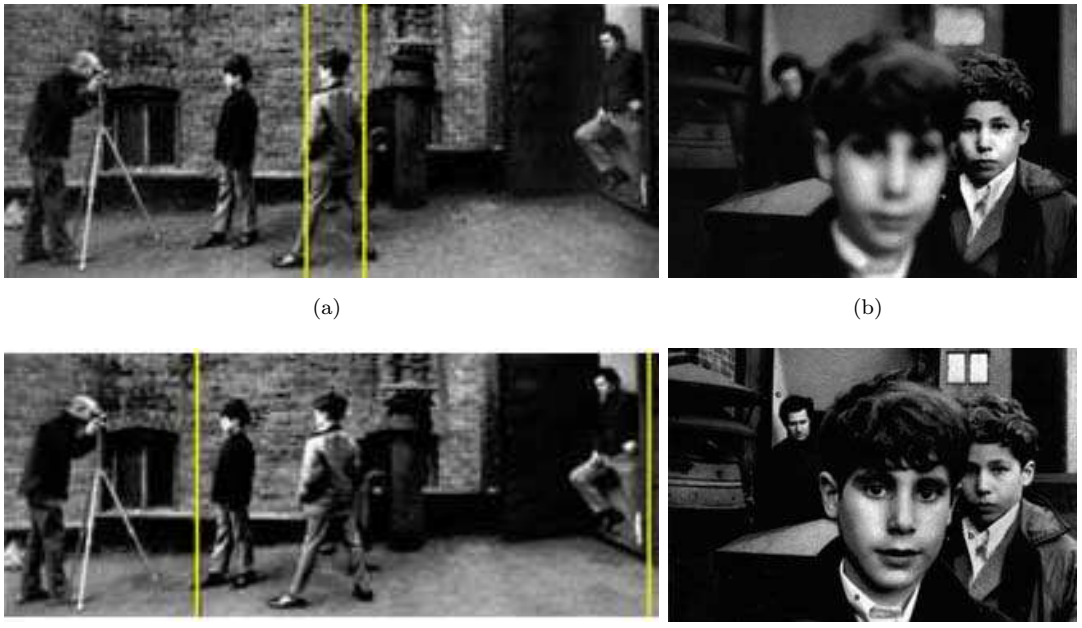


FIGURE A.6 – Influence de l'ouverture du diaphragme sur la profondeur de champs : plus l'ouverture du diaphragme est faible, plus la profondeur de champs est grande.



(a) (b)

(c) (d)

FIGURE A.7 – Influence de l'ouverture du diaphragme sur la profondeur de champs : dans (a) et (b), on utilise une ouverture $f = 1/2$ et la profondeur de champs est faible, dans (c) et (d) on utilise une ouverture $f = 1/16$ et la profondeur de champs est importante.

Annexe B

Le processeur multimédia TriMedia TM3270

Le processeur multimédia TriMedia TM3270 est un processeur VLIW (Very Long Instruction Word) de la famille des processeurs TriMedia de la société NXP Semiconductors, anciennement Philips Semiconductors. L'architecture des processeurs TriMedia trouve ses origines dans le projet de recherche LIFE développé par Philips semiconductors dans la ville de Palo Alto en Californie [110–112].

B.1 Architecture du processeur multimédia TriMedia TM3270

Les processeurs TriMedia possèdent une architecture de type Harvard [113] disposant de nombreuses opérations DSP (Digital Signal Processor) et SIMD (Single Instruction, Multiple Data) pour traiter de manière efficace les flux audios et vidéos. L'implémentation des algorithmes sur le processeur TriMedia se fait directement en langage C/C++. Généralement, le jeu d'instructions et l'architecture d'une famille de processeurs évolue avec le temps, permettant d'introduire les nouvelles fonctionnalités nécessaires aux dernières applications du domaine traité. Ces changements d'architectures sont fait de manières incrémentales pour assurer une compatibilité ascendante entre les différentes versions d'un processeur. Les programmes créés pour fonctionner sur une architecture doivent aussi pouvoir fonctionner sur la dernière architecture de la même famille. Dans le cas des processeurs TriMedia, la compatibilité se fait au niveau du programme source plutôt qu'au niveau binaire. La transformation du programme source en code binaire est assurée par le compilateur TriMedia. Ceci permet une liberté supplémentaire dans

l'implémentation de l'architecture d'un processeur. L'évolution de l'architecture du processeur TriMedia est illustrée sur la figure B.1, sur cette figure, on peut voir quatre niveaux d'architectures ainsi que les processeurs qui y sont associés. De manière générale, les processeurs de la famille TriMedia sont dédiés aux domaines des applications multimédias. Originellement, ce domaine incluait le sous domaine des applications graphiques 3D [114]. Cependant, l'augmentation des demandes dans ce domaine à conduit à la mise en œuvre de processeurs graphiques spécifiques [115]. Par conséquent, les processeurs TriMedia sont désormais dédiés dans les domaines des traitements audios, vidéos (capacité à supporter la norme H.264 [116, 117]) et la photographie.

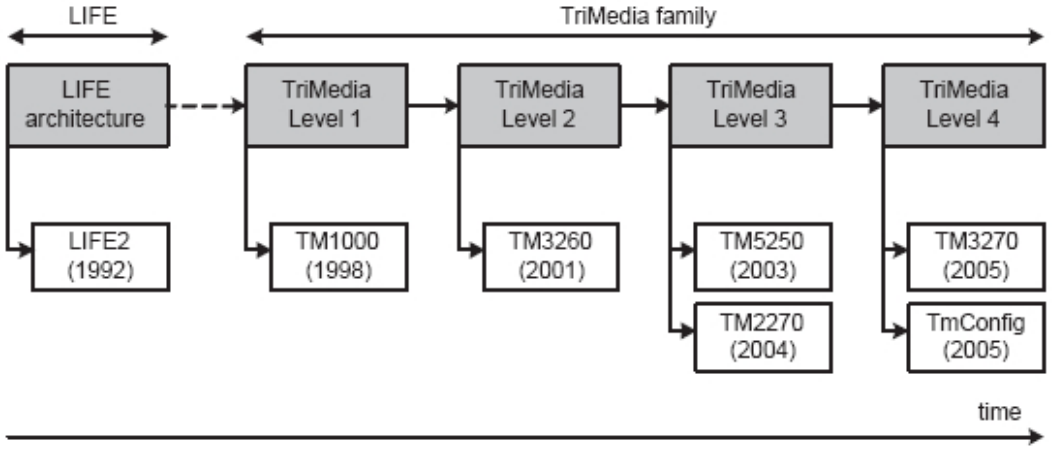


FIGURE B.1 – Évolution de l'architecture de la famille des processeurs TriMedia.

Le tableau B.1 présente les principales caractéristiques du processeur TriMedia TM3270, un processeur standard utilisé en téléphonie mobile. Il opère à une fréquence maximale de 350MHz et peut exécuter jusqu'à 5 instructions simultanées par cycle d'horloge. Il est développé pour correspondre aux performances imposées par les traitements vidéos standards. Le processeur TM3270 supporte les opérations SIMD multimédias et les opérations IEEE-754 en virgules flottantes. Il utilise 128 registres de 32-bits, une mémoire cache de 128 kilos octets, un cache d'instruction de 64 kilos octets et un ensemble d'instructions multimédias dédiées pour le traitement d'image et le traitement vidéo.

La figure B.2 montre l'architecture de la puce multimédia Philips Nexperia PNX4103 [118]. C'est un exemple de circuit intégré dédié aux traitements des images et des vidéos pour les téléphones mobiles utilisant le processeur multimédia TM3270. Il comporte un capteur avec une résolution de 10 millions de pixels pour la photographie et un capteur de résolution VGA (640×480) pour la téléphonie vidéo. Il possède aussi deux écrans LCD et une connexion TV-out intégrée. Les algorithmes que nous avons développés sont entre autres destinés à être implémenté sur ce type de puce.

TABLE B.1 – Présentation de l'architecture du processeur TM3270

Dispositif de l'architecture	Quantité
Architecture	VLIW à 5 slots, Opérations RISC sans pénalités de branche
Profondeur des pipelines	7-13 niveaux
Taille des adresses	32-bits
Taille des données	32-bits
Banc de registres	Unifié, 128 registres 32-bits
Virgule flottante	IEEE-754
SIMD	1×32-bits ; 2×16-bits ; 4×8-bit
Cache d'instructions	64 kilo octets, associatif par ensemble avec 8 lignes de 128 octets
Cache de données	128 kilo octets, associatif par ensemble avec 4 lignes de 128 octets

B.2 Environnement de programmation TriMedia

Le développement logiciels pour le processeur TriMedia est supporté par un système de compilation et de simulation, le TCS (TriMedia Compilation System). Ce système permet d'augmenter la rapidité de création d'applications multimédias en utilisant les langages de programmation C et C++ et un compilateur conforme à la norme ANSI C/C++ [119, 120]. L'ensemble des outils du TCS fournissent une suite de logiciels pour compiler et débugger les applications multimédias, analyser et optimiser les performances et simuler et optimiser les exécutions sur le processeur TriMedia. Il contient un jeu d'instructions dédiées au processeur TriMedia pour les applications audios, vidéos et la photographie.

La figure B.3 montre le flux de développement logiciel TriMedia. Le compilateur et le préprocesseur intégré (C/C++ Front End) procèdent à la substitution des macros, la compilation conditionnelle et l'inclusion des fichiers « header ». Le compilateur du noyau (Core Compiler) traduit le code en arbres de décisions, un langage de représentation intermédiaire. Un arbre de décision est la combinaison de blocs basiques ayant un point d'entré et un ou plusieurs points de sorties. Le « scheduler » traduit le code de l'arbre de décision en code assembleur. Cette première étape de compilation permet de prendre en compte les contraintes d'adéquations algorithmiques et matériels. Un rapport comportant le rendement d'utilisation des cinq slots du processeur pour un arbre de décision donné est généré. Lors d'une deuxième compilation, le compilateur peut aussi utiliser ces données pour augmenter le rendement d'utilisation des slots. Le programmeur peut

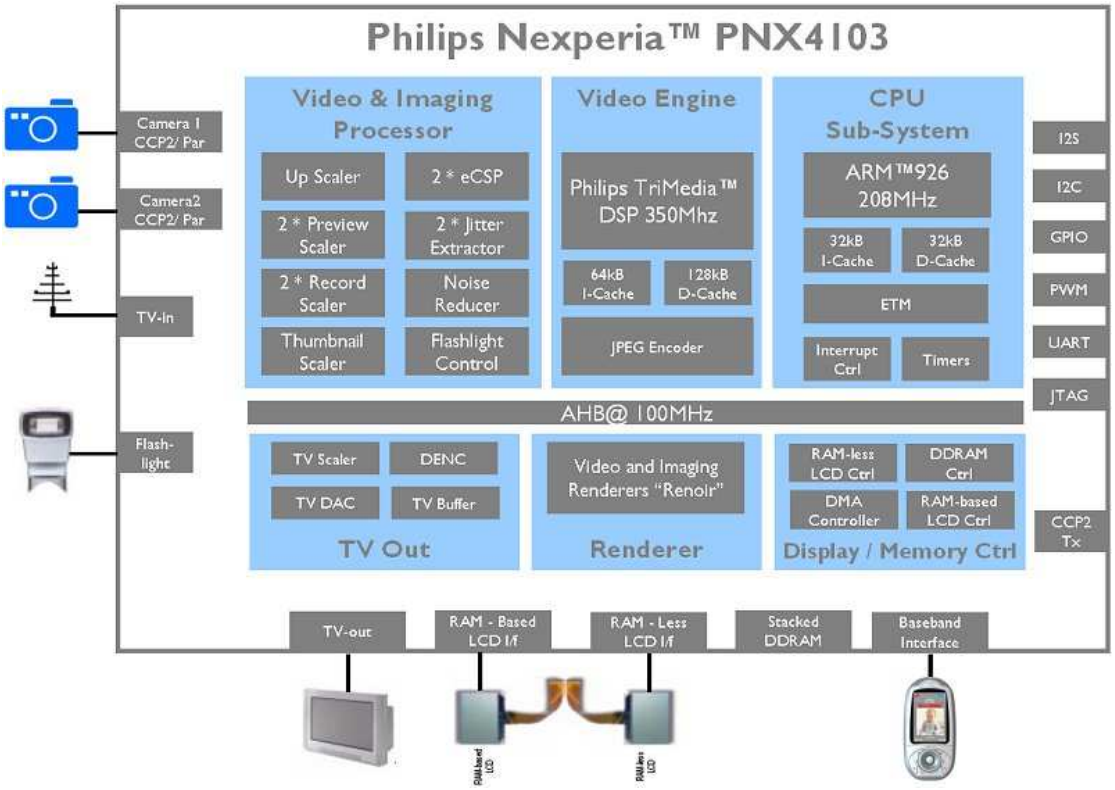


FIGURE B.2 – Processeur multimédia TM3270 intégré dans la puce Philips Nexperia PNX4103.

utiliser le simulateur de base pour vérifier le comportement du code et les résultats générés. Un comportement plus détaillé de l'algorithme peut être obtenu en utilisant des options de compilation plus élaborées mais plus longues à exécuter. Ces options permettent d'avoir accès au nombre de cycles nécessaires à l'exécution de chaque partie du code.

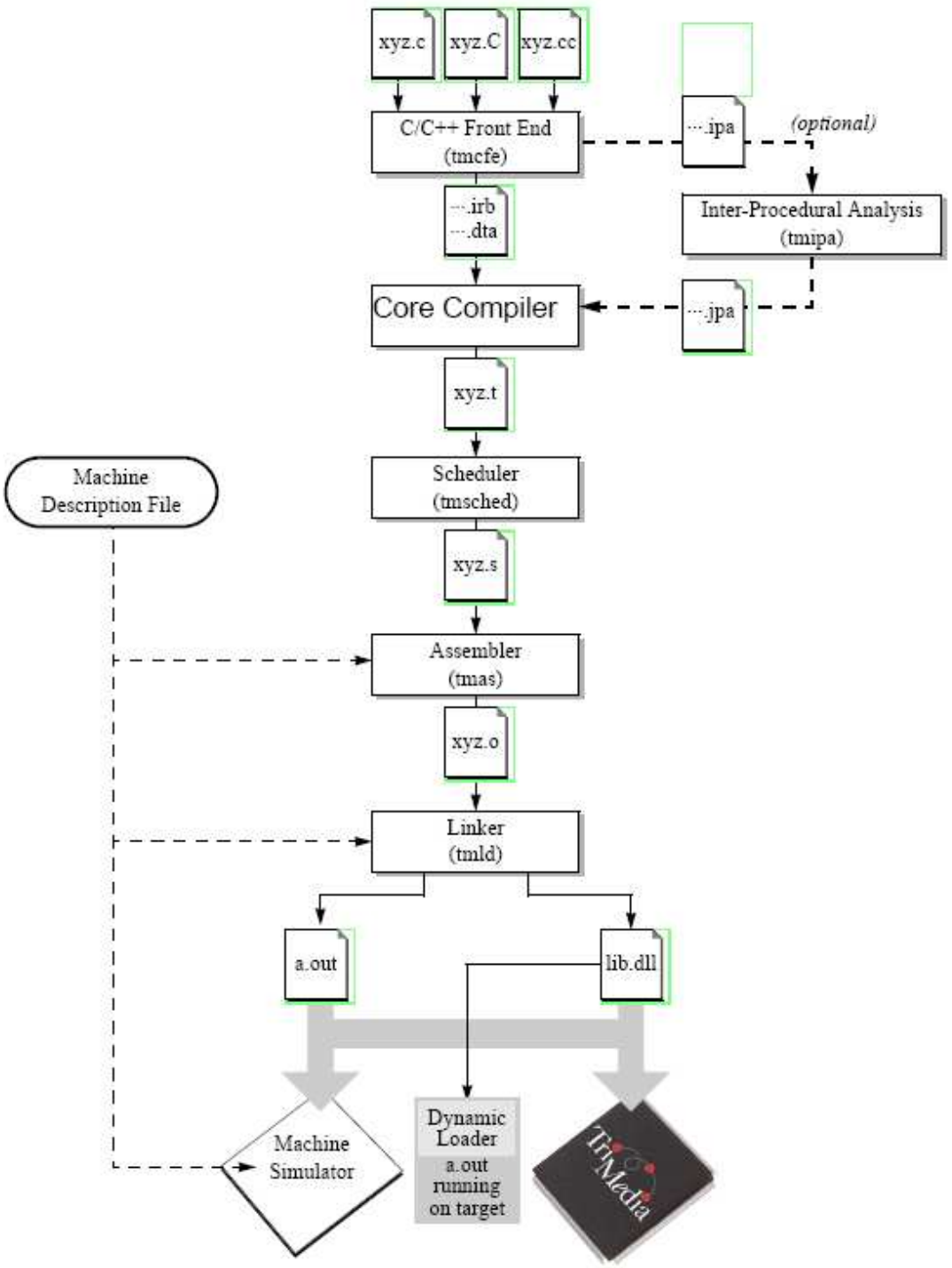


FIGURE B.3 – Flux de développement software TriMedia.

Annexe C

Pseudos-codes des algorithmes étudiés dans la partie III

Algorithm 3 Réduction des artéfacts d'interpolation

Filtrage médian sur les plans rouge et bleu
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    Charger un masque de taille 3×3 *pour les plans rouge, vert, et bleu*
    $M_R = median\ (R_k - G_k)$ *où* R_k *and* $G_k \in$ *masque* 3×3
    $M_B = median\ (B_k - G_k)$ *où* B_k *and* $G_k \in$ *masque* 3×3
    $R' = M_R + G$
    $B' = M_B + G$
  fin pour
fin pour
Filtrage médian sur le plan vert
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    Charger un masque de taille 3×3 *pour les plan rouge, vert, et bleu*
    $M_R = median\ (G_k - R'_k)$ *où* R'_k *and* $G_k \in$ *masque* 3×3
    $M_B = median\ (G_k - B'_k)$ *où* B'_k *and* $G_k \in$ *masque* 3×3
    $G' = (M_R + R + M_B + B)/2$
  fin pour
fin pour

Algorithm 4 Reconstruction des plans rouge et bleu par la méthode de constance des teintes

pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    si $Bayer_{j,i}$ *est rouge* **alors**
      $Red = Bayer(j,i)$
      $Blue = (Bayer(j-1,i-1) + Bayer(j-1,i+1) + Bayer(j+1,i-1) + Bayer(j+1,i+1))/4 + (4 \cdot G(j,i) - G(j-1,i-1) - G(j-1,i+1) - G(j+1,i-1) - G(j+1,i+1))/4$
    sinon si $Bayer_{j,i}$ *est bleu* **alors**
      $Blue = Bayer(j,i)$
      $Red = (Bayer(j-1,i-1) + Bayer(j-1,i+1) + Bayer(j+1,i-1) + Bayer(j+1,i+1))/4 + (4 \cdot G(j,i) - G(j-1,i-1) - G(j-1,i+1) - G(j+1,i-1) - G(j+1,i+1))/4$
    sinon si $Bayer_{j,i}$ *est vert dans une ligne rouge* **alors**
      $Red = (Bayer(j,i-1) + Bayer(j,i+1))/2 + (2 \cdot G(j,i) - G(j,i-1) - G(j,i+1))/2$
      $Blue = (Bayer(j-1,i) + Bayer(j+1,i))/2 + (2 \cdot G(j,i) - G(j-1,i) - G(j+1,i))/2$
    sinon si $Bayer(j,i)$ *est vert dans une ligne bleu* **alors**
      $Blue = (Bayer(j,i-1) + Bayer(j,i+1))/2 + (2 \cdot G(j,i) - G(j,i-1) - G(j,i+1))/2$
      $Red = (Bayer(j-1,i) + Bayer(j+1,i))/2 + (2 \cdot G(j,i) - G(j-1,i) - G(j+1,i))/2$
    fin si
    $R(j,i) =$ **clip** $(Red, 0, 255)$
    $B(j,i) =$ **clip** $(Blue, 0, 255)$
  fin pour
fin pour

Algorithm 5 Hamilton : Reconstruction du plan vert

pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    si $\text{Bayer}_{i,j}$ *est rouge ou bleu* **alors**
      $\Delta h = |2 \cdot Bayer(j,i) - Bayer(j,i-2) - Bayer(j,i+2)| + |Bayer(j,i-1) - Bayer(j,i+1)|$
      $\Delta v = |2 \cdot Bayer(j,i) - Bayer(j-2,i) - Bayer(j+2,i)| + |Bayer(j-1,i) - Bayer(j+1,i)|$
      si $\Delta h \leq \Delta v$ **alors**
        $Green = (2 \cdot Bayer(j,i) - Bayer(j,i-2) - Bayer(j,i+2))/4 + (Bayer(j,i-1) + Bayer(j,i+1))/2$
      sinon
        $Green = (2 \cdot Bayer(j,i) - Bayer(j-2,i) - Bayer(j+2,i))/4 + (Bayer(j-1,i) + Bayer(j+1,i))/2$
      fin si
    sinon
      $Green = Bayer_{j,i}$
      $G_{j,i} =$**clip** $(Green, 0, 255)$
  fin pour
fin pour

Algorithm 6 GEDI : Reconstruction du plan vert

pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    $\Delta h(G_h(j,i)) = |G(j-1,i) - G(j,i)| + |G(j+1,i) - G(j,i)|$
    $\Delta v(G_h(j,i)) = |G(j,i-1) - G(j,i)| + |G(j,i+1) - G(j,i)|$
    si $(\Delta h(G_h(j,i)) + \Delta h(G_v(j,i))) \leq (\Delta v(G_h(j,i)) + \Delta v(G_v(j,i)))$ **alors**
      $MAP_h(j,i) = 1$
    sinon
      $MAP_h(j,i) = 0$
    fin si
  fin pour
fin pour
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    $E = \sum_{kj \in [-1,1],\ ki \in [-1,1]} MAP_{hj+kj,i+ki}$
    si $E < (n^2 - 1)/2$ **alors**
      $G(j,i) = (2 \cdot Bayer(j,i) - Bayer(j-2,i) - Bayer(j+2,i))/4 + (Bayer(j-1,i) + Bayer(j+1,i))/2$
    sinon
      $G(j,i) = (2 \cdot Bayer(j,i) - Bayer(j,i-2) - Bayer(j,i+2))/4 + (Bayer(j,i-1) + Bayer(j,i+1))/2$
    fin si
  fin pour
fin pour

Algorithm 7 Hamilton corrigé par LMDC : Reconstruction du plan vert

pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    $\Delta h = |2 \cdot Bayer(j,i) - Bayer(j,i-2) - Bayer(j,i+2)| + |Bayer(j,i-1) - Bayer(j,i+1)|$
    $\Delta v = |2 \cdot Bayer(j,i) - Bayer(j-2,i) - Bayer(j+2,i)| + |Bayer(j-1,i) - Bayer(j+1,i)|$
    si $\Delta h \leq \Delta v$ **alors**
      $MAP_h(j,i) = 1$
    sinon
      $MAP_h(j,i) = 0$
    fin si
  fin pour
fin pour
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    $E = \sum_{kj \in [-1,1],\ ki \in [-1,1]} MAP_{hj+kj,i+ki}$
    si $E < (n^2 - 1)/2$ **alors**
      $G(j,i) = (2 \cdot Bayer(j,i) - Bayer(j-2,i) - Bayer(j+2,i))/4 + (Bayer(j-1,i) + Bayer(j+1,i))/2$
    sinon
      $G(j,i) = (2 \cdot Bayer(j,i) - Bayer(j,i-2) - Bayer(j,i+2))/4 + (Bayer(j,i-1) + Bayer(j,i+1))/2$
    fin si
  fin pour
fin pour

Algorithm 8 Hirakawa : conversion CIELab

pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    Conversion de l'espace RGB vers l'espace XYZ
    C est une matrice de coefficients
    $X = C_{R,X}R(j,i) + C_{G,X}G(j,i) + C_{B,X}B(j,i)$
    $Y = C_{R,Y}R(j,i) + C_{G,Y}G(j,i) + C_{B,Y}B(j,i)$
    $Z = C_{R,Z}R(j,i) + C_{G,Z}G(j,i) + C_{B,Z}B(j,i)$
    Conversion de l'espace XYZ vers l'espace Lab
    pour tout t=X,t=Y,t=Z **faire**
      si $t \leq 0.008856$ **alors**
        $f(t) = (7.787 \cdot t + 16/116)$
      sinon
        $f(t) = \sqrt[3]{t}$
      fin pour
    fin si
    $L_{j,i} = 116 \cdot f(X) - 116$
    $a_{j,i} = 500 \cdot (f(X) - f(Y))$
    $b_{j,i} = 200 \cdot (f(Y) - f(Z))$
  fin pour
fin pour

Algorithm 9 Hirakawa : Calcul des paramètres d'homogénéïté

Données $L_h, a_h, b_h, L_v, a_v, b_v, eC, eL, RGB_h, RGB_v, m, n$
Résultats e_L, e_C
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    $d_{L_{left}} = |L_{h(j,i-1)} - L_{h(j,i)}|$
    $d_{L_{right}} = |L_{h(j,i+1)} - L_{h(j,i)}|$
    $d_{L_{top}} = |L_{v(j-1,i)} - L_{v(j,i)}|$
    $d_{L_{bot}} = |L_{v(j+1,i)} - L_{v(j,i)}|$
    $eL(j,i) \leftarrow min(max(d_{L_{left}}, d_{L_{right}}), max(d_{L_{top}}, d_{L_{bot}}))$
    $d_{ab_{left}} = (a_{h(j,i-1)} - a_{h(j,i)})^2 + (b_{h(j,i-1)} - b_{h(j,i)})^2$
    $d_{ab_{right}} = (a_{h(j,i+1)} - a_{h(j,i)})^2 + (b_{h(j,i+1)} - b_{h(j,i)})^2$
    $d_{ab_{top}} = (a_{v(j-1,i)} - a_{v(j,i)})^2 + (b_{v(j-1,i)} - b_{v(j,i)})^2$
    $d_{ab_{bot}} = (a_{v(j+1,i)} - a_{v(j,i)})^2 + (b_{v(j+1,i)} - b_{v(j,i)})^2$
    $eC(j,i) = min(max(d_{ab_{left}}, d_{ab_{right}}), max(d_{ab_{top}}, d_{ab_{bot}}))$
  fin pour
fin pour

Algorithm 10 Hirakawa : Séléction par homogénéité

Données $L_h, a_h, b_h, L_v, a_v, b_v, eC, eL, RGB_h, RGB_v, m, n$
Résultats RGB
H_h = *Hirakawa_Homogeneity_Eval(L_h, a_h, b_h, eC, eL, m, n)*
H_v = *Hirakawa_Homogeneity_Eval(L_v, a_v, b_v, eC, eL, m, n)*
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    si $H_{v(j,i)} \geq H_{h(j,i)}$ **alors**
      $RGB \leftarrow RGB_h$
    sinon
      $RGB \leftarrow RGB_v$
    fin si
  fin pour
fin pour

Algorithm 11 Hirakawa : fonction Hirakawa_Homogeneity_Eval

Evaluation locale
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    $H_{j,i} = 0$
    pour tout voisins $\in BS$ **faire**
      $Lum = |L(j,i) - L_{BS}| \leq eL(j,i)$
      $Crom = (a(j,i) - a_{BS})^2 + (b(j,i) - b_{BS})^2 \leq eC(j,i)^2$
      $H(j,i) = H(j,i) + (Lum\ and\ Crom)$
    fin pour
  fin pour
fin pour
Somme des évaluations
pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    $H_{j,i} = \sum_{kj\in[-1,1],\ ki\in[-1,1]} H_{j+kj,i+ki}$
  fin pour
fin pour
return H

Algorithm 12 Version naïve du code du filtre bilatéral Bayer

pour $j = 0; j < n; j++$ **faire**
  pour $i = 0; i < m; i++$ **faire**
    si le pixel est rouge **alors**

$$R'(j,i) = \sum_{x=-2}^{2}\sum_{y=-2}^{2} exp\frac{-\sqrt{(2x)^2+(2y)^2}}{\rho^2} \times exp\frac{-|u(j,i)-u(j+2x,i+2y)|}{h^2} R(j+2x, i+2y)$$

    sinon si le pixel est bleu **alors**

$$B'(j,i) = \sum_{x=-2}^{2}\sum_{y=-2}^{2} exp\frac{-\sqrt{(2x)^2+(2y)^2}}{\rho^2} \times exp\frac{-|u(j,i)-u(j+2x,i+2y)|}{h^2} B(j+2x, i+2y)$$

    sinon si le pixel est vert **alors**

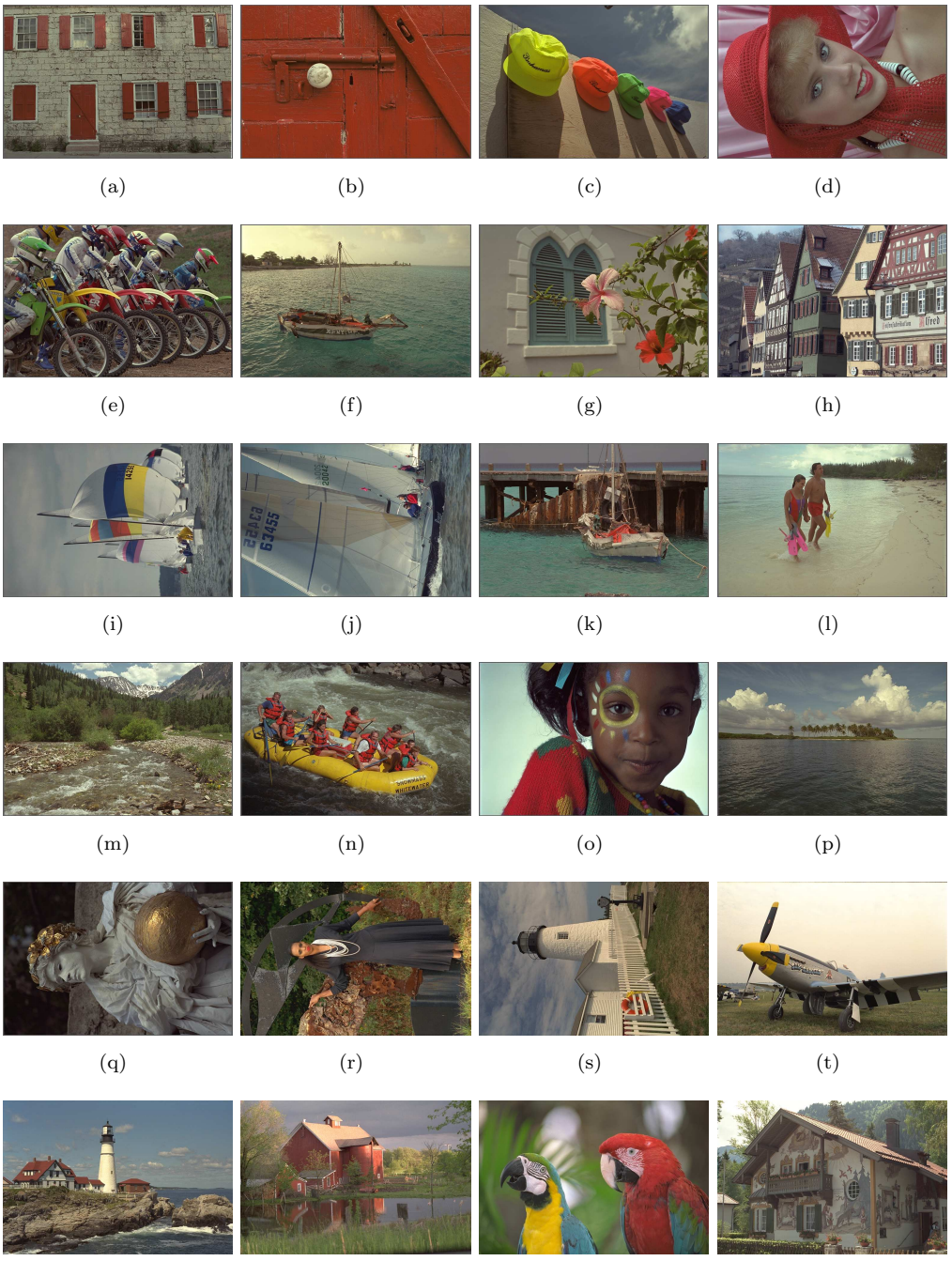
$$G'(j,i) = \sum_{x=-1}^{1}\sum_{y=-1}^{1} exp\frac{-\sqrt{(2x+1)^2+(2y+1)^2}}{\rho^2} \times exp\frac{-|u(j,i)-u(j+2x+1,i+2y+1)|}{h^2} G(j+2x, i+2y) + exp\frac{-\sqrt{(2x)^2+(2y)^2}}{\rho^2} \times exp\frac{-|u(j,i)-u(j+2x,i+2y)|}{h^2} G(j+2x, i+2y)$$

    fin si
  fin pour
fin pour

Annexe D

Base d'images Kodak PhotoCD

La figure D.1 montre les 24 images de la base de données Kodak PhotoCD. Ces 24 images ont été acquises avec un appareil-photo argentique puis scannée sur les trois palns de couleur rouge vert et bleu en suivant la procédure PhotoCD professionelle de Kodak.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

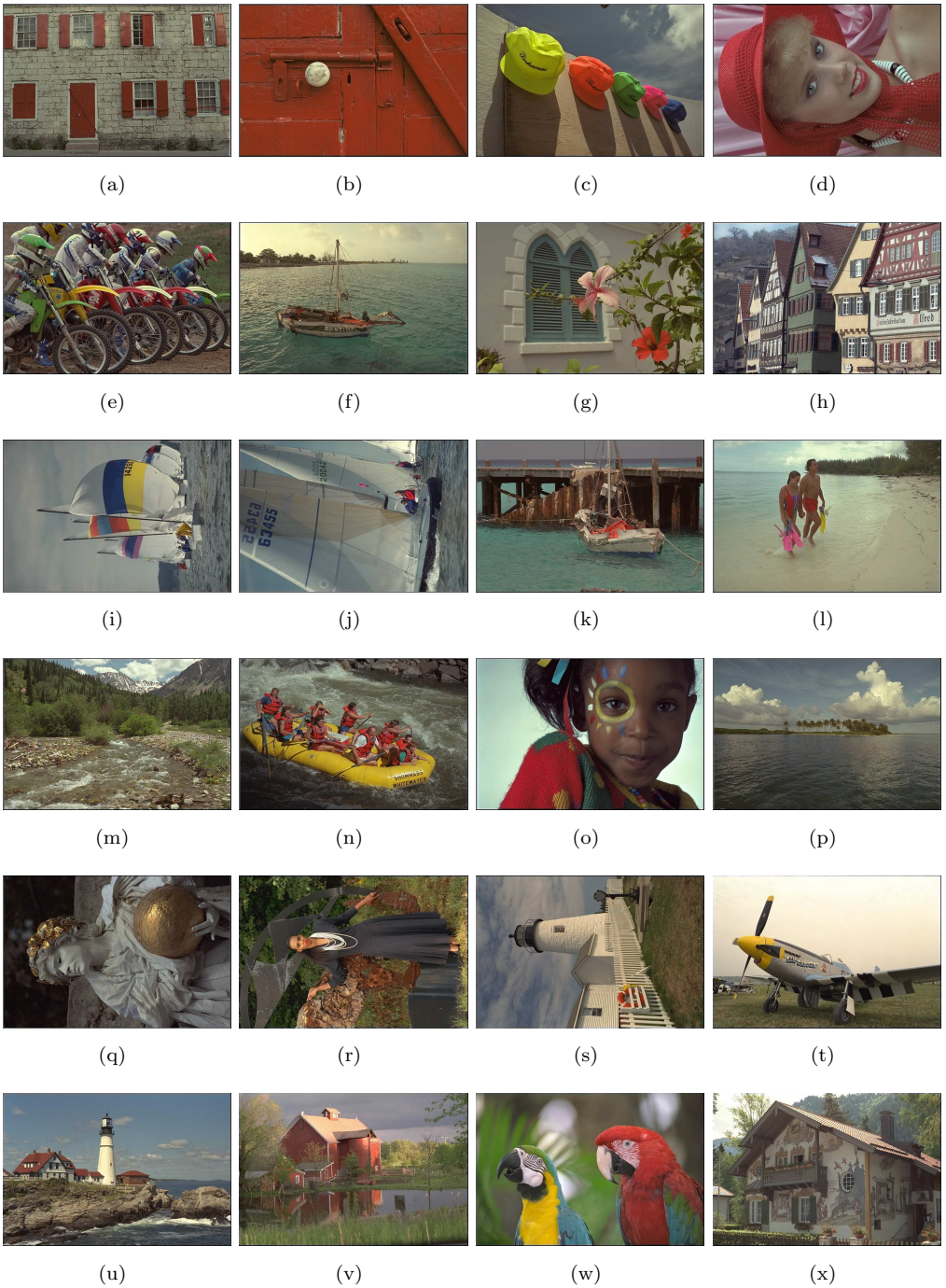
(q) (r) (s) (t)

(u) (v) (w) (x)

FIGURE D.1 – Les 24 images de la base Kodak PhotoCD

Annexe E

Application de GEDI sur l'ensemble des images de l'annexe D



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

(q) (r) (s) (t)

(u) (v) (w) (x)

FIGURE E.1 – Résulats du dématriçage avec l'algorithme GEDI sur les 24 images de la base Kodak PhotoCD de l'annexe D.

Bibliographie

[1] D.Litwiller. Ccd vs. cmos : Facts and fiction. *Photonics Spectra*, 2001.

[2] D.Alleyson. 30 ans de demosaicage - 30 years of demosaicing. *GRETSI 2004*, 21 : 561–581, 2004.

[3] B.C.deLavarène. *L'échantillonnage spatio-chromatique dans la rétine humaine et les caméras numériques*. PhD thesis, Université Joseph Fourier, 2007.

[4] H.S.Malvar, L.W. He, and Y.Yacobi. High-quality linear interpolation for demosaicing of bayer-patterned color images. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.

[5] D.R.Cok. *Signal processing method ans apparatus for sampled image signals*. us patent 4,630,307 to Eastman Kodak Compagny, Patent and Trademark office, Washington, 1986.

[6] D.R.Cock. Reconstruction of ccd images using template matching. *IS&T's 47th Annual Conference / ICPS*, pages 380–385, 1994.

[7] K.Hirakawa and T.W.Parks. Joint demosaicing and denoising.

[8] J.F.Hamilton and J.E.Adams. *Adaptive color plan interpolation in single sensor color electronic camera*. US Patent 5,629,734 to Eastman Kodak Compagny, Patent and Trademark office, Washington, 1997.

[9] C.A.Laroche and M.A.Prescott. *Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradient*. us patent 5,373,322 to Eastman Kodak Compagny, Patent and Trademark office, Washington, 1994.

[10] J.F.Hamilton and J.T.Compton. Processing color and panchromatic pixels. *Eastman Kodak Company*, 2007.

[11] B.E.Bayer. *Color Imaging Array*. United State Patent, 3,971,065, 1976.

[12] W.T.Freeman. *Median filter for reconstructing missing color samples*. US Patent 4,724,395, to Polaroid Corporation, Patent and Trademark Office, Washington, D.C., 1988.

[13] H.Phelippeau, S.Bara, M.Akil, and H.Talbot. Green edge directed interpolation. *STMicroelectronics*, 2008.

[14] H.Phelippeau, H.Talbot, S.Bara, and M.Akil. Local majoritary direction choice enhancement mehod for edge directed demosaicing. *STMicroelectronics*, 2008.

[15] H.Phelippeau, H.Talbot, M.Akil, and S.Bara. Green edge directed interpolation. *IEEE Transaction on Consumer Electronics*.

[16] C.Tomasi and R.Manduchi. Bilateral filtering for gray and color images. pages 839–846, 1998. URL citeseer.ist.psu.edu/tomasi98bilateral.html.

[17] H.Phelippeau, H.Talbot, M.Akil, and S.Bara. Shot noise adaptive bilteral filter. *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pages 864–867, 2008.

[18] K.Hirakawa and T.Parks. Adaptive homogeneity-directed demosaicing algorithm. *Image Processing, IEEE Transactions on*, 14 :360–369, 2005. URL citeseer.ist.psu.edu/article/hirakawa05adaptive.html.

[19] H.Phelippeau, M.Akil, B.D.Rodrigues, H.Talbot, and S.Bara. Bayer bilateral denoising on trimedia3270. *IS&T/SPIE Annual Symposium*, 2009.

[20] J.Tarrant. *Understanding Digital Cameras : Getting the Best Image from Capture to Output*. Focal Press, 2007.

[21] URL http://www.photokina-cologne.com/index.php.

[22] URL http://fr.wikipedia.org/wiki/Application_Specific_Integrated_Circuit.

[23] URL http://fr.wikipedia.org/wiki/Joint_Photographic_Experts_Group.

[24] . URL http://www.epi-centre.com/reports/9403cdi.html.

[25] . URL http://en.wikipedia.org/wiki/Apple_QuickTake.

[26] URL http://www.wherry.com/gadgets/qv10/.

[27] URL www.pcpro.co.uk/reviews/2077/olympus-camedia-c800l.html.

[28] URL http://www.olympus.fr/consumer/29_C-1400L.htm.

[29] URL http://www.canon.fr/for_home/product_finder/cameras/digital_slr/eos_300d/index.asp.

[30] R.Lukac. *Single Sensor Imaging : Methods and Applications for Digital Cameras*. CRC Press, 2008.

[31] E.R.Fossum, P.C.Hung, T.Koyama, T.Mizoguchi, J.Nakamura, K.Sato, I.Takayanagi, K.Toyoda, S.Watanabe, T.Yamada, and Hideaki Yoshida. *Image sensors and signal processing for digital still cameras*. CRC Press, 2006.

[32] P.B.Catrysse. *The Optics Of Image Sensors*. PhD thesis, Stanford University, 2003.

[33] F.A. Jenkins and H.E.White. *Fundamentals of optics 4th ed.* 1976.

[34] E.Hecht. *Optics*. 1998.

[35] S.F.Ray. *Applied photographic optics, 3 rd ed.* 2002.

[36] W.J.Smith. *Modern optical engineering, 3rd ed.* 2000.

[37] New York John Wiley & Sons, editor. *Physics of Semiconducteur Devices*. 1981.

[38] C.Cavadore. *Conception et caractérisation de capteurs dimages à pixels actifs CMOS-APS*. PhD thesis, Supaero, 1998.

[39] Y.Ishihara and K.Tanigaki. A high photosensitivity il-ccd image sensorwith monolitic resin lens array. *IEDM tech. Dig*, pages 497–500, 1983.

[40] H. Senda N. Niisoe H. Aoki T. Otagaki Y. Shigeta M. Asaumi Y. Miyata Y. Sano T. Kuriyama A. Tsukamoto, W. Kamisaka and S. Terakawa. High-sensitivity pixel technology for a 1/4-inch pal 430-kpixel it-ccd. *IEEE Custom Integrated Circuit Conference*, pages 39–42, 1996.

[41] R.Lukac and K.N.Plataniostis. Universal demosaicking for imaging with an rgb color filter array. *Pattern Recognition*, 38 :2208–2212, 2005.

[42] R.Lukac and K.N.Plataniostis. Universal demosaicking for imaging pipelines with an rgb color filter array. *Pattern Recognition*, 38 :2208–2212, 2005.

[43] A.J.Theuwissen. *Solid-State Imaging with Charge-Coupled Devices*. Edition Kluwer Academic Publishers, 1995.

[44] B.K.Gunturk, J.Glotzbach, Y.l Altunbasak, R.W.Schafer, and R. M.Mersereau. Demosaicking : Color filter array interpolation. *IEEE Signal processing magazine*, 22 :44–54, 2005.

[45] J.A.Weldy. Optimized design for a single-sensor color electronic camera system. *Proc. SPIE*, 1071 :300–307, 1988.

[46] J.E.Adams. Interactions between color plane interpolation and other image processing functions in electronic photography. *Proc. SPIE*, 2416 :144–151, 1995.

[47] R.H.Hibbard. *Apparatus and method for adaptively interpolating a full color image utilizing luminance gradient.* us patent 5,382,976 to Eastman Kodak Compagny, Patent and Trademark office, Washington, 1995.

[48] W.T.Freeman. *Method and apparatus for reconstructing missing color samples.* us patent 4,774,565 to Polaroid Corporation, Cambridge, Mass, 1988.

[49] R. Kimmel. Demosaicing : image reconstruction from color ccd samples. *Image Processing, IEEE Transactions on*, 8(9) :1221–1228, 1999.

[50] S.C.Pei and I.K.Tam. Effective color interpolation in ccd color filter array using signal correlation. *IEEE Trans. Circuits Syst. Video Technol.*, 13 :503–513, 2003.

[51] D.R.Cock. Single-chip electronic color camera with color dependent birefringent optical spatial frequency filter and red and blue signal signal interpolating circuit. *United States Patent number 4,605,956*, 1986.

[52] H.D.Crane, J.D. Peter, and E.Martinez-Uriegas. Method and apparatus for decoding spatiochromatically multiplexed color images using predetermined coefficients. *US patent 5,901,242 to SRI International, Patent and Trademark Office Washington D.C*, 1999.

[53] R.Ramanath and W.E.Snyder. Adaptive demosaicking. *Journal of Electronic Imaging*, 12(4) :633–642, 2003.

[54] D.Alleyson, S.Süstrunk, and J.Herault. Color demosaicing by estimating luminance and opponent chromatic signals in the fourier domain. *Proc. Color Imaging Conf : Color Science, Systems, Applications*, pages 331–336, 2002.

[55] B.K.Gunturk, Y.Altunbasak, and R.M.Mesereau. Color plane interpolation using alternating projections. *IEEE transactions on image processing*, 11 :997–1013, 2002.

[56] J.W.Glotzbach, R.W.Schafer, and K.Illgner. A method of color filter array interpolation with alias cancellation properties. *IEEE International Conference on Image Processing*, pages 141–144, 2001.

[57] D.H.Brainard. Bayesian method for reconstructing color images from trichromatic samples. *47th Annual Conference ICPS, Rochester*, pages 375–376, 1994.

[58] D.H.Brainard and D.Sherman. Reconstructing images from trichromatic samples, from basic research to practical applications. *Proc. IS&T/SID 3rd Color Imaging Conference*, pages 4–10, 1995.

[59] D.Taubman. Generalized wiener reconstruction of images from colour sensor data using a scale invariant prior. *International conference on image processing*, 3 : 801–804, 2000.

[60] X.Li and M.T.Orchard. New edge-directed interpolation. *IEEE transaction on Image Processing*, 10, 2001.

[61] Z.Wang, A.C.Bovik, and L.Lu. Why is image quality assesment so difficult. *IEEE international conference on Acoustics, Speech, Signal Processing*, 4, 2002.

[62] B.Girod. What's wrong with mean-squarred error. *Digital Images and Human Vision, A.A. Watson, Ed Cambridge, MA : MIT Press*, 1993.

[63] W.Lu and Y.P.Tan. Color filter array demosaicking : New method and performance measures.

[64] Kodak. Ccd image sensor noise sources. 2005.

[65] Charasteristics and use of fft ccd area image sensor, hamamatsu photonics.

[66] R.Costantini and S.Susstrunk. Virtual sensor design. *Proc. IS&T/SPIE Electronic Imaging 2004 : Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications*, 5301 :408–419, 2004.

[67] G.E.Healey and R.Kondepudy. Radiometric ccd camera calibration and noise estimation. *IEEE transaction on pattern analysis and machine intelligence*, 16 : 267–276, 1994.

[68] J.Medkeff. Using image calibration to reduce noise in digital images. URL http://photo.net/learn/dark_noise.

[69] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6) :2325–2383, 1998.

[70] M.Antonini and V.Ricordel. *Chapiter "Quantification", Traité IC2*. Hermès, 2002.

[71] R.L.Baer. A model for dark current charactezation and simulation. *Proceeding of the SPIE*, 6068, 2006.

[72] H.Faraji and W.J.MacLean. Ccd noise removal in digital images. *Transaction on image processing*, 15, 2006.

[73] J.R.Janesick. Scientific charge coupled devices. *Bellingham SPIE*, 2001.

[74] J.Buzzy and F.Guichard. Noise in imaging chains. *ICIP*, 2005.

[75] H.T.Hytti. Characterization of digital image noise properties based on raw data. *SPIE-IS&T Electronic Imaging, Image Quality and System Performance*, 2005.

[76] A.Bosco, A.Bruna, G.Messina, and G.Spampinato. Fast method for noise level estimation and integreted noise reduction. *IEEE transactions on Consumer Electronics*, 51, 2005.

[77] Ce Liu, R Szeliski, Sing Bing Kang, C.L. Zitnick, and W.T. Freeman. Automatic estimation and removal of noise from a single image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30 :299–314, 2008.

[78] Noura Azzabou. *Variable Bandwidth Image Models for Texture-Preserving Enhancement of Natural Images*. PhD thesis, Ecole centrale de Paris, 2008.

[79] A.Buades. *Image and film denoising by non-local means*. PhD thesis, Universitat de les Illes Balears, 2008.

[80] J.S.Lee. Speckle analysis ans smoothing of qynthétique aperture radar images. *Computer Graphics ans image Processing*, 17 :24–32, 1981.

[81] D.T.Kuan, A.A.Sawchuk, T.V.Strand, and P.Chavel. Adaptive noise smoothing filter for images with signal-dependent noise. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 7 :165–177, 1985.

[82] J.S.Lim. Tow-dimensional signal and image processing. *Englewood Cliffs, NJ, Prentice Hall*, 1990.

[83] I.Bloch, Y.Gousseau, H.Maître, D.Matignon, B.Pesquet-Popescu, F.Schmitt, M.Sigelle, and F.Tupin. *Le Traitement des images - tome 2.* .

[84] J.Astola, P.Haavisto, and Y.Neuvo. Vector median filters. *Proceedings of the IEEE*, 78(4) :678–689, 1990.

[85] M.Nagao and T.Matsuyama. Edge preserving smoothing. *Computer graphics and image processing*, 9 :394–407, 1979.

[86] Y.Wu and H.Maître. Smoothing speckled synthetic aperture radar images by using maximum homogeneous region filter. *Optical Engineering*, 31 :1785–1792, 1992.

[87] D.C.C.Wang, A.H.Vagnucci, and C.C.Li. Gradient inverse weighted smoothing scheme and the evaluation of its performance. *Computer Graphics and image processing*, 15 :167–181, 1981.

[88] S.M.Smith and J.M.Brady. Susan a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1) :45–78, 1997. ISSN 0920-5691.

[89] D.Harwood, M.Subbarao, H.Hakalahti, and L.S. Davis. A new class of edge-preserving smoothing filters. *Pattern Recogn. Lett.*, 6(3) :155–162, 1987. ISSN 0167-8655.

[90] J.S.Lee. Digital image smoothing and sigma the filter. *Computer Graphics and Image Processing*, 24 :255–269, 1983.

[91] V.Aurich and J.Weule. Non-linear gaussian filters performing edge preserving diffusion. *Proceedings of the DAGM Symposium*, 1995.

[92] I.Bloch, Y.Gousseau, H.Maître, D.Matignon, B.Pesquet-Popescu, F.Schmitt, M.Sigelle, and F.Tupin. *Le Traitement des images - tome 1.* .

[93] L.Rudin, S.Osher, and E.Fatemi. Nonlinear total variation based noise removal algorithm. *Physica D*, 60 :259–268, 1992.

[94] P.Perona and J.Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 :629–639, 1990.

[95] D.Tschumperlé. *PDE's based regularization of multivalued images and applications*. PhD thesis, Iniversité de Nice-Sophia Antipolis, 2002.

[96] D.L.Donoho and I.M.Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81 :425–455, 1994.

[97] J.S.DeBonet. Noise reduction through detection of signal redundancy. *Rethinking Artificial Intelligence*, MIT AI Lab, 1997.

[98] J.S.DeBonet, P.A.Viola, and J.Fisher. Flexible histograms : A mult-resolution target discrimination model. *SPIE, Automatic Target Recognition*, 3371 :519–530, 1998.

[99] J.S.DeBonet. Multiresolution sampling procedure for analysis and synthesis of texture images. *Computer GRaphics*, pages 361–368, 1997.

[100] Z.Wang and D.Zhang. Restoration of impulse noise corrupted images using long-range correlation. *IEEE Signal Processing Letters*, 4 :4–7, 1998.

[101] D.Zhang and Z.Wang. Image information restoration based on long-range correlation. *IEEE Transaction on Circuits and systems for video Technomogy*, 12 : 331–341, 2002.

[102] A.Criminisi, P.Perez, and K.Toyama. Region filling and object removal by exemplar- based impaiting. *IEEE Transactions on Image Processing*, 13 :12001212, 2004.

[103] A.Buades, B.Coll, and J-M.Morel. On image denoising methods. *Internal Report*, 2004. URL http://www.cmla.ens-cachan.fr/Cmla/Publications/2004/CMLA2004-15.pdf.

[104] A.Buades. *Image and film denoising by non-local means*. PhD thesis, Iniversité des îles Baléares, 2006.

[105] S.Paris and F.Durand. A fast approximation of the bilateral filter using a signal processing approach. *European Conference on Computer Vision*, 2006.

[106] C.Sun. Fast algorithm for local statistics calculation for n-dimensional images. *Journal of real time imaging*, 7 :519–527, 2001.

[107] M.Shinozuka and C.M.Jan. Digital simulation of random processes and its applications. *Journal of sound and vibration*, 25-1 :11–128, 1972.

[108] D.E.Knuth. *The Art Of Computer Programming*. Addison-Wesley, 1999.

[109] F.J.Anscombe. The transformation of poisson, binomial and negative-binomial data. *Biometrika*, 35 :246–254, 1948.

[110] T.Halfhill. Philips power up for video. *IN processor report*, 2003. URL http://www.mpronline.com.

[111] J.L.Hennessy and D.A.Patterson. Computers architecture : A qantitative approach 3rd edition,morgan kauffman. 2003.

[112] S.Rathman and G.Slavenburg. An architecural overview of the programmable multimedia processor, tm-1. *Procedings of the COMPCOM'96*, pages 319–326, 1996.

[113] I.Yasui and Y.Shimazu. Microprocessor with harvard architecture, 1991. URL http://www.freepatentsonline.com/5034887.html.

[114] S.Dutta, I.Mehra, Z.Weiwen, D.Singh, M.Janssens, R.Vengalasetti, B.Ben-Nun, V.Adusumilli, J.Y.H.Huang, L.Ling, C.Nelson, B.Jain, and S.Wu. Architecure and design of a talisman-compatible multimedia processor. *IEEE Transactions on Circuits and Systems for Video Technology*, 9 :565–579, 1999.

[115] J.Montrym and H.Moreton. The geforce 6800. *IEEE MICRO Magazine*, 38 : 114–117, 2005.

[116] P.Sunna. Avc/h.264, un système de codage vidéo évolué pour la hd et la sd. *Union Européenne de la radio-télévision*, pages 54–58, 2005.

[117] Mpeg la terms of h.264/mpeg-4 avc patent license. URL http://www.mpegla.com/news/n_03-11-17_avc.html.

[118] Philips nexperia pnx4103. 2006. URL http://www.hotchips.org/archives/hc18/2_Mon/HC18.S1/HC18.S1T3.pdf.

[119] B.Kernighan and D.Ritchie. *The C programming Language*. Prentice-Hall, 1978.

[120] H.Schildt. *C++ The Complete Reference Third Edition*. Osborne McGraw-Hill, 2000.