Image processing for a RGB-Z mixed matrix

Valentin Rebiere



Sorbonne Université

École Doctorale Informatique, Télécommunications et Électronique
ED130

Laboratoire d'informatique de Paris 6

Image processing for a RGB-Z mixed matrix

Thèse de Doctorat

Par Valentin REBIERE

Dirigée par Andréa Pinna
Encadrée par Antoine Drouot

Présentée et soutenue le 30 juin 2021

Devant un jury composé de :

Virginie Fresse	Université Jean-Monnet-Saint-Étienne	Rapporteuse
Gilles Sicard	CEA-Leti	Rapporteur
Catherine Achard	Sorbonne Université	Examinatrice
François Berry	Institut Pascal	Examinateur
Christine Guillemot	INRIA	Examinatrice
Andrea Pinna	Sorbonne Université	Directeur
Bertrand Granado	Sorbonne Université	Co-Encadrant
Antoine Drouot	STMicroelectronics	Co-Encadrant

Remerciements

Je voudrais tout d'abord remercier grandement mon directeur de thèse, Andréa Pinna, ainsi que mes encadrants, Antoine Drouot et Bertrand Granado, pour toute leur aide. Je suis ravi d'avoir travaillé en leur compagnie car outre leur appui scientifique, ils ont toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse. Je tiens également à remercier Arnaud Bourge qui a grandement participer à l'encadrement de ce travail de recherche. Merci pour votre bienveillance à tous les quatre.

J'adresse tous mes remerciements à Virginie Fresse ainsi qu'à Gilles Sicard de l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de cette thèse.

Je tiens également à remercier les équipes du CEA Leti pour leur collaboration dans le projet plus large dans lequel s'inscrit mon travail de thèse. Merci à Clémence Jasmin pour le partage des premières images réelles et Regis Perrier pour le partage de ses travaux.

Je remercie Christine Guillemot, Catherine Archard et François Berry pour l'honneur qu'ils me font d'être dans mon jury de thèse.

Je tiens à remercier également les collègues du site de Paris de STMicroelectronics ainsi que du Lip6 qui ont été très sympathiques pendant ces trois années et m'ont permis d'évoluer dans un cadre dynamique de travail : Stéphane Auberger, Gwladys Hermant, Antoine Chouly, Pol Perrin, Heloïse Gresset, Olivier Pothier, Victor Macela, Thierry Lebihen, Estelle Lesellier, Cedric Lecoutre, Nicolas Florenchie, Jules Barnoud, Charlotte Jaffre, Khalil Hachicha, Orlando Chuquimia, Sylvain Feruglio, Farouk Vallette, Thomas Garbay, Julien Denoulet, Amine Rhouni, Olivier Tsiakaka, Songlin Li, Faten Sahel, Sylvain Takougang, Petr Dobias...

Je remercie aussi les doctorants et ingénieurs de l'ISIR et notamment mes amies Mégane Millan et Angélina Bellicha avec qui j'ai partagé mes études et notamment ces années de thèse.

Mes derniers remerciements vont à tous mes amis et ma famille qui m'ont apporté un soutien indéfectible, et notamment durant ces derniers mois.

PhD Thesis of
Sorbonne Université

Image processing for a RGB-Z mixed matrix

Abstract:

A RGB-Z system is a Vision System-on-Chip (V-SoC) that captures both color and depth information. It generally consists of two sensors: one for the color acquisition and one for the depth information acquisition. In last years they have been increasingly used: Microsoft's Kinect or the iPhone X with facial recognition are great examples. Recently, the idea of combining color and depth acquisition at the pixel level and in the same matrix has emerged. Such a sensor would allow a more compact integration while reducing the calibration problems compared to a two-sensor system. However, new constraints at different technical levels are raised. In this thesis, we focus on the problem of reconstructing the missing color information due to the heterogeneity of the pixel matrix. There is currently no monolithic RGB-Z sensor on the market as described in the thesis, and the state of the art is not exhaustive on the subject.

First, we propose different RGB-Z matrices. These matrices are based on two different Z-pixel architectures that are provided by the CEA. The distribution and the size of the Z pixels vary according to the proposed matrices.

In a second step, several algorithms for reconstructing missing information inspire by the state of the art are adapted and implemented to the proposed matrices. These solutions are not satisfactory and exhibit color and structural artifacts. We propose an original adaptive algorithm to reconstruct the color information using a new operator called *semi-gradient*. This operator allows to better reconstruct the image features such as edges or thin structures. To do so, it studies the close neighborhood of the missing pixel to determine to which texture it belongs. The different methods are evaluated in simulation and the results show a significant improvement of the quality of the reconstructed image with the method using semi-gradients.

Finally, a first hardware implementation is proposed to evaluate the latency of the algorithm using semi-gradients. For the moment, the solution is computationally complex. However, parallelism proposals could improve this point.

As a general conclusion, the results of this thesis allow to set up a simulation environment that facilitates the evaluation of new RGB-Z architectures and the associated reconstruction algorithms. The efficiency of the proposed solution using semi-gradient is demonstrated compared to the state-of-the-art adapted reconstruction solutions. The quality of the reconstructed image is significantly improved, getting closer to the quality of a classic RGB sensor. However, there are still some artifacts present for the reconstruction of particular structures such as corners or very fine structures.

Thèse de Doctorat de
Sorbonne Université

Image processing for a RGB-Z mixed matrix

Résumé :

Un système Red, Green, Blue and Depth (RGB-Z) est un système de vision sur puce (V-SoC) qui capture à la fois les informations de couleur et de profondeur. Il se compose généralement de deux capteurs : un pour l'acquisition des couleurs et le second pour l'acquisition des informations de profondeur. Ces dernières années on les utilise de plus en plus : la Kinect de Microsoft ou l'iPhone X avec reconnaissance faciale en sont de bons exemples. Récemment, l'idée de combiner l'acquisition de la couleur et de la profondeur au niveau du pixel et dans la même matrice a émergé. Un tel capteur permettrait une intégration plus compacte tout en réduisant les problèmes de calibration par rapport à un système composés de deux capteurs. Cependant, une telle intégration soulève de nouvelles contraintes à différent niveaux techniques. Dans cette thèse, nous nous focalisons sur les problèmes de reconstruction de l'information de couleur manquante due à l'hétérogénéité de la matrice de pixels. Actuellement, il n'existe pas sur le marché de capteur RGB-Z monolithique tel que décrit dans la thèse. De plus l'état de l'art n'est pas exhaustif sur le sujet.

Dans un premier temps, nous avons proposé différentes matrices RGB-Z. Ces matrices sont basées sur deux architectures de pixels Z différentes fournies par le CEA. La distribution et la taille des pixels Z varient en fonction des matrices proposées.

Dans un deuxième temps, plusieurs algorithmes de reconstruction des informations manquantes inspirés de l'état de l'art ont été adaptés et implémentés aux matrices proposées. Ces solutions ne sont pas satisfaisantes et présentent des artefacts de couleur et structurels. Nous proposons un algorithme adaptatif original pour reconstruire l'information de couleur manquante en utilisant un nouvel opérateur appelé *semi-gradient*. Cet opérateur permet de mieux reconstruire les structures situées le long des bords fins. Pour ce faire, il étudie le voisinage proche du pixel manquant pour déterminer à quelle texture il appartient. Les différentes méthodes ont été évaluées en simulation et les résultats montrent une amélioration significative de la qualité de l'image reconstruite avec la méthode utilisant les semi-gradients.

Enfin, une première implémentation matérielle est proposée pour évaluer la latence de l'algorithme utilisant les semi-gradients. Pour l'instant, la solution est complexe, cependant, des propositions de parallélisme pourraient améliorer ce point.

En conclusion générale, les résultats de cette thèse ont permis de mettre en place un environnement de simulation qui facilite l'évaluation de nouvelles architectures RGB-Z et des algorithmes de reconstruction associés. L'efficacité de la solution proposée utilisant les semi-gradient est démontrée par rapport aux solutions de reconstruction adaptées de l'état de l'art. La qualité de l'image reconstruite est

significativement améliorée, se rapprochant de la qualité d'un capteur RGB classique. Cependant, il reste des artefacts présents lors la reconstruction de structures particulières tel que les coins ou des structures très fines.

Contents

List of Figures

List of Tables

Acronyms

APS	Active Pixel Sensor.
AR/VR	Augmented Reality / Virtual Reality.
ASIC	Application Specific Integrated Circuit.
BSI	Backside-illuminated.
CCD	Charge Coupled Device.
CCM	Color Correction Matrix.
CDFA	Color Depth Filter Array.
CNN	Convolutional Neural Network.
C-PSNR	Color Peak signal-to-noise-ratio.
CDTI	Capacitive Deep Trench Isolation.
CFA	Color Filter Array.
CMOS	Complementary Metal-Oxide-Semiconductor.
d-ToF	direct-Time of Flight.
EDI	Edge-Based Directional Interpolation.
FF	Fill Factor.
FPGA	Field-Programmable Gate Array.
FSI	Frontside-illuminated.
HDL	Hardware Description Language.
HDR	High Dynamic Range.
HLS	High Level Synthesis.
HVS	Human Visual System.
IoT	Internet of Things.
i-ToF	indirect-Time of Flight.
IQA	Image Quality Assessment.
IR	Infra-Red.
ISP	Image Signal Processor.
LUT	Lookup Table.
MSE	Mean Squared Error.
M-SSIM	Multi scale Structural Similarity index.

NIR	Near Infra-Red.
PPS	Passive Pixel Sensor.
PSNR	Peak signal-to-noise-ratio.
RGB	Red, Green and Blue.
RGB-IR	Red, Green, Blue and Infra Red.
RGBW	Red, Green, Blue and White.
RGB-Z	Red, Green, Blue and Depth.
RTL	Register-Transfer Level.
SG	Semi-Gradient.
sRGB	standard RGB (ISO 12640-2:2004).
SSIM	Structural Similarity index.
ToF	Time of Flight.
VSoC	Vision System on Chip.
Z	Depth.

Introduction

Image capture systems are becoming more and more complex, not only in terms of volume of data generated, but also in terms of quality [28] and integration of new functionalities [29]. Today, the intimate cohabitation of signal acquisition and processing on the same chip, makes it possible to create a new generation of Vision System on Chip (VSoC). This new generation of sensors captures elements other than a flat color image (RGB), such as its depth (Depth (Z)), using different techniques: active or passive stereo-vision, structured light and Time of Flight (ToF). Since the development of the *Microsoft Kinect* [30], many applications using both color and depth information, like gesture recognition, virtual reality, 3D mapping and modeling applications [31], have been proposed. More recently the face recognition of the *iPhoneX* [32] using both 2D and 3D sensing has introduced the need of integrating a RGB-Z system in mobile phone.

For now, such RGB-Z systems are comprised of two different sensors. The first sensor is a classical color image sensor while the second acquires depth information. In the past few years [7], thanks to advances in semiconductor technology, the idea of combining both color and depth sensing at the pixel level and in the same matrix has emerged. This would provide an elegant, integrated, and compact solution. For example, smartphone manufacturers [33] are torn between the need to add more and more sensors / cameras for more functionalities on the one hand, and the will to limit the number of slots (as many holes in the shell) for design reasons on the other hand. The example of the smartphone can be extended to other areas such as autonomous navigation (drones, robots) and Internet of Things (IoT), virtual or augmented reality headsets, or even the automobile and medical devices. An other point is the intrinsic spatial co-location of these different data: in a 2-sensor system (RGB on one side, Z on the other), a calibration / rectification phase is necessary to correctly align the data. A risk of error is present, and in any case it consumes time and resources. Moreover, it is still sensitive to occlusions. These problems would not exist with a single RGB-Z imager.

This work is part of a larger project exploring monolithic RGB-Z sensor system design. Due to the absence of existing circuit and technology on the market, it aims at achieving a complete chain of simulation and reconstruction of a RGB-Z

imager. This work is done in parallel with the realization of a first prototype of a RGB-Z sensor by ST with the collaboration of CEA. Initially, it was planned to work with images from this sensor, however the first real data acquired with a RGB-Z test chip arrived only at the very end of the thesis. The objective is to anticipate and evaluate the impacts of the different technological issues concerning the reconstruction of color and depth images through an algorithmic performances evaluation approach.

The innovative aspects of this work are the exploration and proposition of mixed matrix architectures and the implementation of methods for reconstructing missing information for the proposed mixed matrices.

This work is a CIFRE[1] thesis between the algorithm team of the Imaging division from STMicroelectronics and the SYEL[2] team of the Lip6[3] at Sorbonne University. STMicroelectronics is a French-Italian multinational electronics and semiconductors manufacturer. It is one of the world's leading players in the semiconductor production industry. The Lip6 is a computer science research laboratory dedicated to the modeling and the resolution of fundamental problems driven by applications, as well as to the implementation and the validation through academic and industrial partnerships.

Thesis outline

In Chapter 1, the field of imaging for color and depth is introduced. First an overview of color image sensing is provided. Then technologies used to acquire depth information are presented. Finally we introduce monolithic RGB-Z systems and the problematic of the thesis.

In Chapter 2, the state-of-the-art on missing pixel reconstruction methods is described. Demosaicing methods are firstly introduced then we describe known methods used for RGB-Z systems.

In Chapter 3, we present several RGB-Z architectures with different CDFA patterns and different Z-pixel sizes. Some universal reconstruction methods are adapted and tested on our RGB-Z configurations. Finally, we introduce a new missing pixel reconstruction method dedicated to a specific RGB-Z matrix.

In Chapter 4, the different reconstruction methods and the RGB-Z architectures

[1]CIFRE stands for *Convention Industrielles de Formation par la REcherche*, i.e. Industrial Agreement of Training through Research. The research undertaken by a CIFRE fellow is within the framework of a public and private partnership between a French company and a laboratory and is formulated by both parties.

[2]SYEL stands for *SYstemes ELectroniques*, i.e. Electronic Systems.

[3]Lip6 stands for *Laboratoire d'informatique de Paris 6*, i.e. Computer Lab of Paris 6.

are compared and analyzed. The test environment and the databases are introduced and the simulation results are presented. Then, we use High Level Synthesis (HLS) tools to evaluate the resources, performances and latency of our solution for a hardware implementation on a Field-Programmable Gate Array (FPGA) target.

Finally, the conclusion and the ways of improvement of this exploratory work are discussed.

Chapter 1

Context and Problematic

1.1 Introduction

This chapter introduces the bases of the CMOS image sensing and the range sensing. It is important to analyze the characteristics of both systems to identify their constraints and understand the challenges of RGB-Z systems.

1.2 Color Imaging

Digital color imaging is the process by which photons, in the visible range of the electromagnetic spectrum, produce an image with color information at each of its pixels. Usually the process is separated into two steps: acquisition step and processing step. The acquisition step corresponds to the image sensor that produces a raw signal which is then processed by an ISP, which is a processing chain that produces a viewable color image from the raw data.

1.2.1 Image Sensor

An imager (or image sensor) is a system that is sensitive to visible light and converts it into an electrical signal. There are two main types of imaging sensors: Charge Coupled Device (CCD) sensors and CMOS sensors. Both CCD and CMOS sensors are based on MOS technology. The first ones initially allowed to have a better image quality but they were gradually replaced by the second ones in small consumer products because CMOS technology allows lower power consumption, better integration of the camera on a chip, and lower manufacturing costs [34]. Since we are looking for a mixed sensor addressing consumer applications with large volumes (smartphone market, Augmented Reality / Virtual Reality (AR/VR) or autonomous navigation), we focus on CMOS technology, which allows a better integration.

An imager is conventionally composed of pixels, arranged in a matrix, and associated with control and readout circuits as shown in Figure 1.1.

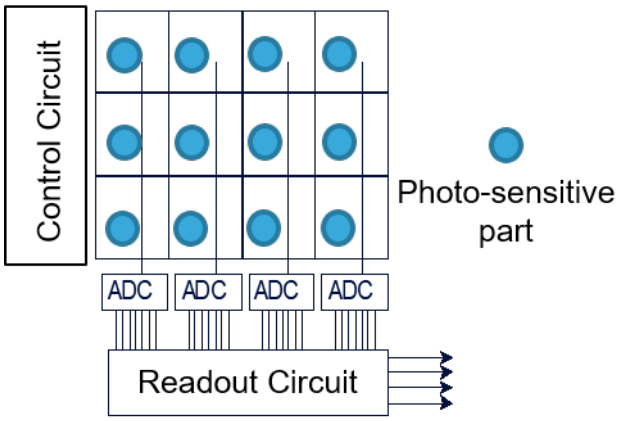


Figure 1.1: Structure of a typical imager.

1.2.1.1 Pixel

A pixel is the unitary element of a sensor. Its principal component is the photodiode, which is the photosensitive[1] part. It converts light into electrical current in proportion to the quantity of photons absorbed. However, the photodiode does not correspond to the whole surface of the pixel. There are also electronic elements such as the transistors that allow the polarization, the accumulation, the readout of the electric signal generated by the photons integration, or their routing.

Several types of pixels exist. For example the first CMOS pixel invented is a PPS [35]. It is composed of a photodiode and a transistor used as a switch. A schematic representation of a PPS is shown on the Figure 1.2.a. During the integration time of the light, the transistor is opened, allowing the photodiode to store its photogenerated charges. Then, to read the pixel value, the transistor switches: the photodiode is reset to its high voltage, and simultaneously the value corresponding to the received amount of light is read. Thus, reading such a pixel destroys the information it contained. Another type of pixel is Active Pixel Sensor (APS). A well known example is the 3T-APS [36], it is composed of a photodiode and three transistors: a selection transistor, a source follower (SF) transistor and a reset transistor. The SF transistors is used to transmit the voltage of the photodiode to the pixel output when the selection transistor is closed. The reset of the photodiode is no longer done directly by the reading, the dedicated reset transistor is used. It only resets the voltage of the photodiode to the high supply voltage before the next integration. Thus the information is not destroyed by the reading, it can be read

[1]There are also photo-transistors but we do not develop them in this work.

again if needed. A schematic representation of a 3T-APS is shown on the Figure 1.2.b. More details on the different types of pixels can be found in [37].

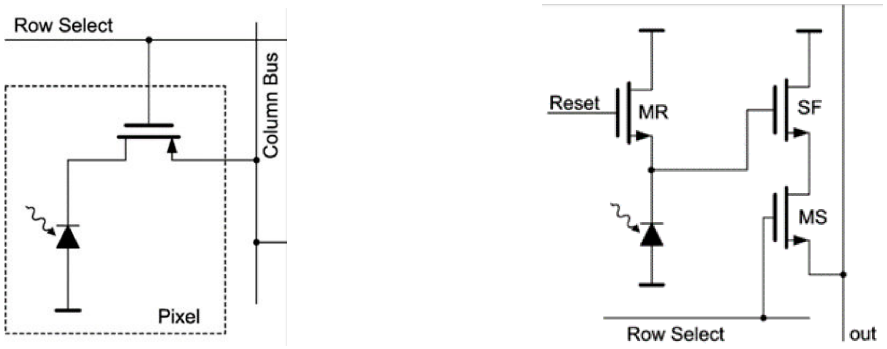
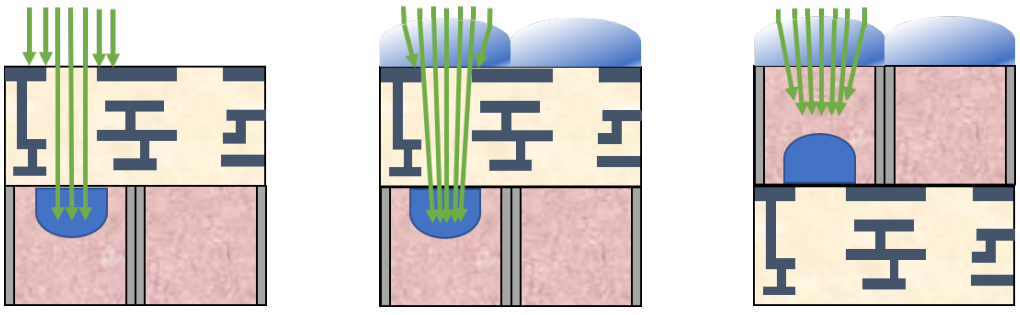


Figure 1.2: Schematic of two types of CMOS pixel [1]. (a) PPS. (b) 3T-APS.

Then, the pitch and Fill Factor (FF) are important geometrical parameters of a pixel. The pitch corresponds to the total pixel width: photosensitive part plus the electronic elements. The FF corresponds to the area of the photosensitive area over the total area of the pixel including other electronic elements (1.1). For a given pitch, the larger the FF, the more photons are integrated. It is often expressed as a percentage.

$$FF = \frac{A_{photopart}}{A_{tot}} \tag{1.1}$$

Where $A_{photopart}$ corresponds to the area of the photosensitive part and A_{tot} corresponds to the total area of the pixel. Different techniques are used to artificially increase the FF. A micro-lens can be added to converge the light rays in the photo-sensor. In addition, the circuit can be illuminated from the back side (BSI). This allows the light to reach the photo-sensor directly, without having to pass through the metal layers as it is the case with a FSI device (Figure 1.3).



(a) (b) (c)

Figure 1.3: Different CMOS pixel representations: (a) FSI with no micro-lens, (b) FSI with a micro-lens, (c) BSI with a micro-lens.

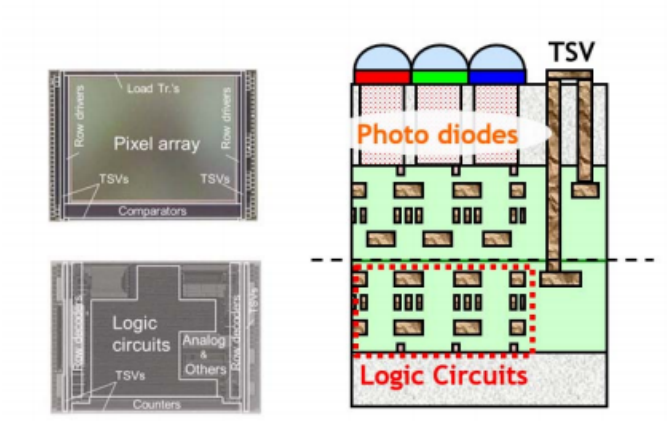


Figure 1.4: Layouts and schematic diagrams of a BSI 3D-stacked sensor [2].

1.2.1.2 Pixel Array

An image sensor is thus composed of multiple juxtaposed pixels. It forms a pixel array that produces a spatially sampled image. To have a great spatial resolution, it is important to have a lot of pixels. However, the more pixels there are, the larger the sensor is. To counter this phenomenon, the pixel pitch is reduced, but this also reduces the photosensitive surface of each pixel. More recently, 3D-stacked technology [2] has been introduced. It allows to implement different logic process or electronics functions in the substrate instead of a blank carrier wafer. The Figure 1.4 illustrates how the pixel array part and the logic circuit part are stacked. This technology increases the photosensitive area by dissociation of pixel array and electronic elements between two layers. This way, the FF is improved and the size of the chip can be reduced while maintaining a sufficient photosensitive surface. It is also possible to integrate more processing directly in the sensor, in the non-photosensitive layer.

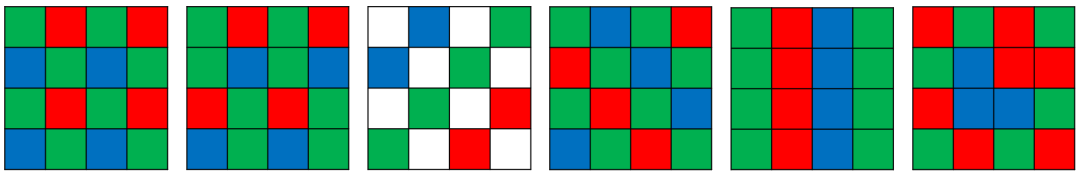
1.2.1.3 Light acquisition and Dynamic Range

The integration time is the time during which the photons hitting the photodiodes are acquired. It corresponds to the time between the reset and the readout (Section 1.2.1.5). If the integration takes too long, an overflow phenomenon introduces additional electrons in neighboring of the overloaded pixel and can therefore distort their value.

The maximum amount of charges (full well capacity), before overflow, limits the intrinsic Dynamic Range of the pixel. This is the range of light intensities that the imager can capture, between the maximum achievable signal and the noise level in the dark. High Dynamic Range (HDR) can be achieved by improving this intrinsic pixel dynamic or by using post-processing (merge of several images with different integration times).

1.2.1.4 Color Filter Array and IR cut filter

A set of lenses and filters allows to focus the light and to select it by wavelength in order to obtain a mosaic image of the scene. The wavelength selections is made by color filters at the pixel level location. Most of the time, filtered colors correspond to green, red and blue. They are arranged into a regular pattern to form a mosaic called CFA. The CFA patterns are susceptible to introducing aliasing artifacts. The most widespread CFA is the Bayer Pattern [10]. It is composed of four pixels with twice as many green elements as red or blue. However, different CFAs have been proposed (Figure 1.5). The differences between the CFAs are based on different assumptions following different quality objectives [3]. Some objectives are focused on reducing the cost of image reconstruction or on the robustness of the network to reconstruction artifacts. For example, the assumption that the physical world can be oriented horizontally rather than vertically, or on the opposite assumption. The RGBW filters use a white pixel, sensitive to the full visible spectrum, to improve the luminance of the image [4][38].



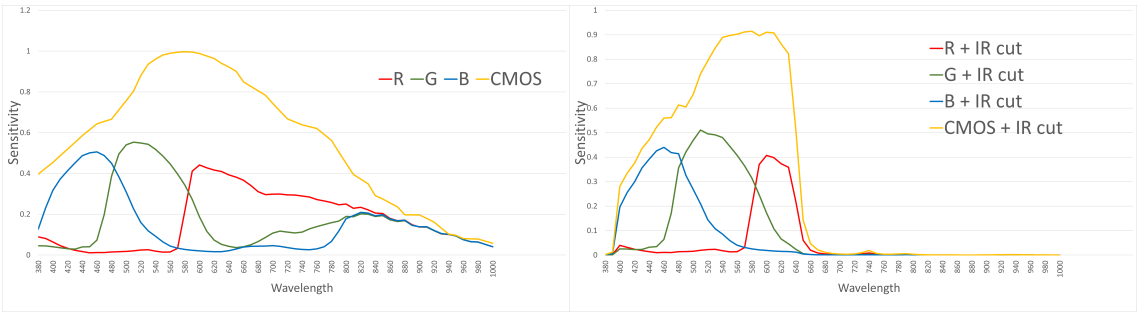
(a) (b) (c) (d) (e) (f)

Figure 1.5: Different CFA patterns: (a) Bayer pattern, (b) Lukac [3], (c) RGBW [4], (d) modified Bayer [3], (e) vertical strip [3], (f) pseudo-random [3].

Silicon is also sensitive to infrared wavelengths and the color filters are not effective to block IR photons (Figure 1.6). In order to avoid the pollution of pixels by infrared photons, an IR cut filter is used to reject the wavelengths above 700 nm. This filtering system is often positioned at the level of the camera lens in order to protect the entire pixel array from IR pollution.

1.2.1.5 Readout Circuit

Each pixel provides an electric representation of the absorbed light during a given integration time. This electrical signal, a voltage, must be read, then converted into a numerical value in order to record an image. The simplest idea to manage the readout step may be to route each pixel to read them in parallel, but this would require a lot of communication buses that would take up a lot of space in the matrix. Therefore, pixels are typically read row by row: only one output route per column. A control circuit selects the row to be read and the column outputs are read in parallel. The output signal of a pixel is therefore available on its column bus during



(a) (b)

Figure 1.6: Sensitivity responses for a given set of color filters and sensitivity response of a typical monochrome CMOS sensor. (a) RGB sensor without an IR cut filter. (b) RGB sensor with an IR cut filter.

its reading.

1.2.1.6 Light integration modality

There are two different techniques to integrate light: the rolling shutter technique and the global shutter technique.
The rolling shutter technique essentially means that adjacent rows of the array are exposed at slightly different times. The time shit between two consecutive rows depends on the readout duration. This first operation mode is represented by the top diagram in the Figure 1.7. Distortion artifacts can be present when some objects are moving at a rate that is fast compared to the reading time of a row.

On the contrary, the global shutter technique means that all pixels of the array are exposed simultaneously. Before the exposure begins, all pixels in the array are held in a reset state. At the start of the exposure, each pixel simultaneously begins to collect photons and is allowed to do so for the duration of the exposure time. At the end of exposure, the pixel value is stored until it is read (row after row). This method avoids distortion artifacts and motion blur but requires APSs to maintain their value in a memory. Global shutter pixels are thus larger, more complex and more costly than rolling shutter ones. The bottom diagram of the Figure 1.7 illustrates how the global shutter mode works. An example of the two shutter modes is given in the Figure 1.8, distortions due to the rolling shutter are clearly visible on the first image.

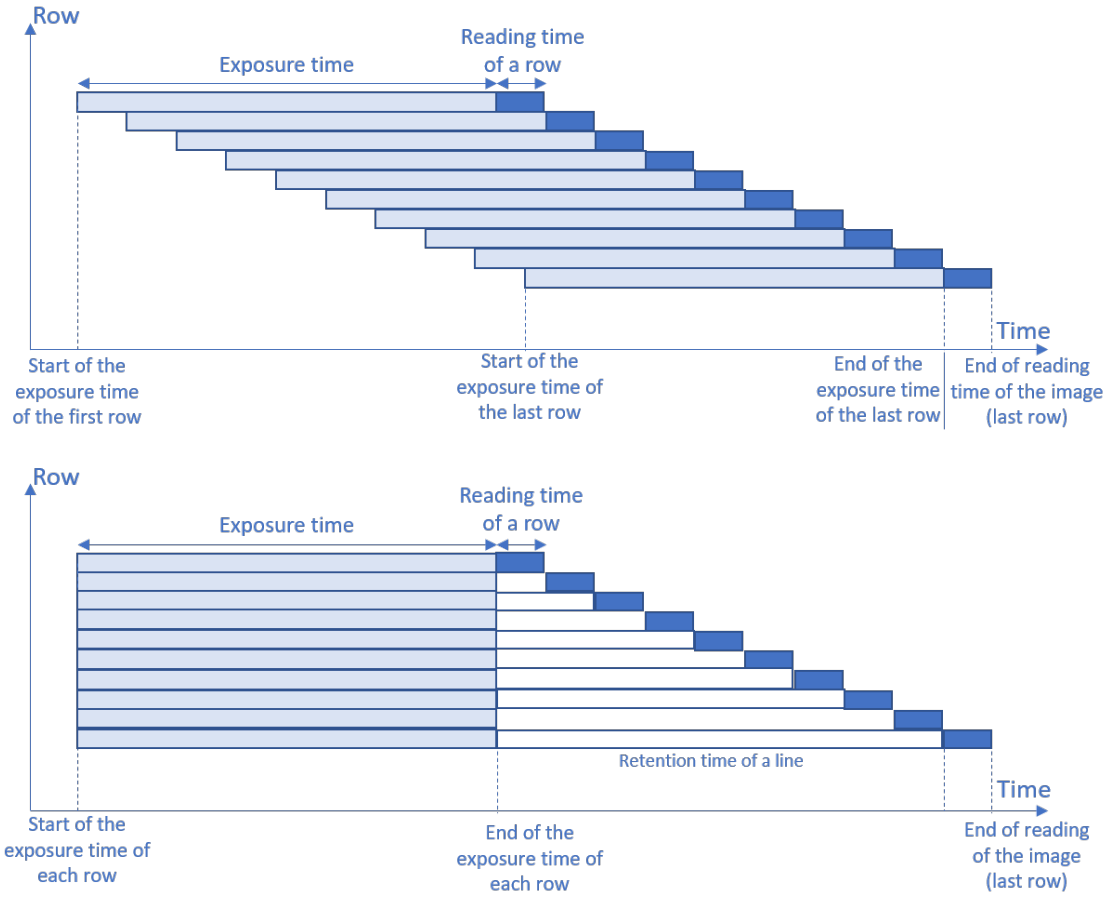
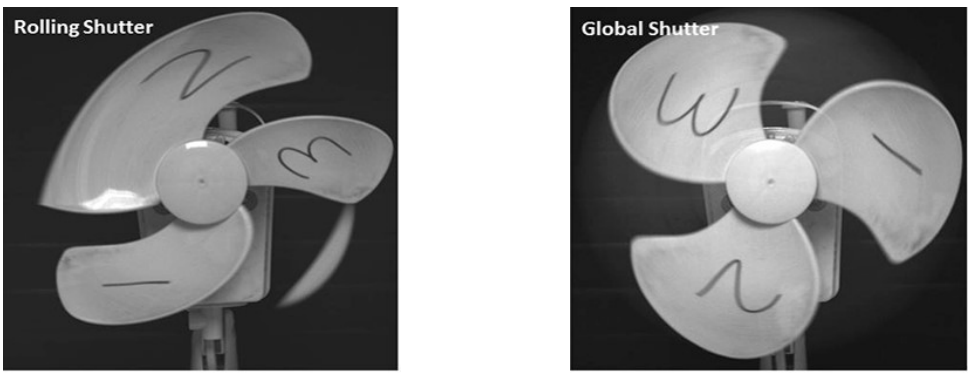


Figure 1.7: Diagram comparison of the exposure time and readout time between the rolling shutter mode (top) and the global shutter mode (bottom). The retention time is the delay during which the pixel maintains its value before before its readout starts.



(a) (b)

Figure 1.8: Example of reconstructed images acquire with a rolling shutter mode (a) and global shutter mode (b). On the first image, fan blades are distorted due to the rolling shutter effect. [source: https://andor.oxinst.com/]

1.2.2 ISP

During image acquisition, many alterations of the image can occur: distortions due to the optical system, sensor error, signal degradation, quantification error, transmission error, etc. An ISP is a digital signal processor used for image processing after CMOS imager readout. It contains all the different processing steps applied on raw sensor information in order to obtain a full color image [39]. The different steps are summarized in the Figure 1.9. The order of the steps may vary from one ISP to another, but the general idea remains the same.

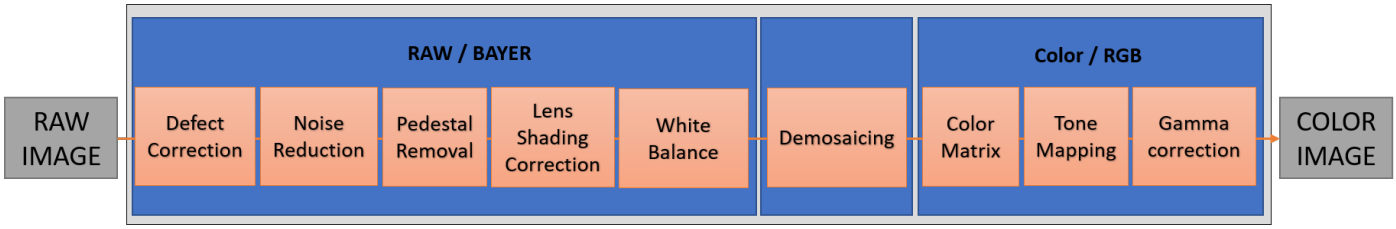


Figure 1.9: Typical ISP process scheme. The raw/Bayer block represents the processing applied to the mosaic image while the color/RGB block represents the processing applied to the reconstructed color image after the demosaicing step.

1.2.2.1 Defect Correction

The defect pixels are due to silicon impurities and manufacturing effects. They appear as white spots. They have to be corrected at the beginning of the ISP processing chain in order to avoid spreading of the error during the next filtering and reconstruction steps. Correction methods are usually based on a defect pixel map where the known defective pixels or indexes of the defectiveness are stored. The defect pixel map could be up-dated with a defect pixel detector. A soft or hard correction is then applied according to the defectiveness index of the corresponding pixel [40]. There are also methods that automatically detect the defect pixels.

1.2.2.2 Noise Reduction

Noise is characterized by pixels values which deviate visibly from the correct value. It appears as random speckles on a smooth area and can significantly degrade image quality. Impulsive noises, such as salt-and-pepper noise are examples of noise that can be removed by a median filtering [41][42]. There are many other methods of noise reduction. Most of spatial filters employed to remove noise remove also higher frequencies, which may blur the image. We can cite Gaussian filters. There is also the bilateral filter [43] which smooths images while preserving edges. The defect

correction and the noise reduction operations are similar and can sometimes be performed during the same processing step [44][45].

1.2.2.3 Offset Correction

During the acquisition phase, the photodiodes are sensitive to the dark current. It is an electric current that flows through photosensitive devices even when no photons are hitting the device. It depends on temperature and integration time, so it is a non constant offset over time and space. In order to quantify it, dark lines are used. They are pixel lines that are not exposed during the integration time, which are used as an estimation of the dark current. The dark current is therefore eliminated by subtracting the average value of the dark lines to the pixels values. However, to avoid clipping the negative values due to the previous subtraction, a pedestal is added. This is a fixed and defined offset to work in unsigned arithmetic [39].

Indeed, clipping the negative pixel values during dark current removal would cause the dark areas of the image to be brightened. During the phase of offset correction the pedestal is removed in order to map the black pixels to zero.

1.2.2.4 Lens Shading Correction

Lens shading (or lens vignetting) corresponds to the reduction in light falling on the photodiodes away from the center of the image. It is mainly caused by the geometry of the camera. Rays with a wide angle of incidence induce a greater vignetting effect. It can be simplified as the higher field of view of the camera, the larger the shading.A secondary cause of shading is the lens assembly that hinders the amount of light falling in the border pixels of the sensor. It usually depends on the aperture, and smaller apertures normally show lower shading. In general, this effect is asymmetrical and has unique characteristics for each camera due to the misalignment between the lens assembly and the sensor.

Different methods are used to compensate for vignetting. Traditional methods use a polynomial approximation of the vignetting function. This approximation is estimated a priori in a laboratory environment [46][47]. There are also more expensive methods that try to adapt the vignetting function from one sensor to another using an adaptive shading correction algorithm [48].

1.2.2.5 Auto White Balance

The human visual system is very good at judging what is white under different light sources, whereas raw color pixels do not compensate for the color temperature of the light source. Objects that appear white in the scene could be rendered not white in the image depending on the spectrum of the light source (Figure 1.10). For

example, a photo taken under the light of a candle is more yellowish than the same scene under day light (D65), which is more bluish.

Automatic white balance is a two-step operation, the first of which is to identify the main light source, and the second to automatically adapt the colors to the determined light. A color gain is calculated to compensate each color channel. Common white balance processing techniques are based on gray world theory [49], which says that most visual scenes in the world are, on average, gray. Other methods consider that the brightest pixel in the image is a white pixel [50]. White balance gains for each color channel are calculated based on these assumptions and then applied to each pixel.

(a) (b) (c)

Figure 1.10: Doll image [5] reconstructed with (b) and without (c) an Auto White Balance step compare to the Ground truth image (a) under a black body 3200 light source.

1.2.2.6 Demosaicing

Demosaicing is the process that consists in reconstructing a full color image from a CFA image (Figure 1.11). The objective is to recover for each pixel the full set of red, green and blue information from the available one. This subject has been widely explored over the last decades due to the difficulty of reconstructing information in high frequency areas of an image. It can induce artifacts such as zipper effect or aliasing. Demosaicing methods are explained in more details in the Chapter 2.3.

1.2.2.7 Color Matrix

The spectral sensitivity of a sensor depends on its pixel characteristics and color filters (Figure 1.6). Thus, for a given scene and given acquisition conditions, each sensor has a specific output in terms of color. To get accurate colors, which can be displayed on a standard monitor, the color image has to be transformed into a defined and known colorspace (Figure 1.12). The sRGB colorspace is used in most

(a) (b)

Figure 1.11: Crop of the Doll image [5](a) before the demosaicing step and (b) after the demosaicing step.

cases. To transform the values from sensor RGB to sRGB, a practical and cost-effective method is to apply a 3x3 Color Correction Matrix (CCM) [51] to the data. The CCM is approximated using the minimum mean squared color error between corrected test chart patches and the corresponding reference values. The reference image usually used is a 24-patch Colorchecker and the default color error parameter is the mean of ΔE 2000 [52], calculated for all patches.

(a) (b)

Figure 1.12: Doll image [5]. (a) sRGB image reconstructed with the color matrix step. (b) RGB image without the color matrix step (producing false colors).

1.2.2.8 Tone Mapping

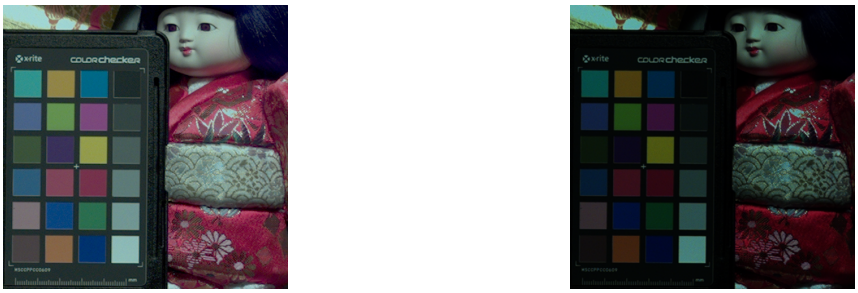
Tone mapping deals with reducing the tonal range within an image to make it suitable to be viewed on a digital screen. For example, it is necessary to apply a tone mapping algorithm on a 16-bit depth image to display it on a 8-bit screen while maintaining as much as possible the contrast. There are two types of tone mapping algorithms : global operators and local operators. Global tone mapping is simpler to implement but tends to lose details. It applies the same mapping scheme for all the pixels in the image [53]. Local tone mapping compresses the dynamic of each pixel according to its neighboring pixels preserving better the local contrast and details [54].

1.2.2.9 Gamma Correction

The final step of the ISP chain is the gamma correction which is a non-linear operation that codes the luminance of linear images. It is a power-law function that is applied to correctly display the data on a screen. For historical reasons and for optimization with respect to the human visual system, the use of bits when encoding an image takes advantage of the non-linear way in which humans perceive light and color [55]. The correction redistributes tones level closer to human visual perception: fewer bits are used to describe brighter values and more bits are used to describe darker values (Figure 1.13). Standard color spaces (such as sRGB) require that data are coded using a specific power-law function. The power-law expression of the gamma correction function is defined by:

$$V_{out} = V_{in}^{\gamma} \tag{1.2}$$

Where V_{out} and V_{in} represent respectively the output pixel value and the input pixel value. And γ is the gamma value. In the sRGB color space, gamma correction value is set to $1/2.2 = 0.45$.



(a) (b)

Figure 1.13: Doll image [5] reconstructed with (a) and without (b) the gamma correction step.

1.3 Range sensing

Range sensing is the name of a set of techniques that are used to produce a 2D image representing the distance of a scene from the sensor. Each value of the pixel array corresponds to the measured distance at this 2D location. There are several types of range cameras using different technologies to acquire the distance information. Some of them are active technologies, which means that they use their own light projector. Conversely, passive technologies do not use an active illumination.
Below are presented the technologies of stereo-vision, structured light and ToF.

1.3.1 Stereo-vision

Stereo-vision is a passive technology that use two cameras, arranged horizontally and separated by a baseline distance, to obtain two different views of a scene. It is similar to the human binocular vision. A matching step is necessary to identify the same features in both images. Depth information can then be computed from the disparity map using triangulation methods (Figure 1.14). The disparity is the difference between the x coordinate of two corresponding points [56]; it is typically encoded with a grey scale image. The depth is inversely proportional to the disparity. The two images need to have enough details and textures to find correlations between them. The stereo-vision is suitable for applications with a large field of view and for outdoor usage. Indeed, compared to other methods, it works well in sunlight, where the stereo images have a good signal to noise ratio.

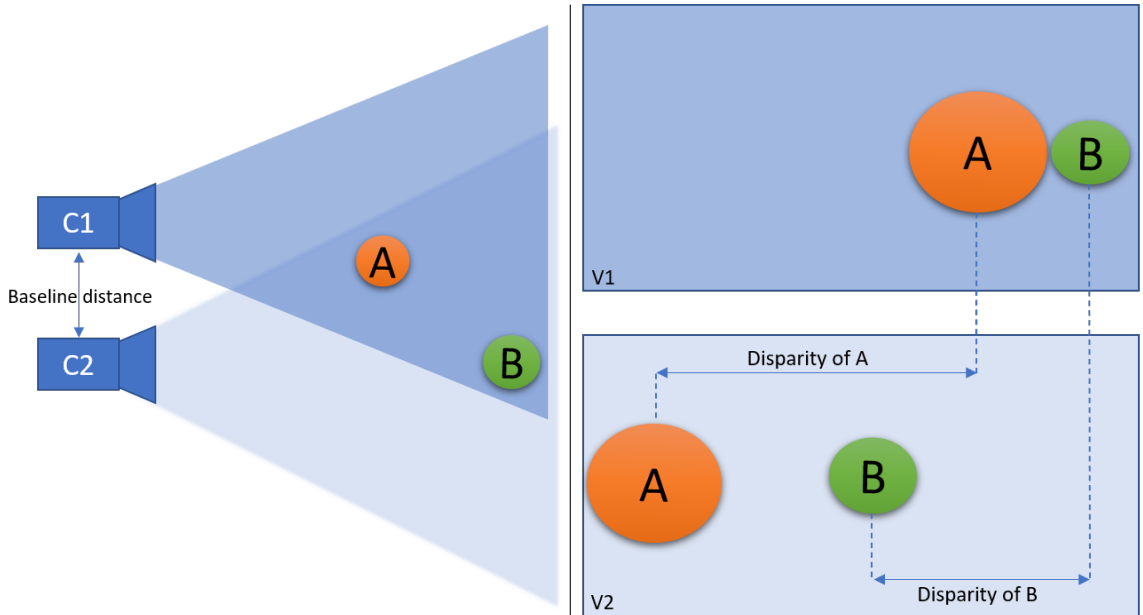


Figure 1.14: Principle of stereo-vision. C1 and C2 are the two cameras and V1 and V2 their corresponding view.

1.3.2 Structured Light

Structured light is a an active depth sensing solution based on the same approach as the stereo-vision, with the difference that one of the two cameras is replaced by a NIR projector. It is a diffractive beam splitter that projects a predefined pattern onto a scene or a surface. The reflected pattern is then recorded by the camera creating an image with depth-dependent deformations. The deformations compared to the projected known pattern are used to compute the depth map by optical

triangulation [57]. Compared to the classical stereo-vision, the matching problem is simplified thanks to the dot pattern that is projected on the scene. The structured light technology is convenient for indoor scene at medium distance because of the active illumination. But it loses its effectiveness in direct sunlight and in regions with a high interference of the same external light source technology used. It performs well on non-textured areas thanks to the projected pattern.

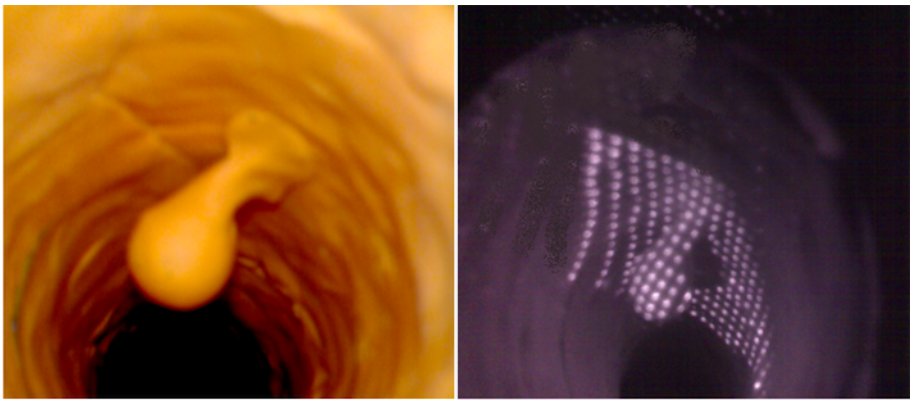


Figure 1.15: Examples of structured light application for polyp detection. The pattern is distorted because of the 3D structure of the polyp.

1.3.3 Time of Flight

ToF is an active solution to measure the distance between an object and a sensor. A signal is emitted from a light source, reflected on the target and then acquired by the sensor. The time difference between the emission and the acquisition of the signal is measured and used to compute the depth (Figure 1.16). A ToF system is composed by a NIR illumination source (850 nm or 940 nm) and a camera that acquires reflected photons for each pixel of the sensor array. Illuminator and sensor are properly synchronized to guarantee the accurate measurement of the time. It usually provides three measures at the same time: the depth map, the amplitude image and the offset. The amplitude image corresponds to the amount of returning active light signal. The offset is a function of the ambient light and the residual system offset. They are considered as strong indicators of the quality of measurements. The ToF systems can be categorized into direct-Time of Flight (d-ToF) and indirect-Time of Flight (i-ToF) sensors.

1.3.3.1 direct-Time of Flight

A d-ToF sensor [58] computes distances directly from the measured time that it takes for photons to travel between two points, from the sensor's emitter to a target and then back to the sensor's receiver. The d-ToF technology uses short pulses of

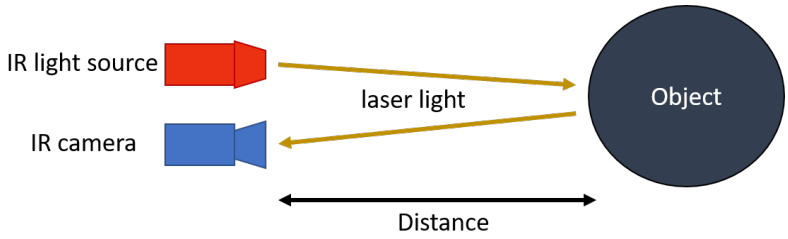


Figure 1.16: Principle of Time of Flight imaging.

light, and the time until each pulse returns to the sensor is measured. The depth is then directly computed with:

$$d = c\frac{t}{2} \tag{1.3}$$

Where d represents the distance, c is the speed of light and t is the measured time by the sensor.

Direct-Time of Flight technologies are usually made with Single-photon avalanche diode (SPAD) technology [59]. However, it is difficult to shrink the pixel size under $8\,\mu m^2$ [60]. In the context of a mixed RGB-Z matrix, the presence of SPAD has disadvantages, particularly because of their size compared to a color pixel size. The i-ToF technology is therefore privileged in the context of a RGB-Z sensor [61].

1.3.3.2 indirect-Time of Flight

A i-ToF sensor emits a continuous and modulated light. The depth is calculated according to the phase difference between the emitted light and the reflected light from an object [62]. Each pixel is typically composed of two or four [63] shutters (also called taps) synchronized with the light source modulation frequency in order to measure the amount of light reflected four times for every period (M_1, M_2, M_3 and M_4). It allows a measurement of the phase. The four shutters are out-of-phase and they have a 90-degree phase delay from each other. This technique is known as "four bucket sampling" [64] and requires a special type of photodiode (fast photodiode or pinned photodiode) [65], because of the high frequencies used in practice (Figure 1.17). The phase difference is computed using:

$$\varphi = \arctan\left(\frac{M_3 - M_4}{M_1 - M_2}\right) \tag{1.4}$$

the amplitude :

$$A = \frac{\sqrt{[M_1 - M_2]^2 + [M_3 - M_4]^2}}{2} \tag{1.5}$$

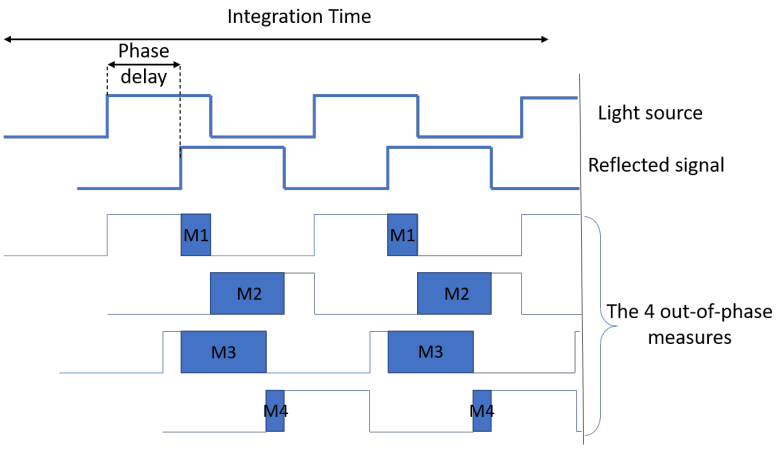


Figure 1.17: Principle of Indirect-Time of Flight imaging. Phase delay is computed using the four shutters measures.

the offset :

$$A = \frac{M_1 + M_2 + M_3 + M_4}{4} \tag{1.6}$$

Where M_1, M_2, M_3 and M_4 represent the four accumulated charges during each sampling. The distance is then calculated using the phase difference with :

$$d = \varphi \frac{c}{4\pi f} \tag{1.7}$$

Where c is the speed of light and f the modulation frequency of the light source. The maximum range d_{max} that can be measured without ambiguity for a given modulation frequency is defined by :

$$d_{max} = \frac{c}{2f} \tag{1.8}$$

Beyond the d_{max}, the measured distance d is ambiguous. It is due to the periodicity of the modulated signal. When a distance to an object is superior to d_{max} (i.e. the phase difference is superior to 360°), the object appears to be closer than the real distance (modulo the d_{max}). This phenomenon is called phase wrapping. One solution to this problem is to lower the modulation frequency. It allows increasing the unambiguous range measured (1.8) but the measurement uncertainty also increases proportionally to the unambiguous range [66]. Another solution is to use the multiple modulation frequencies method [67][68], also called *disambiguity*. Several series of phase measurements are performed for the same scene. The light source is modulated at different frequencies for each measure. The actual distance is determined by comparing the multiple unwrapped phases modulo 2π (Figure 1.18). This method requires taking several captures of the same scene with different light source modulation frequencies. That involves a motion sensitivity.

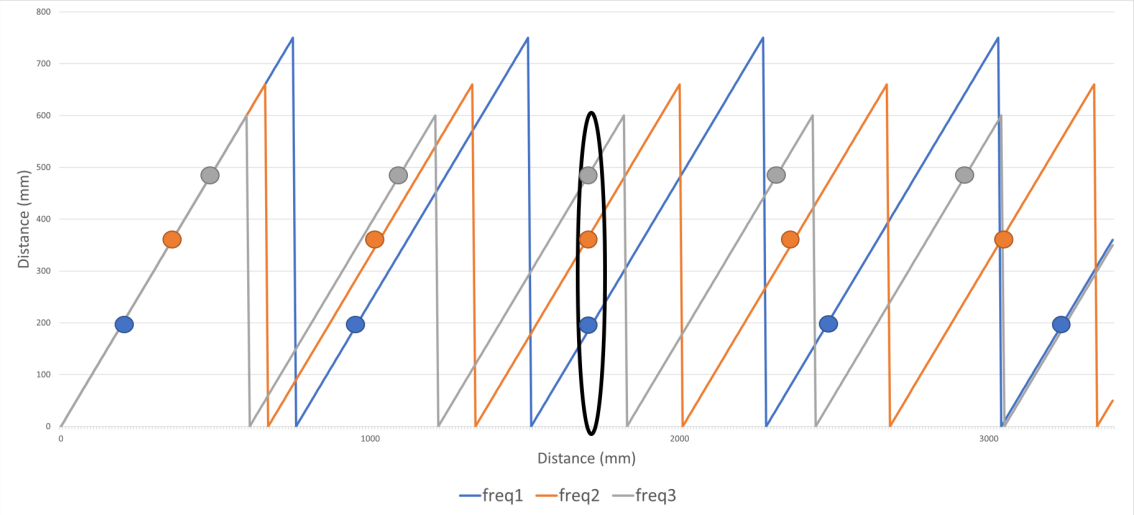


Figure 1.18: Example of multi-frequency phase unwrapping with a set of 3 frequencies (freq1= 200 MHz ; freq2= 225 MHz ; freq3= 250 MHz). It allows identifying target up to 6 m. A wrapped phase is measured (denoted by the colored dots) for each frequency. The distance is then estimated where the unwrapped phases are equal (denoted by the black circle).

1.4 RGB-Z Systems

RGB-Z systems are sensing devices that simultaneously provide a color image and a depth map characterizing the distance of objects seen in the image. Popularized by Microsoft with the first Kinect released in 2010 [69], RGB-Z systems are now provided by different companies. There are two categories of RGB-Z devices: multi-sensors systems and monolithic systems.

1.4.1 Multi-sensors Systems

Multi-sensors systems are the most common RGB-Z systems. They are composed of two separate sensors, one for the color image acquisition and one for the depth map acquisition. They use the same kind of color and depth sensor technologies that were presented in the previous sections (sections 1.2 and 1.3). Consumer systems are mainly based on Structured light (paragraph 1.3.2) and i-ToF (paragraph 1.3.3) technologies to acquire the depth map. During the last decade, a lot of research studies were made in order to improve applications in which a high accuracy is needed. One of the main research topic is the calibration [70]. Hach and Steurer [71], proposed a RGB-Z camera where both sensors share the same optical system, resulting in an inherent parallax-free representation of the RGB and Z images. Their system uses a mirrors system to alternately reflect the light into the RGB sensor or

the ToF sensor.

1.4.2 Monolithic Systems

A RGB-Z monolithic system captures both color and depth information on the same chip. Several acquisition methods of both color and depth images can be considered. However, these systems are much more recent, and are still in prototype state. A first RGB-Z sensor is presented, where all the pixels are used to alternatively acquire color and depth images. Then, a second solution, using an array composed of two types of pixels, is presented. In this example, both images are acquired simultaneously.

1.4.2.1 Alternate Color and Depth Acquisitions

Seong-Jin et al. [6] propose a first 640×480 single VSoC RGB-Z sensor. In their matrix, all the pixels are used to acquire both color and depth information. Their solution has no dedicated pixel, they all have the same structure. Images are alternatively captured by switching the operation mode. They demonstrate that a pinned-photodiode [65] can demodulate an i-ToF signal where each photodiode is split into two regions to realize high speed charge transfer. One Z-pixel corresponds to a 4-shared pixels cluster. The figure 1.19 shows the conceptual diagram of a time multiplexing architecture for obtaining alternatively color and depth images.

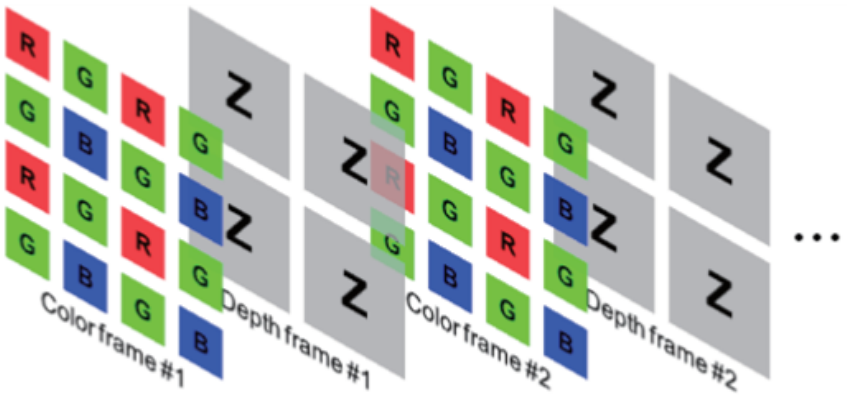


Figure 1.19: Conceptual diagram of a time-division multiplexing architecture [6].

In [72], Kim et al. present a 2^{nd} generation of a RGB-Z sensor based on the same time-division capturing architecture as [6]. The pixel array resolution has been increased to 1920×1080 by reducing the color pixel size (from 6 µm [6] to 3.65 µm [72]). However, in this case, one Z-pixel corresponds to a 16-shared pixels cluster. A mechanical system allows to switch between the IR pass filter and the visible pass filter [61]. It limits the frame rate in the depth-and-color acquisition mode because

the switching delay is around 200 ms to change the filter. This mechanical system is not desirable for a commercial product due to the low frame rate.

1.4.2.2 Simultaneous Acquisitions

Kim et al. introduce the first RGB-Z sensor [7] that simultaneously captures both color (RGB) and depth (Z) information. It is a 1920×1080 sensor composed of two types of pixels individually optimized for color acquisition by the first family of pixels and for depth measurement by the second family of pixels. The pixel array is divided into two interlaced arrays with a first 1920×720 array of color pixels (pitch of 2.25 µm) and a second 480×360 array of range pixels (range pixel is 2.25 µm × 9 µm^2). There is a row of Z-pixels every three rows of color pixels (Figure 1.20). The paper is focused on the implementation and performances of relatively small Z-pixels (2.25 µm × 9 µm^2 for [7] compared to 12 µm × 12 µm^2 for [6]).

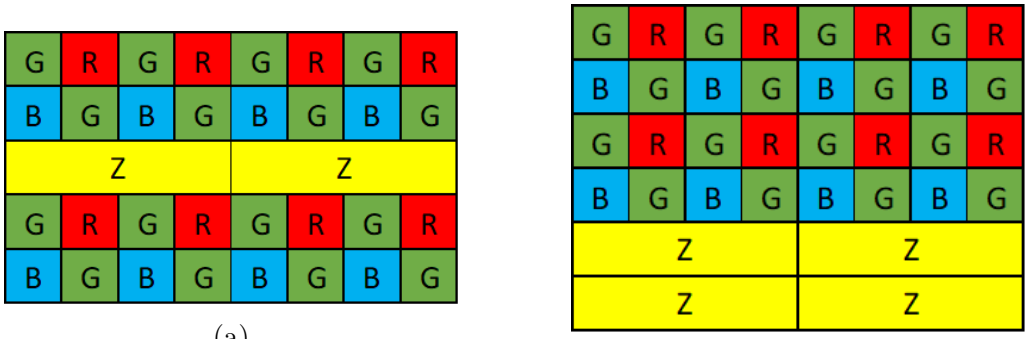


Figure 1.20: Representation of (a) the theoretical RGB-Z matrix introduced in [7] and (b) the RGB-Z matrix introduced in [8].

However, at the publication date, CFA and micro-lenses have not yet been applied to the RGB-Z pixels. A major issue is the absence of IR filtering. Consequently color pixels receive photons from both visible and IR spectrum. Simulations of NIR "fogging" suppression carried out by the ISP have been made and seem to be a possible solution. However no test with real data has been carried out. Concerning the visible light contamination in the Z pixels, it is suppressed using a commercially available visible-light-blocking filter. The details of this filter are not provided. The last issue concerns the missing information due to the heterogeneity of the pixels matrix. Color information is missing at the Z-pixel location and need to be reconstructed in order to obtain a full CFA image (a Bayer image in most cases).

In [8], Shi et al. propose a missing pixel reconstruction method followed by a demosaicing to reconstruct a full color image from an RGB-Z pixel matrix (chapter 2.4.2). The pixel matrix is similar to the one proposed in [7] but in this case, there is an alternation of four rows of RGB pixel and two rows of Z pixels (Figure 1.20).

The reconstruction method of the missing information has been patented [73]. It is explained in the next chapter.

	Seong-Jin et al. [6]	Kim et al. [72]	Kim et al. [7]
Die size	6 mm × 6 mm	9.7 mm × 6.2 mm	-
Emitter	LED / 850 nm wavelength	LED / 850 nm wavelength	LED / 850 nm wavelength
Modulation frequency	10 MHz	20 MHz	20 MHz
Fill Factor	34.5%	38.5%	48% (depth pixel)
Color pixel size	6 µm^2	3.65 µm^2	2.25 µm^2
Color resolution	640 × 480	1920 × 1080	1920 × 720
CFA	Bayer	Bayer	Bayer based
Depth pixel size	12 µm^2	14.6 µm^2	2.25 µm × 9 µm
Depth resolution	320 × 240	480 × 270	480 × 360
Measurement range	1 m-3 m	0.75 m-4.5 m	1 m-7 m
Depth accuracy	62 mm@3 m	38 mm@4.5 m	5 mm@1 m

Table 1.1: Characteristics of monolithic RGB-Z sensors.

1.5 Problematic

We have seen in the previous sections that the implementation of a monolithic RGB-Z system presents new constraints at different technical levels [7]. We summarize them in order to highlight the thesis objectives. The constraints are divided into two categories:

- Intrinsic issues at the sensor level. As it is a heterogeneous system where two different types of pixel share the same pixels matrix the color filtering system must be adapted. Most of the depth acquisition systems operate with an IR illumination. The color pixels should not be exposed to NIR light and the range pixels should not be exposed to visible light. In this context, a new filtering system must be designed to apply a local NIR filtering on color pixels and a local filtering of the visible light on depth pixels. The layout and the readout circuit should also be adapted. The architecture of the matrix depends on the pixel distribution and the difference between the color and depth pixel pitch sizes.

- Extrinsic issues at the image processing level. New types of defects are induced by the technological limits of the sensor. Due to the matrix of mixed pixels, color information is missing at the Z-pixel location and vice versa. This leads to resolution problems and incomplete information.

The objectives of this thesis are mainly focused on the extrinsic issues. The first objective is to design a processing chain to optimize the image quality of a new RGB-Z sensor. The idea is to study reconstruction solutions allowing to obtain a reconstructed color image quality similar to an image quality acquired with a classical RGB sensor. A second objective is to analyze the impacts of the sensor architecture on the processing chain. There is a strong inter-dependency between the performance of a sensor and the algorithms. For example, the size of the Z-pixel relative to the color pixel and the pattern used for the CDFA have an impact on the performances and on nature of the interpolation solutions used. An additional goal is to study how our solution could be implemented in a digital integrated circuit. Indeed, it is important to study the possibilities and constraints of integrating such an implementation in a real-time image processing chain.

Now that we have explained what a RGB-Z sensor is and what its constraints are, we detail in the next chapter the different methods used to reconstruct the missing color information.

Chapter 2

State of the Art

2.1 Introduction

This work focuses on image reconstruction methods for monolithic RGB-Z sensors. However, few works have been published on mixed RGB-Z matrix, so we will consider methods for RGB demosaicing too. Indeed, in both cases it is a reconstruction problem of missing information in a matrix formed by a mosaic of pixels. It is also necessary to introduce metrics to evaluate the quality of the reconstructed images and thus be able to classify the different reconstruction methods. This chapter will first present the metrics applied to the reconstructed color images to evaluate the reconstruction methods. In a second step, we present demosaicing algorithms used to reconstruct a color image from a RGB CFA image. They are varied both in terms of complexity and performance. Finally, we present the few existing reconstruction methods used for RGB-Z mixed matrices.

2.2 Metrics

In image processing, it is important to evaluate the impact of algorithms applied to an image. For this, we can make visual analysis of images or use IQA methods. Among them, there are the so-called Full Reference methods. It means that a complete reference image is used to evaluate the reconstructed one. We can differentiate two types of methods:

- methods using only numerical criteria. They are based on an error calculation between a reference pixel and a reconstructed pixel.
- methods based on the hypothesis that the HSV[1] is highly adapted for extracting structural information. In this case, we do not compare individual pixels, but local patterns of pixels.

[1] Human Visual System

In order to evaluate the quality of the reconstructed images for the different algorithms, several metrics are used. They are presented in the following sections.

2.2.1 PSNR and C-PSNR

The PSNR is one of the most commonly used objective image quality metrics. It is calculated based on the Mean Squared Error (MSE).

$$MSE = \frac{1}{NM}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(I_d(m,n) - I_r(m,n))^2 \tag{2.1}$$

Where I_d and I_r represent respectively the pixel intensity of the reconstructed image and the reference image. M and N are the dimension of the given image. The PSNR is then computed by 2.2.

$$PSNR = 10log_{10}\frac{s^2}{MSE} \tag{2.2}$$

Where s represents the maximum range of the image, defined as $s = 2^b - 1$, where b is the bit depth of the image. For a 8-bit image, $s = 255$. The C-PSNR is also defined for the color images. It is computed using C-MSE 2.3 and 2.4, with k indexing to the 3 color channels.

$$C - MSE = \frac{1}{3NM}\sum_{k=1}^{3}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(I_d(m,n,k) - I_r(m,n,k))^2 \tag{2.3}$$

$$C - PSNR = 10log_{10}\frac{s^2}{C - MSE} \tag{2.4}$$

The C-PSNR is used on the final full color image while we will use the PSNR to directly evaluate our intermediate reconstruction of a CFA image.

It is not the most efficient method to discriminate structural deformations and it does not correlate well with the measure of perceived quality. Its major advantage is the simplicity of calculation.

2.2.2 SSIM and M-SSIM

The SSIM [74] is considered to be correlated with the quality perception of the Human Visual System (HVS). The idea is to measure the similarity of structure between two images, rather than a pixel-to-pixel difference as PSNR does. It is based on the assumption that the HVS is more sensitive to changes in image structure. It compares local patterns of pixel intensities that have been normalized for luminance

and contrast. It is designed by three factors that describe the distortion between the two images. The three factors describe:

– the luminance comparison using the mean luminance (2.6),

– the contrast comparison using the standard deviation (2.7),

– the structure comparison using the covariance between both images (2.8).

The SSIM metric is calculated over several windows between two images (windows from distorded image are noted (d) and from reference image are noted (r)). The calculation for a given window is given by equation 2.5.

$$SSIM(d,r) = \left[l(d,r)^{\alpha} \times c(d,r)^{\beta} \times s(d,r)^{\gamma}\right] \tag{2.5}$$

With,

$$l(d,r) = \frac{2\mu_d\mu_r + c_1}{\mu_d^2 + \mu_r^2 + c_1} \tag{2.6}$$

$$c(d,r) = \frac{2\sigma_d\sigma_r + c_2}{\sigma_d^2 + \sigma_r^2 + c_2} \tag{2.7}$$

$$s(d,r) = \frac{\sigma_{dr} + c_3}{\sigma_d + \sigma_r + c_3} \tag{2.8}$$

Where, μ_d and μ_r are respectively the average of d and r, σ_d and σ_r are respectively the variance of d and r, σ_{dr} is the covariance of d and r. The three variables c_1, c_2 and c_3 are used to stabilize the division, avoiding weak denominators. They are defined by $c_1 = (0.01 \times L)^2$, $c_2 = (0.03 \times L)^2$ and $c_3 = c_2/2$, with L representing the maximum range of the image. It is defined by $L = 2^b - 1$, where b is the bit depth of the image. Conventionally, to simplify the expression, we set $\alpha = \beta = \gamma = 1$. To evaluate the overall image quality, we compute the mean of each local window SSIM index.

M-SSIM [9] is an improvement of the SSIM. It includes in its computation the impact of the differences in structures of different scales. The diagram in the Figure 2.1 illustrates the operation of M-SSIM. The system iteratively applies a low-pass filter and downsamples the filtered image by a factor of 2. The M-SSIM is computed using 2.9.

$$M - SSIM(d,r) = [l_M(d,r)]^{\alpha_M} \times \prod_{j=1}^{M} [c_j(d,r)]^{\beta_j} \times [s_j(d,r)^{\gamma_j}] \tag{2.9}$$

In [9], the number of iterations is fixed to 5 and the parameters are determined to be equal to $\beta_1 = \gamma_1 = 0.0448$, $\beta_2 = \gamma_2 = 0.2856$, $\beta_3 = \gamma_3 = 0.3001$, $\beta_4 = \gamma_4 = 0.2363$,

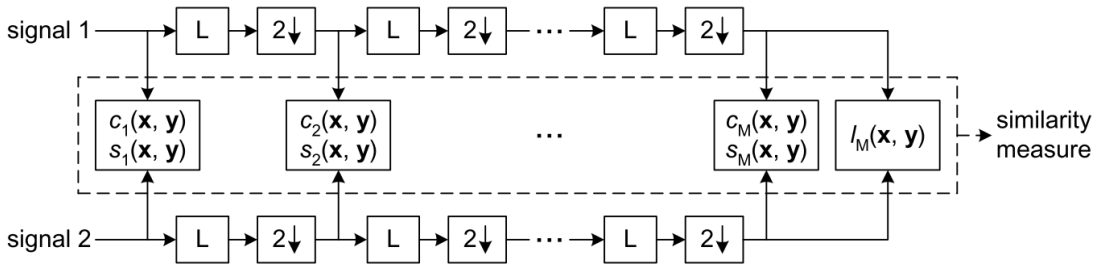


Figure 2.1: Multi-Scale SSIM measurement system [9]. L: low-pass filtering; 2 ↓: downsampling by factor 2.

and $\alpha_5 = \beta_5 = \gamma_5 = 0.1333$.
Both SSIM and M-SSIM indices are decimal value between 0 and 1. An index equal to 1 means that the both image are identical and therefore indicates perfect structural similarity.

2.2.3 Zipper metric

The zipper effect is an artifact that corresponds to unwanted repetitive changes in intensity between neighboring pixels along an edge (Section 2.3.1). A reference-based zipper metric is described in [75]. The intensity differences between a pixel and its eight neighbors are computed in the reference image, and the neighbor with the smallest difference is identified. We then compute the difference between the two pixels at the same location in the reconstructed images. We finally study the variation between these two differences, in order to determine if there is a noticeable change between them. If it is confirmed, the pixel is considered to have a zipper effect. Details steps are described below.

First, to compute this metric, the CIELAB color space [76], specified by the CIE[2]. It is intended as a perceptually uniform space, where a given numerical change corresponds to similar perceived change in color. The Zipper effect corresponds to an abrupt change in intensity, so we use the CIELAB color space in this case because it can better capture the color difference perceived by human observers. It is therefore necessary to convert the sRGB images into the CIELAB color space. This conversion is performed in 3 steps:

– Conversion of the image from sRGB to linear RGB by applying an inverse gamma.

[2]stands for International Commission on Illumination. It is the international authority on light, illumination, color, and color spaces.

– Conversion of the RGB image to XYZ image. XYZ is another color space specified by the CIE. Details on this transform are given in [77].

– Conversion of the XYZ image to CIELAB image. Details on this transform are given in [77].

Once the images are converted into the CIELAB color space, the zipper metric is computed. The difference between two pixels in the CIELAB color space is defined as the Euclidean distance between color values of the pixels, noted ΔE^*_{ab} 2.10.

$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \tag{2.10}$$

Where, (L^*_1, a^*_1, b^*_1) and (L^*_2, a^*_2, b^*_2) are the CIELAB color values of the two pixels. The pixel I corresponding to the smallest difference between the eight neighbors of a given pixel is given by 2.11.

$$I = argmin_{i \in \epsilon} \Delta E^*_{ab}(P, i) \tag{2.11}$$

The variation between the two differences computed with the same two pixels in the reference image, and the reconstructed image is given by 2.12.

$$\psi = \Delta \tilde{E}^*_{ab}(P, I) - \Delta E^*_{ab}(P, I) \tag{2.12}$$

$\Delta \tilde{E}^*_{ab}(P, I)$ corresponds to ΔE^*_{ab} between the two considered pixels in the reconstructed image. A pixel is considered to have a noticeable change in color difference when $\mid \psi \mid > \delta$. The threshold δ is set to 2.3 according to [78]. The final score of the zipper metric corresponds to the percentage of pixels in the image for which a noticeable change is noticed.

2.3 RGB Demosaicing techniques

In 1976 Bruce Bayer [10], of the Kodak company, proposed a new system of acquisition of digital color images where each pixel represents one color information (CFA image). He proposed a matrix where for each 2×2 set of pixels, two diagonally opposed pixels have green filters, and the other two have red and blue filters (Figure 2.2). It forms a mosaic image, this is why algorithms used to reconstruct full color image (i.e. three color components per pixel) are called demosaicing algorithms. The demosaicing methods can be sorted in various ways. One way is to divide them into adaptive and non-adaptive interpolation methods. Non-adaptive methods interpolate according to a fixed filter for each pixel, while adaptive methods interpolate the missing information according to the local properties of the image.

Another way is to divide them into spatial based algorithms and frequency based algorithms. The spatial methods exploit assumptions about either spatial or spectral correlations between colors of neighboring pixels. The frequency method is based on Fourier analysis of the CFA image.

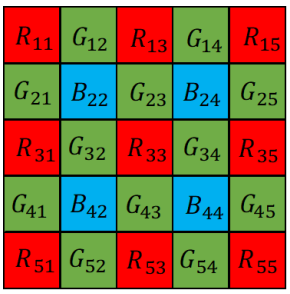


Figure 2.2: CFA matrix proposed by Bayer [10] with pixel numbering.

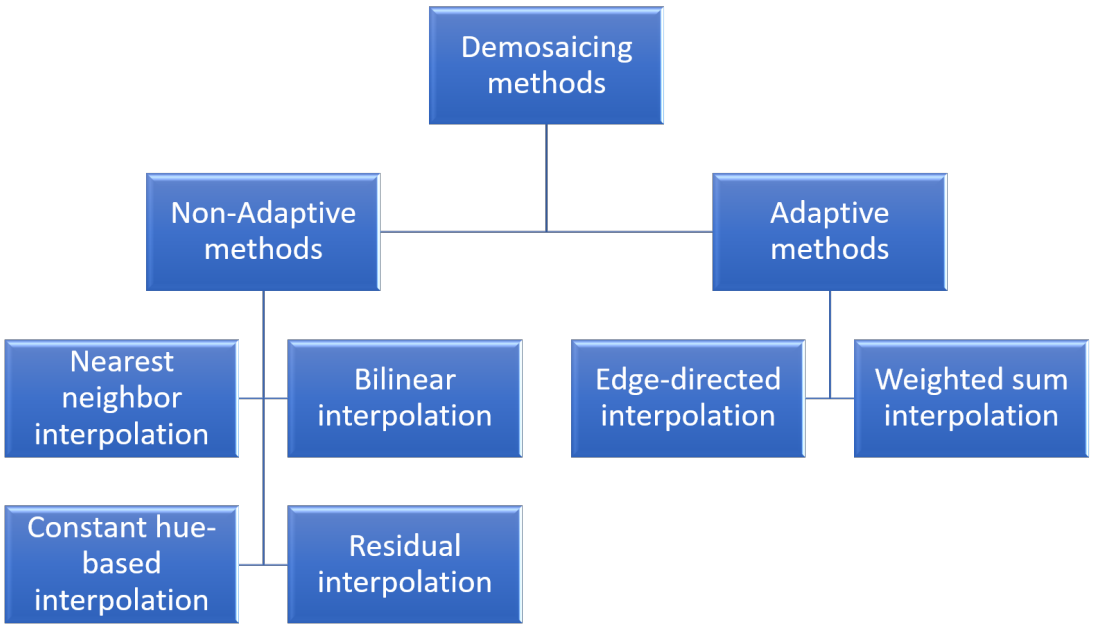


Figure 2.3: Representation of demosaicing methods into adaptive and non-adaptive methods.

In the following sections, we start by presenting demosaicing artifacts. It is important to understand the reconstruction errors in order to better assess the reconstruction methods presented below. After that, non-adaptive and adaptive methods are introduced. They are all spatial based reconstruction but one, which is a frequential adaptive method, presented in the section 2.3.4. This demosaicing method is reused in the Methodology part (section 3.3.1.3). The Figure 2.3 summarizes the presented methods.

2.3.1 Artifacts analysis

There is no perfect demosaicing method. Some artifacts appear when a method fails to correctly reconstruct the content of an image. The common artifacts due to the demosaicing are:

- Zipper effect: abrupt or unnatural changes of color differences between neighboring pixels, manifesting as an "on-off" pattern (Figure 2.4).
- Labyrinth effect: structures are created in non-structured areas. This artifacts usually appears using directional demosaicing methods: the algorithms detect edges where there are none because of noise (Figure 2.4).
- Block artifact: creation of bloc structures (Figure 2.4).
- Aliasing: this is a phenomenon that occurs when the signal is undersampled. This induces false color artifacts or jagged edges[3]. It appears along the diagonal edges or in areas with repetitive structures. (Figure 2.4).
- False color: when a color is wrongly reconstructed (Figure 2.5).
- Blurring effect: structures appear less sharp.

2.3.2 Non-adaptive spatial interpolations

Non-adaptive methods include nearest neighbor, bilinear and bicubic interpolations. They perform an interpolation according to a fixed weight pattern for each pixel. Even though they are computationally efficient, they introduce large errors in edge areas that degrade the resulting image; in fact, they do not take into account the presence of object boundaries. The advantage of non-adaptive algorithms is that they are easy to implement.

2.3.2.1 Nearest neighbor interpolation

The nearest neighbor interpolation, also called the "box-car" interpolation is described by Dillon and Bayer [79]. This method is an interpolation by pixel copy. A known pixel value is held and copied to the location of the unknown neighboring pixel during scan lines. This has the effect of simply making a pixel bigger in each color plan. The nearest neighbor method creates wrong edges and a lot of color artifacts.

[3]example images are from https://medium.com/@rishabhgupta05/aliasing-in-images-73258df1dbd2

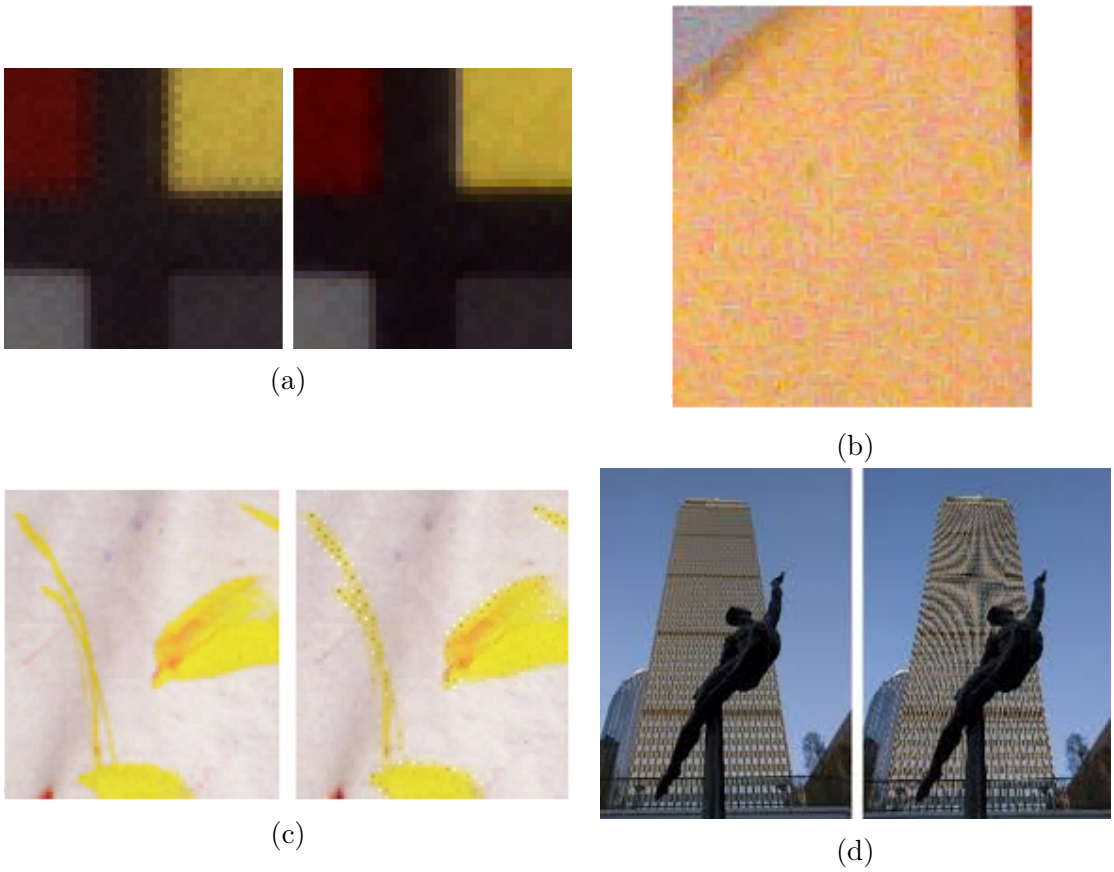


Figure 2.4: Example of common artifacts (a) Zipper effect. (b) Labyrinth effect. (c) Bloc artifacts. (d) Spurious spatial frequencies due to aliasing.

2.3.2.2 Bilinear interpolation

The bilinear interpolation consists in linearly interpolating the missing pixels from the existing horizontal and vertical neighboring pixels. For example, to interpolate the blue pixel at the R_{33} location and the green pixel at the same position, the following formulas are applied (refer to Figure 2.2):

$$B_{33} = \frac{B_{22} + B_{24} + B_{42} + B_{44}}{4} \tag{2.13a}$$

$$G_{33} = \frac{G_{23} + G_{34} + G_{43} + G_{32}}{4} \tag{2.13b}$$

It generates a blurred image, but is less prone to the false colors artifacts than the nearest neighbor method (Figure 2.5.b).

2.3.2.3 Constant-hue-based interpolation

In [12], D.Cok proposed to interpolate the hue, rather than the color information. He explained that an object of constant color has a constant color ratio even though lighting variations may change the measured values. The hue of an image does not

abruptly change between neighboring pixel locations. The hue is defined as the ratio between Red and Green ($\frac{R}{G}$) and Blue and Green ($\frac{B}{G}$). Constant-hue-based algorithms firstly interpolate the green channel using any method (bilinear or more sophisticated methods). Then they compute color hue for known red and blue pixels using the green plan. The red hue and blue hue are interpolated in a bilinear way and then multiplied by the G value to determine the missing red and blue values. For example, the equation 2.14 is used to interpolate the blue pixel at the R_{33} location. Instead of interpolating the color ratios [16], the color differences can also be interpolated [80]. An example of constant-hue based interpolation is shown in Figure 2.5.d.

$$B_{33} = G_{33} \frac{\frac{B_{22}}{G_{22}} + \frac{B_{24}}{G_{24}} + \frac{B_{42}}{G_{42}} + \frac{B_{44}}{G_{44}}}{4} \tag{2.14}$$

This is now a commonly used assumption in demosaicing algorithms and methods for reconstructing missing information. Most of the proposed algorithms no longer use independently the color channels, but exploit the relationships between the channels.

Malvar et al. [22] proposed an improvement of the bilinear interpolation that use constant-hue based concepts. This is a linear demosaicing filter based on bilinear interpolation, where a filter is predefined for each pixel position. For the interpolation of a green value at a red pixel location, the red information is not discarded; it is compared to its estimate for a bilinear interpolation. For example, to interpolate the green pixel at the R_{33} location, the following formulas are applied (refer to Figure 2.2):

$$G_{33} = \hat{G_{33}} + \alpha \Delta_R \tag{2.15}$$

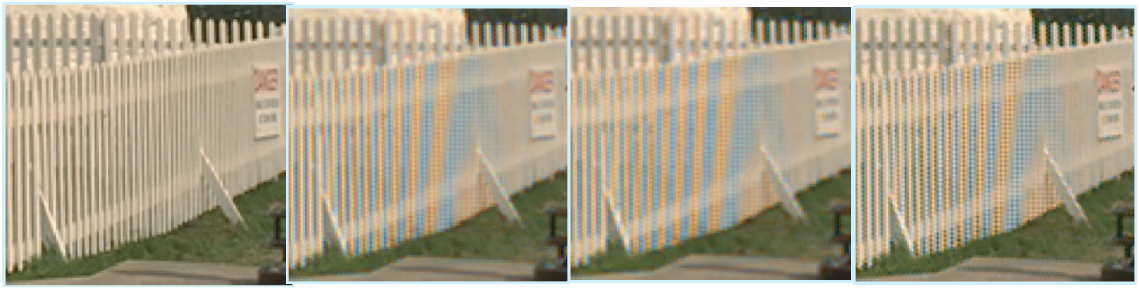
Where $\hat{G_{33}}$ is computed using 2.13b, α is predefined gain parameter and Δ_R is the gradient of the red pixel at that location, computed by

$$\Delta_R = R_{33} - \frac{1}{4}(R_{31} + R_{13} + R_{53} + R_{35}) \tag{2.16}$$

The constant-hue based methods are more efficient than the two previous ones, however, they still suffers from color artifacts because the assumption of constant hue may not hold around edges.

2.3.2.4 Residual Interpolation

Residual interpolation is an alternative to the color difference interpolation or constant-hue-based methods (section 2.3.2.3). The Figure 2.6 shows the differences between a color difference based algorithm and the residual interpolation method. Instead



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

Figure 2.5: Results of cropped images for the "lighthouse" from Kodak database [11] reconstructed using non-adaptive methods. (a) Reference image. (b) Bilinear interpolation. (c) Bicubic interpolation. (d) Constant-hue-based interpolation. [12]. Bilinear interpolation and bicubic interpolation show aliasing and false color artifacts. The constant-hue-based interpolation shows better performances but there are still color artifacts. Images are from [13].

of comparing color differences directly, the residual is a difference between an observed and a tentatively estimated pixel value. In [14], the green channel is firstly interpolated. The tentative estimate of the red image ($\tilde{R}$) is then generated by using the green image as a guided filter [81]. Residual are calculated between the observed pixels and the tentatively estimated R pixels ($R-\tilde{R}$). After that, residuals are interpolated and the red image is finally reconstructed by adding the tentative estimated image ($\tilde{R}$).

The motivation to use the residual differences rather than the color differences is based on the assumption that the residuals should be flatter than the standard color differences. It should be noted that the algorithms used for the interpolation steps during the residual interpolation could be various algorithms such as a bilinear interpolation or an edge directed interpolation.

2.3.3 Adaptive spatial interpolations

Adaptive methods study the local properties of the image to adapt the interpolation to each pixel. Indicators could be based on different features such as the local gradient [82] or the local homogeneity [83]. The detected features are mainly edges, corners and texture variation.

2.3.3.1 Edge-directed interpolation (EDI)

EDI allows the reconstruction of missing information in order to preserve the continuity of structures in an image. It analyzes the area around a given pixel to determine if a preferred interpolation direction exists [84]. The interpolation is then performed according to this direction. The objective is to avoid interpolating across

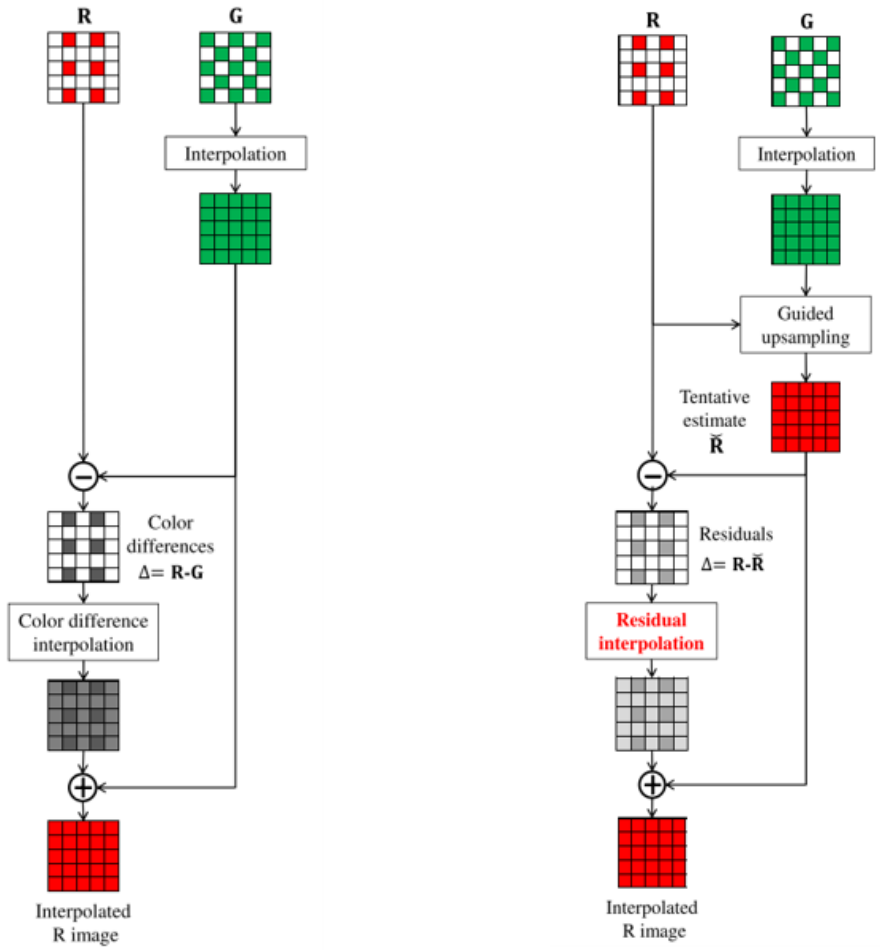


Figure 2.6: Red channel interpolation using color difference interpolation (a) and residual interpolation (b) [14].

edges. In [82], horizontal (ΔH) and vertical (ΔV) gradients are computed using the color channel of the missing pixel. The gradients are then compared to a constant threshold (λ). If the gradient in one direction falls below the threshold, interpolation is performed only along this direction. If both gradients are below or above the threshold, the area is considered as a flat area and a bilinear interpolation 2.13b 2.13a is performed. Example to reconstruct G_{33} (refer to Figure 2.2):

Gradient computation:

$$DeltaH = |G_{32} - G_{34}| \tag{2.17a}$$

$$DeltaV = |G_{23} - G_{43}| \tag{2.17b}$$

if $\Delta H > \lambda$ and $\Delta V < \lambda$:

$$G_{33} = (G_{32} + G_{34})/2 \tag{2.18}$$

else if $\Delta V > \lambda$ and $\Delta H < \lambda$:

$$G_{33} = (G_{23} + G_{43})/2 \tag{2.19}$$

else:

$$G_{33} = (G_{32} + G_{34} + G_{23} + G_{43})/4 \tag{2.20}$$

Gradients can be computed using larger regions and by exploiting different color channel. C.Laroche and M.Prescott [15] use a 5x5 window centered on the missing pixel to computed the gradients. J.Adams [85] proposes to use other estimators for pixel classification and interpolation. In this implementation, he computes a gradient on green pixels and Laplacian on red or blue pixels. The Laplacian corrects the averaging interpolation. Example to reconstruct G_{33} (refer to Figure 2.2):

Gradient computation:

$$\Delta H = |G_{32} - G_{34}| + |2 \times R_{33} - R_{31} - R_{35}| \tag{2.21}$$

$$\Delta V = |G_{23} - G_{43}| + |2 \times R_{33} - R_{13} - R_{53}| \tag{2.22}$$

if $\Delta H > \Delta V$:

$$G_{33} = (G_{23} + G_{43})/2 + (2 \times R_{33} - R_{13} - R_{53})/4 \tag{2.23}$$

else if $\Delta V > \Delta H$:

$$G_{33} = (G_{32} + G_{34})/2 + (2 \times R_{33} - R_{31} - R_{35})/4 \tag{2.24}$$

else:

$$\begin{aligned} G_{33} =& (G_{23} + G_{34} + G_{32} + G_{43}) \\ & /4 + (4 \times R_{33} - R_{13} - R_{35} - R_{53} - R_{31})/8 \end{aligned} \tag{2.25}$$

2.3.3.2 Adaptive Weighted average

The edge-directed methods may have a computational time that varies for each pixel. Since the algorithms are based on conditions, the final interpolation operation is not always the same. It depends on the current pixel and its neighbors. Furthermore, there is a separation between the classification part which compute the gradients and the interpolation part which reconstruct the missing pixel. To avoid these effects, it has been proposed to build a weighted factor directly from the gradients. The missing pixel intensity is then defined as a weighted sum of its neighbors. The weight of each pixel corresponds to an edge estimator, the interpolation automati-

cally adapts to the edges of the image. Such an algorithm was proposed by Kimmel in [16]. Example to reconstruct G_{12} (refer to Figure 2.2):

$$G_{33} = \frac{W_{23}G_{23} + W_{34}G_{34} + W_{43}G_{43} + W_{32}G_{32}}{W_{23} + W_{34} + W_{43} + W_{32}} \tag{2.26}$$

Where W is a weighted factor built depending on the horizontal , vertical and diagonal gradients. To interpolate a pixel P at position (i, j), the formula of equation 2.26 is used, with:

$$W_{i+k,j+l} = \frac{1}{\sqrt{1 + (Grad(P_{i,j})^2 + (Grad(P_{i+k,j+l})^2}} \tag{2.27}$$

with $k \in [-1, 0, 1]$ and $l \in [-1, 0, 1]$, and $Grad$ represents the horizontal, vertical and diagonal gradients:

$$Grad_H(P_{i,j}) = \frac{P_{i,j-1} - P_{i,j+1}}{2} \tag{2.28a}$$

$$Grad_V(P_{i,j}) = \frac{P_{i-1,j} - P_{i+1,j}}{2} \tag{2.28b}$$

$$Grad_{D1}(P_{i,j}) = \frac{P_{i-1,j+1} - P_{i+1,j-1}}{2\sqrt{2}} \tag{2.28c}$$

$$Grad_{D2}(P_{i,j}) = \frac{P_{i-1,j-1} - P_{i+1,j+1}}{2\sqrt{2}} \tag{2.28d}$$

In [86], Alain Horé and Djemel Ziou introduced a weighted average demosaicing algorithm that can be used for various CFA. Their method is directly inspired from [87]. They improved it with an edge-detection model that detects edges and their direction before performing interpolation along them.

2.3.4 Frequency domain: Wavelet based approach

A method based on the wavelet transforms is proposed by L.Alacoque [18] inspired by [88]. This method is validated by STMicroelectronics and it is called Mozart in the rest of this work. It is a frequency based method. In this approach, a simple Haar wavelet transform (Figure 2.8) is applied on the image, which yield the average (S), vertical (V), horizontal (H) and diagonal (D) information of the image. For each pixel, four coefficients are computed using equations 2.29.

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

Figure 2.7: Result images for the "lighthouse" from Kodak database [11] reconstructed using adaptive methods. (a) Reference image. (b) Edge-directed interpolation based on the gradient [15]. (c) Edge-directed interpolation based on the gradient corrected by the Laplacian [15]. (d) Weighted average interpolation [16]. Adaptive methods present less artefacts than non-adaptive methods (Figure 2.5). Images are from [17].

$$S = \frac{1}{4}\begin{bmatrix} +1 & +1 \\ +1 & +1 \end{bmatrix}, V = \frac{1}{4}\begin{bmatrix} +1 & -1 \\ +1 & -1 \end{bmatrix}, H = \frac{1}{4}\begin{bmatrix} +1 & +1 \\ -1 & -1 \end{bmatrix} \text{ et } D = \frac{1}{4}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}$$

Figure 2.8: the four basic functions of the Haar transform used in [18].

$$S = \frac{1}{4}(g + r + b + G) \tag{2.29a}$$

$$V = \frac{1}{4}(g - r + b - G) \tag{2.29b}$$

$$H = \frac{1}{4}(g + r - b - G) \tag{2.29c}$$

$$D = \frac{1}{4}(g - r - b + G) \tag{2.29d}$$

S represents the local average which is similar to a luminance of the form $Lum = \frac{1}{4}(r + 2g + b)$. Assuming that others coefficients (V, H and D) represent the mixture of pure luminance transition information ΔV, ΔH or ΔD and pure chrominance

information Ca or Cb, we can rewrite the coefficients:

$$V = \Delta V + \frac{1}{4}(b - r) = \Delta V + Ca \tag{2.30a}$$

$$H = \Delta H - \frac{1}{4}(b - r) = \Delta H - Ca \tag{2.30b}$$

$$D = \Delta D + \frac{1}{4}(2g - r - b) = \Delta D + Cb \tag{2.30c}$$

Where $Ca = \frac{1}{4}(b - r)$ and $Cb = \frac{1}{4}(2g - r - b)$. An adapted filtering of these coefficients makes it possible to separate the color and luminance information. The inverse Haar transform thus makes it possible to obtain a luminance information at full resolution to which the color information is added to reconstruct the RGB image at full resolution:

$$G = Lum + Cb \tag{2.31a}$$

$$R = \frac{1}{2}(Lum - Cb) - Ca \tag{2.31b}$$

$$B = \frac{1}{2}(Lum - Cb) + Ca \tag{2.31c}$$

An advantage of this method is that it works well on noisy data. This is understandable, as the image is low pass filtered during the algorithm. The computation of the coefficient S corresponds to a low-pass filtering. However, the images may appear a little blurred. Despite this, the edges are correctly reconstructed.

2.3.5 Other methods

There are also other methods that we just mention in this manuscript. Some algorithms reconstruct the missing information by choosing for each pixel the interpolation with the fewest artifacts. Concretely, the decision between a horizontal and vertical interpolation is taken a posteriori, by analyzing the directionally interpolated images [89]. Iterative methods are also presented in [90][91] where reconstruction procedures are repeated until a stopping criterion is achieved. An other interesting approach is the non-local means methods [92] adapted from denoising methods. The aim of this algorithm is to use non-local information from the image to infer missing color when the local geometry cannot be inferred from the neighboring pixels.

Convolutional Neural Network (CNN) are currently the state-of-the-art solution for a wide range of image processing tasks. In [93], Tan et al. propose a method for image demosaicing based on deep convolutional neural networks. The proposed

method divides the demosaicing tasks into an initial demosaicing step and a refinement step. The initial step produces a demosaiced image containing unwanted color artifacts. The refinement step then reduces these color artifacts using a deep residual estimation and multimodel fusion producing a higher quality image.

These methods are often complex and can induce an uncertain computing time, which is not desirable in the context of ISP for embedded sensors.

2.3.6 Comparison of the presented methods

The performances of the different demosaicing algorithms presented are shown in Table 1. These results are computed on the Kodak database [11]. The original images of the database are sampled according to the Bayer pattern, then reconstructed with the presented methods. The results of the first six methods are taken from [17].

Methods	PSNR
Nearest neighbor interpolation	26.767
Bilinear interpolation	30.301
Constant-hue based interpolation [12]	32.918
EDI [82]	33.312
EDI + Laplacian [85]	37.325
Weighted average interpolation [16]	37.385
Wavelet based interpolation [18]	33.963

Table 2.1: Performances of presented demosaicing algorithms in terms of PSNR.

2.4 Mixed matrix reconstruction

A mixed matrix is composed of color pixels (visible wavelength) on the one hand, and other pixels (non-visible wavelength) on the other hand. Those pixels could be range pixels as in a RGB-Z sensor or IR pixels as in a RGB-IR sensor.

2.4.1 RGB-IR matrix

RGB-IR sensors acquire both RGB and NIR images. They are composed of a mosaic of RGB and NIR pixels. This is a first matrix where a part of the information acquired by some pixels does not correspond to the visible spectrum. The information acquired by the IR pixels is similar to the information acquired by the color pixels. The difference is that the photons acquired by IR pixels belong to the spectrum of infrared (more than 800 nm). Several RGB-IR filter arrays exist (Figure 2.9). The density and position of IR pixels vary from one architecture to another. Depending

on the available color and NIR information, the reconstruction methods could vary. However, they often use a guide image to reconstruct the full color image.

(a) (b) (c)

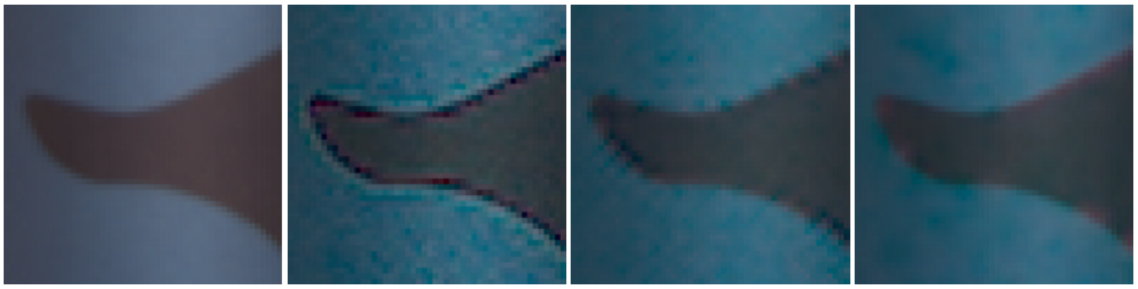
Figure 2.9: Examples of RGB-IR filter arrays where N represents a NIR pixel: (a) 2x2 RGB-IR filter array [19], (b) 4x4 NIR filter array [20], (c) sparse-NIR filter array [20].

In [19], they choose to use the NIR information as guidance because they found that NIR image can be captured with less noise in a low-light scene with a NIR flash [94]. In [21], two RGB-IR filter arrays are introduced: the 4x4-NIR filter array (Figure 2.9.b) and the sparse-NIR filter array (Figure 2.9.c). The same demosaicing method [20] is applied to both patterns, which have a high sampling density of the green band. Authors firstly reconstruct the green channel using residual interpolation [95]. For the sparse-NIR filter array, the green plan is interpolated using both red and blue channels while, the three red, blue and IR channels are used for the 4x4-NIR filter array. The green channel is then used as a guide to reconstruct the red and blue channels.

Moreover, most of RGB-IR sensors have no dedicated IR filter for the color pixels. The issue mentioned in the section 1.4.2.2 about the filter system is still present. IR light is not blocked and is collected in addition to the visible light: color pixels are polluted. However, the IR pixels provide a good measure of the IR signal and this is subtracted from the color pixels [96]. Examples of reconstructed images provide in [21] are shown in Figure 2.10. The 4x4 NIR filter array and the sparse-NIR filter array present less artifact than the 2x2 NIR filter array. This is particularly noticeable along the edges for the sparse-NIR filter array.

2.4.2 RGB-Z matrix

Regarding the RGB-Z sensors introduced in the section 1.4.2.2, L.Shi et al. [8] [73] propose a method of reconstructing a full color image from a RGB-Z sensor. To my knowledge, this is the only state of the art publication concerning the reconstruction of missing pixels in the context of a mixed RGB-Z matrix as defined in section

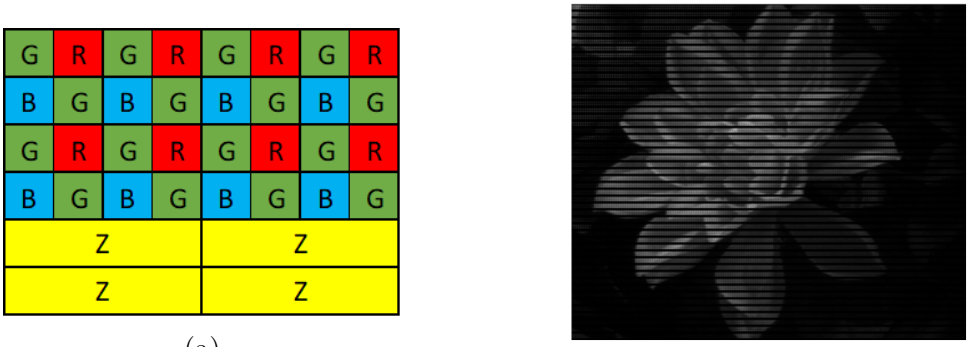


(a) (b) (c) (d)

Figure 2.10: Results of the reconstruction method applied to a simulated RGB-IR image proposed in [21]. (a) Ground truth. (b) 2x2 uniform RGB-IR filter array [19]. (c) 4x4 NIR filter array [20]. (d) Sparse-NIR filter array [20].

1.4.2.2. The reconstruction difficulties differ from the RGB-IR case: the pixel array architecture is unique and the Z pixel information is different from a IR pixel information.

The heterogeneous matrix is an alternation of four lines of colored pixels, which makes it possible to establish a basic Bayer pattern, and then two complete lines of Z pixels (Figure 2.11). A consequence of this matrix architecture is the absence of color pixel in the horizontal neighborhood of the pixels Z. This problem is comparable to a deinterlacing problem where there are whole lines of color pixels missing. In their method, L.Shi et al. [8] does not use the information from the Z-pixels to reconstruct the missing color information. This choice is neither obvious nor argued.



(a) (b)

Figure 2.11: (a) The layout of the RGB-Z color-depth sensor study in[8].(b) An exampple of raw RGB-Z image, where Z-pixel are set to 0 and correspond to the black rows [8].

The reconstruction process is divided into two parts. The first part consists of reconstructing the missing color information in order to obtain a Bayer matrix. Then, in a second part, a demosaicing step will be applied in order to reconstruct the complete color image.

The missing pixel interpolation method is an EDI algorithm based on the directional correlation between the pixels in the adjacent original scan lines. For a given

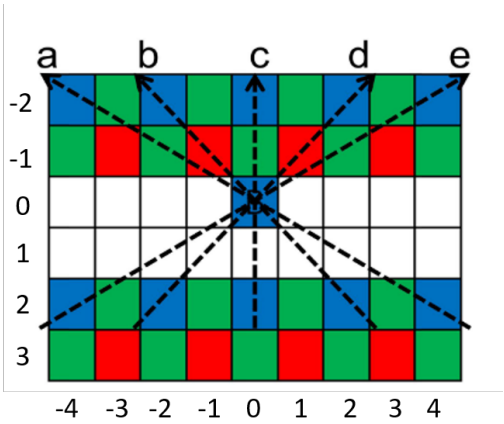


Figure 2.12: Example of a directional interpolation of a missing blue pixel based on a 9x6 pixels kernel [8].

missing color pixel, five edge directions are evaluated. An example is presented on the Figure 2.12. It is a blue pixel reconstruction where the five edge directions (a, b, c, d and e) are computed as follow:

$$a = |I(y-2, x-4) - I(y+2, x+4)| \tag{2.32a}$$
$$b = |I(y-2, x-2) - I(y+2, x+2)| \tag{2.32b}$$
$$c = |I(y-2, x) - I(y+2, x)| \tag{2.32c}$$
$$d = |I(y-2, x+2) - I(y+2, x-2)| \tag{2.32d}$$
$$e = |I(y-2, x+4) - I(y+2, x-4)| \tag{2.32e}$$

Where $I(y, x)$ represents the pixel value at the location (y, x). The five differences are then evaluated. If the region contain a dominant edge, a median filter is performed in the direction of the highest correlation. Otherwise, for a relatively homogeneous area, a bilinear interpolation along vertical region is performed. A dominant edge is defined as the smallest difference regarding a threshold T and the pairs of opposite differences:

If $min(a, b, c, d, e) = a$ & $|a - d| > T$ & $|a - e| > T$,
Then, $I(y, x) = median(I(y-2, x), [I(y-2, x-4) + I(y+2, x+4)]/2, I(y+2, x))$

If $min(a, b, c, d, e) = b$ & $|b - d| > T$ & $|b - e| > T$,
Then, $I(y, x) = median(I(y-2, x), [I(y-2, x-2) + I(y+2, x+2)]/2, I(y+2, x))$

If $min(a, b, c, d, e) = d$ & $|d - a| > T$ & $|d - b| > T$,
Then, $I(y, x) = median(I(y-2, x), [I(y-2, x+2) + I(y+2, x-2)]/2, I(y+2, x))$

If $min(a, b, c, d, e) = e$ & $|e - a| > T$ & $|e - b| > T$,
Then, $I(y, x) = median(I(y - 2, x), [I(y - 2, x + 4) + I(y + 2, x - 4)]/2, I(y + 2, x))$

Otherwise, $I(y, x) = [I(y - 2, x) + I(y + 2, x)]/2$

Once the Bayer image is fully reconstructed, the demosaicing step can be applied. This is an adaptive interpolation solution mainly inspired from [97]. The weights used in the interpolation are computed following the local pixel similarities and their position in the kernel. Compared to the solution proposed by R. Ramanath and W.E.Snyder [97], L.Shi et al. [8] add a colour-selective kernel that specifies neighbouring pixels according to the reconstructed pixel. This allows to only use known pixels during the demosaicing step, rather than the newly reconstructed pixels in step one. Reconstruction results are shown in the Figure 2.13. They show the interest of a dedicated algorithm to interpolate the missing color pixels. There are noticeable structural and color artifacts present in the image reconstructed using a bilinear interpolation (Figure 2.13.d) compared to the image reconstructed using the proposed EDI algorithm (Figure 2.13.e). Unfortunately, in their article, L.Shi et al. They do not provide any comparative points of view using known metrics only propose a visual analysis of their result.

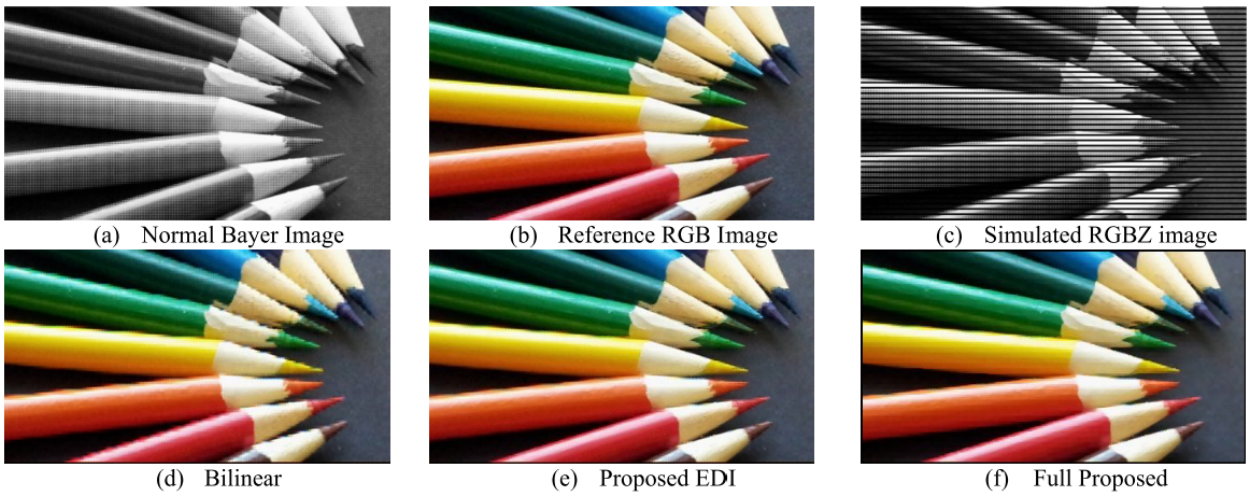


Figure 2.13: Results of the reconstruction method applied to a simulated RGBZ image proposed in [8]. (a) Normal Bayer image. (b) Reference RGB image corresponding to (a), demosaiced by Malv ar et al. [22]. (c) Simulated RGBZ image by blacking out rows in (a). (d) Reconstructed image by filling the missing rows with bilinear interpolation (demosaiced by Malvar et al. [22]). (e) Reconstructed image using EDI method [8] (demosaiced by Malvar et al. [22]). (f) reconstructed image using EDI method [8] and color adaptive demosaicing [8].

2.5 Conclusion

In this chapter we have first seen the different metrics used to quantify the performances of image reconstruction algorithms. The literature concerning mixed RGB-Z matrices and a fortiori the methods used to reconstruct the missing information in such a matrix has few documented examples. This is why, we have mainly studied demosaicing methods: they also correspond to methods for reconstructing missing information. We have also seen a first solution to reconstruct the missing information for a given RGB-Z matrix. With this example, we have seen that a good reconstruction solution is closely related to the architecture of the RGB-Z matrix.

Now that we have studied solutions for reconstructing the missing information, we will first explore some architectures of RGB-Z matrices. Then we will propose algorithmic solutions to reconstruct the missing information in these matrices.

Chapter 3

Proposed patterns and designed algorithms

3.1 Introduction

In the context of RGB-Z sensor where color images are displayed, it is important to have as few color artifacts as possible while having a sufficient depth map resolution. Shi et al.[8] proposed a reconstruction solution dedicated to a RGB-Z matrix [7] where entire lines of colored pixels are missing. A narrow horizontal edge perfectly aligned with this missing pixels lines would be impossible to detect and thus to reconstruct. More generally, such system generates fixed horizontal structures, which are easily annoying for the human eye. One of the goals of this thesis is to propose solutions to reconstruct a color image from a mixed RGB-Z sensor that has a quality comparable to an image acquired with a standard RGB sensor. We also intend to guide and limit the scope of solutions for the development of a RGB-Z sensor. To do this, new RGB-Z architectures must be proposed, with dedicated reconstruction algorithms. The reflection conducted for the implementation of solutions dedicated to specific RGB-Z architectures is illustrated in the Figure 3.1. In a first step, RGB-Z architectures have been proposed according to the constraints related to Z-pixel size. Depending on the nature of these matrices (based on the Bayer pattern or not) a study of the impacts of the demosaicing step is conducted. Finally, solutions to reconstruct the missing information are proposed and studied for each chosen matrix.

The plan of this chapter follows the same approach: in a first part, solutions of RGB-Z architectures are proposed, then in a second part solutions of reconstruction of the missing information are described.

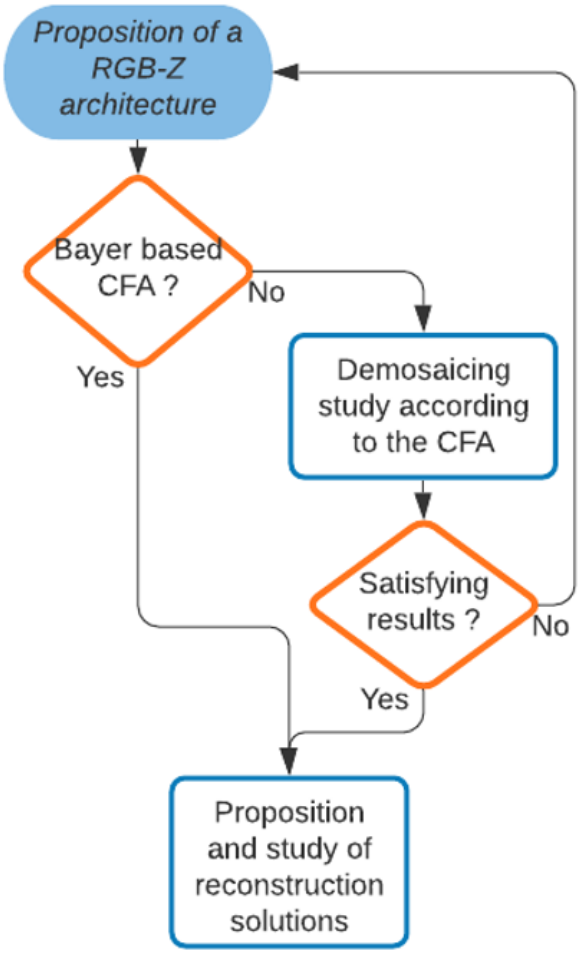


Figure 3.1: Thought process flowchart.

3.2 RGB-Z matrices study

We have seen in the previous chapter that solutions for reconstructing missing information are intimately linked to the structure of the pixel arrays. For example, L.Shi et al. [8] is inspired by reconstruction methods used in full size video frame reconstruction : de-interlacing. This choice is motivated by their RGB-Z matrix architecture which is based on different types of pixel lines (RGB pixels lines and Z pixels lines). In our case, the design of RGB-Z matrices is constrained by the Z-pixel size. Indeed, two new solutions of Z pixels have been proposed internally to design the CDFA.

3.2.1 Constraints

In the context where the quality of the color image is one of the first performance criteria of a RGB-Z sensor, the main constraints concern the missing color reconstruction at the Z pixels location. The two main aspects that impact the color reconstruction are therefore the Z-pixel size and its sampling within the matrix.

3.2.1.1 Z-pixel size

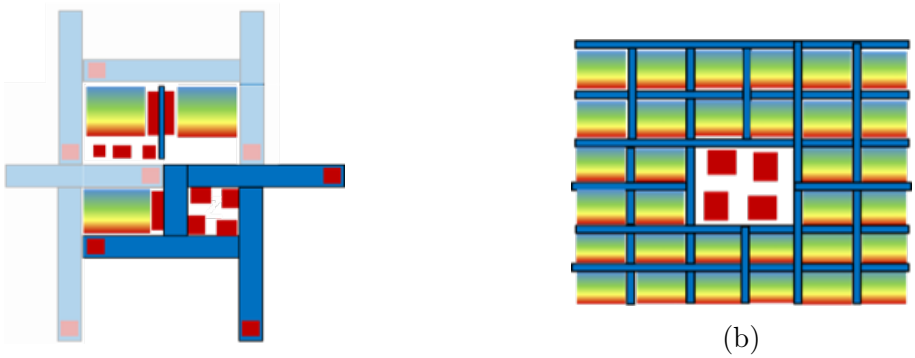
The larger a Z-pixel is compared to a color pixel, the more color information is missing at it location. It is therefore important to have a Z-pixel that is as small as possible to facilitate the reconstruction of missing color pixels. Moreover, if its size is a multiple of the color pixel size, it allows to have a regular unit pattern RGB-Z on the whole matrix. Two different Z-pixel architectures have been proposed internally for our RGB-Z matrix. These two designs are based on the works presented in [98] [99].

The first one has a dimension of $3.2\,\mu m^2$. It is designed to be integrated in a RGB-Z matrix where color pixels have the same dimension ($3.2\,\mu m^2$). In this case, pixels are bordered with CDTI memories of the depth pixels that implies a FF reduction of the color pixels. We describe this pixel as the *1x1-Z-pixel* (Figure 3.2.a).

The second one has the same pitch as the previous one ($3.2\,\mu m$). However, it is designed to be integrated in a RGB-Z matrix where color pixels are half the size of the Z-pixels ($1.6\,\mu m$ in pitch). In this case, a 3D-stacked technology (1.2.1.1) is necessary to have a vertical integration of the memory elements of the Z-pixel. This also improves the FF of the color pixel. We describe this second pixel as the *2x2-Z-pixel* (Figure 3.2.b).

3.2.1.2 Z-pixel sampling

A second constraining aspect is the sampling of the Z-pixel across the matrix. Indeed, the more Z-pixels there are, the less color information is available. However, it is necessary to ensure that the Z resolution is sufficient. To answer this question without real data, we made an assumption based on the recognized *iPhone X TrueDepth* system [100]. It is a well known face recognition system based on a Structured Light technology where 30,000 dots are projected on the target. This implies that a maximum of 30,000 points are recorded to compute the depth map. Our assumption is that a minimum of 30,000 pixels is needed to have a suitable resolution for the Z-pixel distribution. Considering a sensor of 3 Megapixels, a sampling of one Z pixel per hundred color pixels is necessary to have, at least, 30 000 measurement points. However, it can be risky to have a too sparse distribution of the Z pixels on



(a) (b)

Figure 3.2: Proposed Z-pixels. The Z-pixel is represented by the four red squares and color pixels are the rainbow squares. The red squares correspond to the four tabs of each Z-pixel. (a) 1x1-Z-pixel: its dimension is the same as color pixel dimension. Blue rectangles represent the CDTI memories of the Z pixel. (b) 2x2-Z-pixel: its dimension is twice the size than color pixel dimension.

the matrix. Especially concerning the quality of the depth measurement, because denoising methods use kernels of a few pixels. If the Z-pixels are too spread out in the matrix , there is not enough information for a given kernel to process denoising properly. The impact of noise on the depth measurement may be too significant. In addition, undersampling of the Z pixels can induce aliasing effects. This is why the RGB-Z matrices used in this work are designed with a denser distribution of Z pixels. The total area of the Z pixels represents 1/4 of the total area of the pixel matrix.

3.2.2 Bayer based architecture

Two RGB-Z matrix architectures studied are based on the Bayer pattern. Indeed, the spatial arrangement of the color filters in the Bayer matrix is a good trade-off regarding the risk of aliasing. Moreover, it is the most used pattern, the number of matrix reconstruction algorithms dedicated to this pattern is therefore important. The first matrix architecture is built using the 1x1-Z-pixel and the second one is based on the 2x2-Z-pixel.

3.2.2.1 Mask1

The first RGB-Z matrix based on the Bayer pattern uses the 1x1-Z-pixel and is called *Mask1* (Figure 3.3). It is built from a 2-by-2 unit pattern similar to the Bayer unit pattern with the difference that one of the two green pixels is replaced by the 1x1-Z-pixel. There is therefore one quarter of the color information that is missing in the CDFA image. The green information is close to a luminance information. Thus, we lose half of this information. Finally, it is important to note that there are no

green pixels on the rows and columns with Z pixels, despite the presence of four green pixels in the close neighborhood of a missing one.

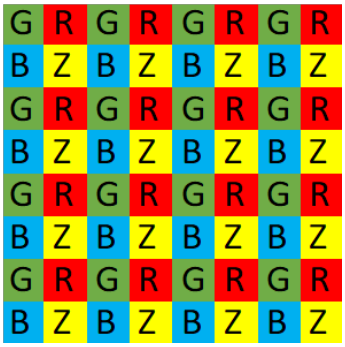
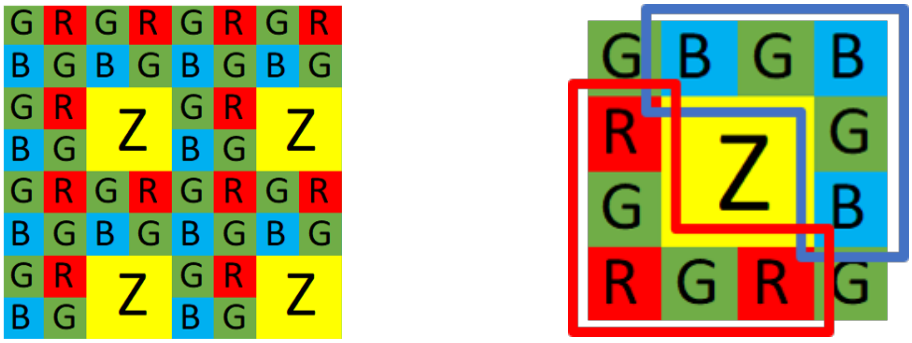


Figure 3.3: Representation of the Mask1: RGB-Z matrix based on the Bayer pattern built using the 1x1-Z-pixel.

3.2.2.2 Mask2

The second RGB-Z matrix based on the Bayer pattern and uses the 2x2-Z-pixel. It is called *Mask2* (Figure 3.4.a). It is built from a 4x4 unit pattern. In this case, four color pixels are missing at a Z-pixel location: one quarter of the color information is missing in the CDFA image. However, compared to the previous architecture, one blue, one red and two green pixels are missing. There is, in this case, no absence of complete color information on rows and columns.



(a) (b)

Figure 3.4: Representation of the Mask2 (a): RGB-Z matrix based on the Bayer pattern built using the 2x2-Z-pixel. (b) Focus on the close environment of a Z-pixel with a noticeable heterogeneous color pixel distribution.

An observation of the close environment of the 2x2-Z-pixels (Figure 3.4.b) allows to highlight a heterogeneous distribution of the color pixels around the Z-pixel. To reconstruct a Bayer pattern, the missing blue pixel is located at the lower left corner of the Z pixel and the missing red pixel is located at the upper right corner.

However, the close distribution of the red (respectively blue) pixels near to the Z-pixel is not homogeneously distributed around the missing color pixel. The grouping of pixels useful to the interpolation in a single direction can induce reconstruction difficulties, especially for thin structures. Considering only this close neighborhood it is impossible to reconstruct some structures. For example, when the pixel Z is located on a corner, if the red (respectively blue) pixels present in the close neighborhood do not belong to the corner structure, then it is impossible to obtain a satisfactory interpolation.

3.2.3 Non-Bayer based architecture

Non-Bayer architectures are designed to optimize the distribution of color pixels in the close neighborhood of the Z-pixel. They are relevant only in the case of a 2x2-Z-pixel based architecture. Indeed in the Mask1 architecture, the distribution of the color pixels in the close neighborhood of the Z-pixel is homogeneous.

3.2.3.1 Non-Bayer1

The first non-Bayer architecture developed and its reconstructed CFA are respectively shown in the Figure 3.5 a and b. The CDFA is composed of a 16x16 unit pattern composed of four 2x2-Z-pixels. The total amount of missing color information due to the Z pixel is the same as the Mask2. The main difference concerns the rearrangement of the available chrominance information (red and blue) around the Z-pixels. The number and distribution of green pixels do not change.

3.2.3.2 Non-Bayer2

The second case presented in the Figure 3.5.c is similar to the previous one. The CDFA is the same as the Non-Bayer1 architecture but the layout of the color pixels after reconstruction is different. The positions of the two reconstructed pixels, red and blue, are switched between the two architectures. The interest of this change is to locally reduce the pixel clusters of the same channel in order to have a more homogeneous spatial distribution of the blue and red pixel for the demosaicing step. In the case of Non-Bayer1, we can see on the Figure 3.5.b the presence of several groupings of 3 successive pixels in diagonal of the same color channel. On the contrary, on Figure 3.5.c, there is no more cluster of 3 red (respectively blue) pixels in a row.

One of the main drawbacks of non-Bayer based architectures is that most of the demosaicing algorithms are optimized or dedicated to Bayer matrices. It is therefore

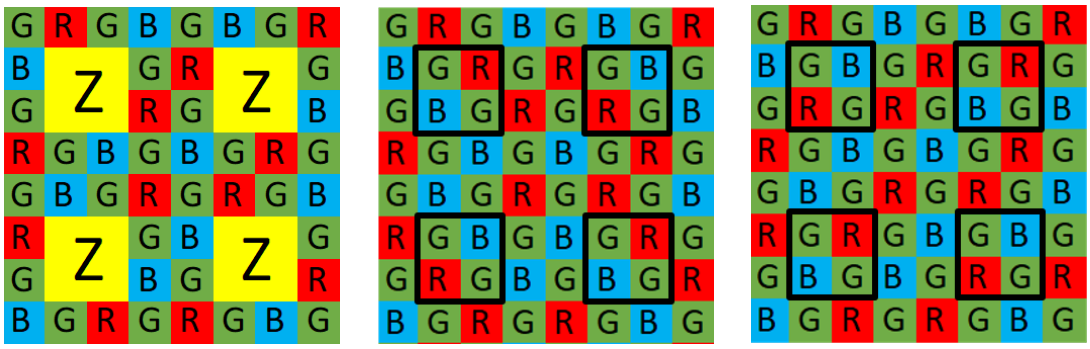


Figure 3.5: Representation of the proposed RGB-Z non-Bayer architecture (a) and its corresponding CFA images. (b) Corresponds to the Non-Bayer1 and (c) Corresponds to the Non-Bayer2. The black squares indicate the positions of the reconstructed pixels.

necessary to have a demosaicing algorithm that is at least as good as Bayer's matrix-specific algorithms. In the next section we will see the demosaicing algorithms used and their impacts on the choice of different RGB-Z architectures.

3.3 Demosaicing algorithms

One of the final objectives of an ISP processing chain is to obtain a good quality output image. The quality of the image reconstruction does not depend on a single step, but on the successive steps of the ISP chain (section 1.2.2) : filtering, color reconstruction, missing information reconstruction...

In the context of a RGB-Z sensor, the choice of one CDFA rather than another may imply to modify the reconstruction solution and thus the quality of this reconstruction. In the same way, the choice of one CFA rather than another impacts the demosaicing step. It is therefore important to study the quality of the reconstructed image at the output of the demosaicing step regarding the CFAs. In this section, we present the algorithms study for the considered CFAs and their impact on the choice of the RGB-Z architecture.

3.3.1 CFA-independent demosaicing algorithm

A CFA-independent demosaicing algorithm has been implemented for the preliminary reconstruction of non-Bayer CFA images. It is mainly inspired from the first part of the methods proposed by Alain Horé and Djemel Ziou [86]. The implemented algorithm is a weighted average sum divided into two steps: the first step reconstructs the green channel and during the second step, the red and blue channels are reconstructed using the interpolated green channel.

3.3.1.1 Method

The algorithm works on a 5x5 computation window ξ centered on the missing pixel. For a given window, the weight $\omega(i,j)$ of each green pixel used for the interpolation is computing (equation 3.1) following three parameters:

- A distance parameter α.
- An intensity parameter β.
- And an edge parameter γ.

$$\omega(i,j) = \alpha(i,j) \times \beta(i,j) \times \gamma(ori) \tag{3.1}$$

The two first parameters α and β adopt the idea of a bilateral interpolation [43]. It is a combination of a spatial filter with a filtering on pixel intensities. The difference with a classical bilateral interpolation is that the pixel intensity filtering does not take into account the missing pixel because the color intensity is not available there.

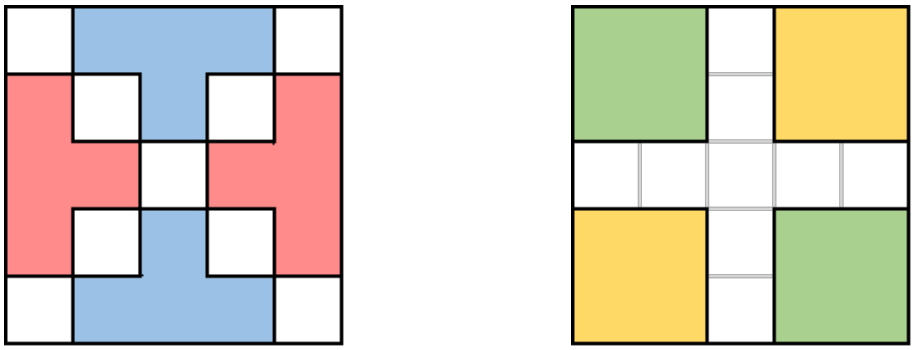
The distance parameter α weights pixels according to their distance from the missing pixel. The farther the pixel is from the reconstructed pixel, the lower is its weight. The Euclidean distance is used to evaluate the distance between the missed pixel (p,q) and the current green pixel location (i,j) in the considered kernel ξ:

$$\alpha((p,q)(i,j)) = \frac{1}{\sqrt{(p-i)^2 + (q-j)^2}} \tag{3.2}$$

The second parameter concerns the pixels intensities, noted β. It corresponds to the photometric distance. It allows reducing the impact of the pixels that have an intensity value very different from the other pixels. For a given kernel, each green pixel intensity $(I(i,j)_G)$ is compared with the intensity of the other green pixels intensity $(I(g,h)_G)$:

$$\beta(i,j) = \frac{1}{1 + \sum\limits_{(g,h)\in\xi_G} |I(i,j)_G - I(g,h)_G|} \tag{3.3}$$

The last parameter is an edge parameter γ. It adapts each weight according to the detected edges. The edges are evaluated along four orientations *ori* (horizontal, vertical and two diagonal orientations) at the missing pixel location. The four orientations *ori* and their corresponding pixels are represented in the Figure 3.6. To evaluate an edge for a given orientation, a gradient is computed as the sum of two-by-two differences between pixels from the same color channel. The pixels used to compute a difference are symmetrically located with respect to the central



(a) (b)

Figure 3.6: Representation of the different overlapped areas corresponding to the four orientations used to compute the four gradients. (a) Blue areas correspond to the vertical gradient, pink areas correspond to the horizontal gradient. (b) Yellow and green areas correspond to the two diagonal gradients.

missing pixel of the computational kernel.

As the algorithm can reconstruct color images from various CFA, the pixels distribution across a given kernel is not predefined, this is why the gradients are normalized according to their number of computed differences:

$$Grad_{scaled}(ori) = \frac{Grad(ori)}{NbDiff(ori)} \tag{3.4}$$

Where $Grad_{scaled}$ is a scaled gradient and $NbDiff$ is the number of differences used to compute the corresponding gradient. The four averaged gradients are then linearly discriminated to obtain the edge parameter γ:

$$\gamma(ori) = 1 - \left(\frac{Grad_{averaged}(ori)}{Grad_{Max}} \right) \tag{3.5}$$

Where $Grad_{Max}$ is the maximum of the four scaled gradients. The discrimination function is used to have a relative factor for each orientation. Indeed, the higher the evaluation is for a given orientation compared to the others ones, the smaller the edge-directed factor is for this orientation. The edge parameter for pixels that overlap two adjacent areas (Figure 3.6) corresponds to the average of the two corresponding edge parameters. For example the edge parameter of a pixel overlapping the vertical area and a diagonal area is calculated as follows: $\gamma(ori) = (\gamma(ori_{vertical}) + \gamma(ori_{diagonal}))/2$. Finally, the weights are normalized (3.6).

$$\omega'(i,j) = \omega(i,j) / \sum_{(g,h)\in\xi} \omega(g,h) \tag{3.6}$$

And then used for the interpolation of the green pixel (3.7).

$$I(p,q)_G = \sum_{(i,j)\in\xi_G} \omega'(i,j) \times I(i,j)_G \tag{3.7}$$

During the second step the red and blue channels are reconstructed using the green plane. This is a constant-hue interpolation: the differences between the red pixels (respectively blue pixels) and the reconstructed green pixels are used instead of their intensity value. The calculation of the weights is similar to the method used for the green plane. However, only the bilateral interpolation part is considered: the distance parameter α (equation 3.2) and the pixel intensity parameter β (equation 3.3). Indeed, the intensity differences take into account the green plane which already contains the edge information calculated during the first step. The calculation of weights is defined by equation 3.8. The final weights are computed using formula 3.6 adapted to the red (respectively blue) channel.

$$\omega(i,j) = \alpha(i,j) \times \beta(i,j) \tag{3.8}$$

The red (respectively blue) pixel interpolation is computed as follow :

$$I(p,q)_R = I(p,q)_G + \sum_{(i,j)\in\xi_R} \omega'(i,j) \times (I(i,j)_R - I(i,j)_G) \tag{3.9}$$

The green value at the interpolated pixel location is added to the weighted sum of intensity differences to obtain a conform reconstructed red (respectively blue) pixel value.

3.3.1.2 Analysis

In this section, we perform a brief visual analysis of the performance of the CFA-independent algorithm on different CFAs. The Figure 3.7 summarizes the three comparisons. In these comparisons the input images are CFA images. They do not include any Z pixels.

Figure 3.7: Simplified diagram of the tested reconstructions to visually assess the quality of the CFA-independent algorithm according to different CFA images.

Examples of color images reconstructed from the different CFA images are shown in Figure 3.8. The reconstructed images show many artifacts. False color and block artifacts are visible, especially at the edges. Zipper effects are also visible. In general, this demosaicing method does not give satisfactory visual results.

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

Figure 3.8: Reconstructed images with the CFA-independent algorithm with different CFAs. (a) Original full color image "lighthouse" [11]. (b) Ground truth. (c) Bayer CFA. (d) CFA corresponding to Non-Bayer1. (e) CFA corresponding to Non-Bayer2. The three reconstructed images present false colors and block artifacts. Moreover, aliasing effect is visible on both images (d) and (e).

It is interesting to note that the color image reconstructed from the Bayer image visually shows less artifacts compared to the other two reconstructed images. This means that even if this algorithm is not optimized for a given CFA, the Bayer pattern allows an image reconstruction with fewer artifacts.

3.3.1.3 Mozart versus CFA-independent algorithm for a Bayer pattern

A second demosaicing algorithm is used. Unlike the previous one, this algorithm, which is used by STMicroelectronics, is an algorithm dedicated to the Bayer pattern. It is called Mozart [18] and is based on wavelets, as explained in the section 2.3.4. The Figure 3.9 summarizes the two comparisons.

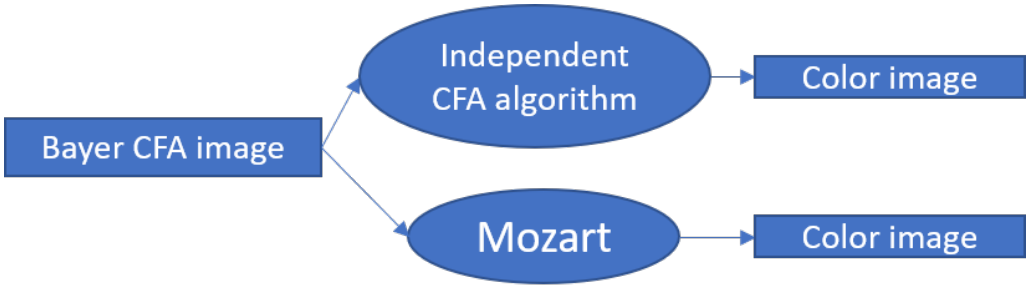


Figure 3.9: Simplified diagram of the tested reconstructions to compare the quality of the reconstruction after demosaicing the Mozart and the CFA-independent algorithm.

Examples of reconstructed image using both algorithms are shown in the Figure

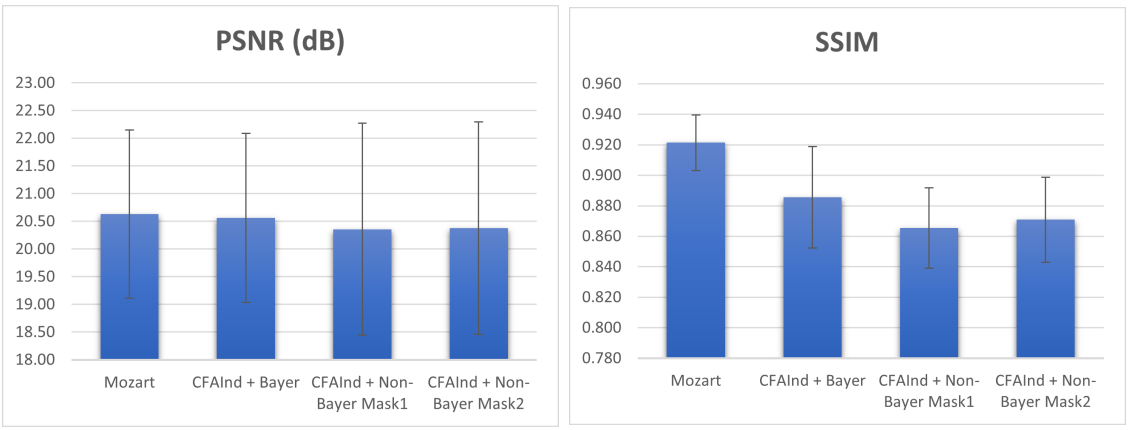
3.10. Mozart presents far fewer color and block artifacts. Some false color artifacts are still present along the edge but they are visually reduced compared to the CFA-independent algorithm.

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

Figure 3.10: Result images for the "lighthouse" from Kodak database [11]. Both demosaicing algorithms are used on Bayer images. (a) Original full color image "lighthouse" [11]. (b) Ground truth. (c) Reconstructed image using Mozart [18]. (d) Reconstructed image using the CFA-independent algorithm presented in the section 3.3.1.

In addition, the quality of the color image reconstruction is compared using two metrics. In order to have a more representative sample, 24 images from the Kodak [11] dataset are evaluated with the different patterns and reconstruction methods (Figure 3.11). The two metrics used are the PSNR and the SSIM, they are detailed in the section 2.2. In terms of PSNR, Mozart presents a slight improvement over the CFA-independent algorithm (averaged difference of 0.7 dB). However, the difference is more pronounced in terms of SSIM. This metric has been developed to measure the visual quality of a reconstructed image compared to a reference image. The SSIM scores are correlated with the visual appreciation of the reconstructed images (Figure 3.10). Furthermore, these data confirm that among the different CFAs used with the CFA-independent algorithm, the Bayer pattern has the best results both in terms of PSNR and SSIM.

Whatever the CFA used, the demosaicing step reconstructs two thirds of the total information of an image (reconstruction of two values per pixel). Whereas the reconstruction of the missing color information due to the Z pixel represents, in the cases presented above, only one quarter of the information of the CFA image, i.e. 1/12 of the total information of the color image. The RGB demosaicing step should thus not be neglected versus the step of color information reconstructing at the Z-pixel location. This is why, in view of this analysis and the results obtained, we have chosen to focus on the study of RGB-Z architectures based on the Bayer CFA. Moreover it allows to use the Mozart algorithm for the demosaicing step which is already validated by STMicroelectronics. In the following sections we therefore



(a) (b)

Figure 3.11: Averaged performances of demosaicing methods in terms of (a) PSNR and (b) SSIM, computed on the Kodak database [11]. Results details are available in appendix A.1.

consider RGB-Z architectures based on the Bayer pattern. The Bayer images reconstructed with Mozart are now considered as reference images quality target to reach in order to evaluate the missing color information algorithms.

3.4 CFA reconstruction algorithms

This thesis mainly focuses on the study and implementation of color information reconstruction methods for mixed RGB-Z matrices. The specifications of the studied matrices and the considerations concerning the arrangement of the color pixels and the depth pixels in the matrix, as well as the choice of the CFA were discussed in the previous sections. In this section, we focus on the implemented reconstruction algorithms at Z pixel location. All the algorithms are implemented using the Python 2.7 environment and some parts of the algorithms, which need to be accelerated, are implemented in C.

The images given in this section illustrate the quality of the reconstructed color images by the presented algorithms according to both Mask1 (section 3.2.2.1) and Mask2 (section 3.2.2.2). The image simulation chain, the processing chain and the results for each method are detailed in the next chapter.

3.4.1 Bilateral based algorithm

The first implemented solution is inspired from the bilateral interpolation [43]. In the same way as the CFA-independent algorithm (3.3.1.1), a distance parameter α and an intensity parameter β are computed. Due to the absence of color information at the location of the interpolated pixel, the intensity parameter of a given weight

is calculated by evaluating its intensity regarding the pixels from the same color channel in the considered neighborhood. The computational kernel used corresponds to a 5x5 window ξ centered on the missing pixel. The weights are computed using 3.8. For each weight, the intensity parameter β, is computed using 3.3, and the range parameter α is computed with a 2-dimension Gaussian filter:

$$\alpha(i,j) = \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right), \tag{3.10}$$

σ was empirically determined according to the kernel size and is equal to 1. The choice of a Gaussian filter rather than the inverse of the Euclidean distance (3.2) allows a faster decrease of the weight according to the distance. Thus, it values the pixels close to the missing pixel. On the Figure 3.12 are represented two simulated RGB-Z images (Mask1 and Mask2) reconstructed using the bilateral algorithm and demosaiced using Mozart compared to the reference Bayer image demosaiced with Mozart. On the Mask1, we can see a blurring effect along the edges of the fence and also more aliasing/false colors. On the Mask2, the RGB-Z pattern is clearly visible. This is similar to a Zipper effect of two pixels wide, which corresponds to the width of a Z-pixel. We see that this algorithm is not able to prevent interpolation across edges.

(a) (b) (c)

Figure 3.12: Color images reconstructed from simulated RGB-Z data using the bilateral solution presented in section 3.4.1 compared to the reference image. RGB-Z data are simulated using both Mask1 (3.2.2.1) and Mask2 (3.2.2.2). The demosaicing algorithm used for the three reconstructed images is Mozart [18]. (a) Reference cropped image. (b) simulated RGB-Z image reconstructed based on the Mask1.(c) simulated RGB-Z image reconstructed based on the Mask2.

3.4.2 Edge directed algorithm

An edge-directed interpolation solution has been implemented to correct the structural artifacts that are mainly located around the edges. This method uses gradients

calculation to determine the direction of the edges. Two methods for calculating the gradients are explored. The first solution is directly inspired from the first step on the CFA-independent algorithm. The gradients are calculated by central symmetry around the missing pixel. For the second one, the gradients are now calculated with respect to the axes of symmetry of the kernel. A representation of these axes of symmetry is given in the figure 3.13

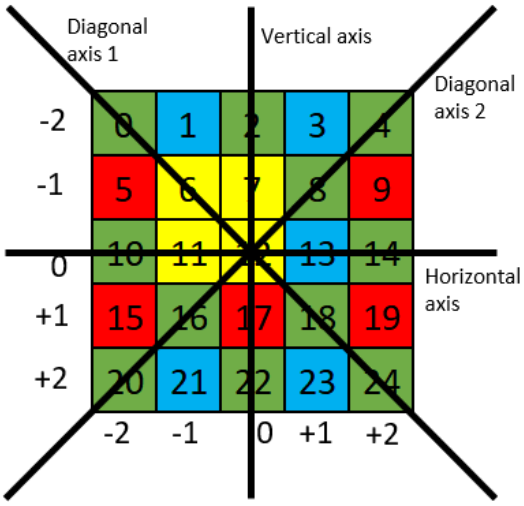


Figure 3.13: Representation of a computational kernel of the Mask2. The four axes of symmetry are illustrated by the black lines.

3.4.2.1 Gradients computation around the missing pixel

This first solution is very similar to the solution implemented for the CFA-independent algorithm. To compute the weights used for the interpolation of a missing color pixel, equations 3.1, 3.10 and 3.3 are still valid. The calculation of the edge parameter is different. The areas are no longer overlapped to compute it. Indeed, the majority of artifacts in the reconstructed RGB-Z images are located along the edges. Overlapping areas when calculating the edge parameters does not result in sharp edges after interpolation. The overlapping may value the weights of some pixels that are located between two areas that do not necessarily correspond to the main direction determined for the interpolation. In this case, the reconstructed edges are more blurred. This is why the edge parameter is now computed using the four areas illustrated on the Figure 3.14.

A second change is the discrimination function used for the gradients. While a linear discriminated function is used in the CFA-independent algorithm, a Gaussian discrimination function is used for the missing color pixel interpolation. The reason of this change is the same as for the use of a Gaussian function for the bilateral filter: the slope of the function decreases faster. This allows a stronger discrimination of

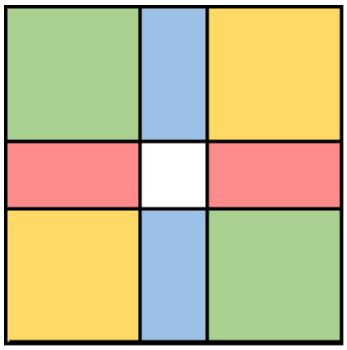


Figure 3.14: Representation of the four non-overlapped areas corresponding to the orientations *ori* to compute the four gradients. The blue areas correspond to the vertical gradient, the red areas correspond to the horizontal gradient. The yellow and green areas correspond to the two diagonal gradients.

the gradients.

$$\gamma(ori) = \exp\left(-\frac{Grad_{averaged}(ori)^2}{2\sigma^2}\right) \tag{3.11}$$

In this case, σ corresponds to the minimum gradient between the four averaged gradients. On the Figure 3.15 are represented two simulated RGB-Z images (Mask1 and Mask2) reconstructed using the presented edge-directed algorithm. There is a significant reduction of structural artifacts for the Mask2. However, there are still many remaining artifacts. Concerning the Mask1, the color artifacts are always present along the edges.

(a) (b) (c)

Figure 3.15: Color images reconstructed from simulated RGB-Z data using the edge directed solution using central symmetry presented in section 3.4.2.1 compared to the reference image. RGB-Z data are simulated using both Mask1 (section 3.2.2.1) and Mask2 (section 3.2.2.2). (a) Reference cropped image. (b) Simulated RGB-Z image reconstructed based on the Mask1. (c) Simulated RGB-Z image reconstructed based on the Mask2.

3.4.2.2 Gradients computation along the kernel axes

A second edge-directed algorithm solution has been implemented. It is a weighted-sum based interpolation using the same three factors (distance factor (3.10), intensity factor (3.3) and edge-directed factor (3.11)). The main difference concerns the edge-directed factor computation. In the previous algorithm, the gradients are computed using a central symmetry with the missing pixel, while in this algorithm the gradients are computed using the axes of symmetry of the kernel.
The method is inspired from the Prewitt edge-detection operator [101]. The Prewitt operator is only sensitive to the vertical and horizontal orientations. It is not sensitive to the diagonal orientations. To address this issue, we propose a gradient computation along the diagonal orientations based on the diagonal axes. The gradients are then computed by the sum of pairwise differences between known color pixels. The color pixels used for each difference are located on either side of a given axis of symmetry of the kernel. The Figure 3.16 represents the four masks used to compute the gradients. The relationships between the three channels based on the constant color difference assumption presented in section 2.3.2.3, are used to compute inter-channel differences. The computation of the four gradients is detailed in the equations 3.12 to 3.15. Pixel indices refer to the Figure 3.13.

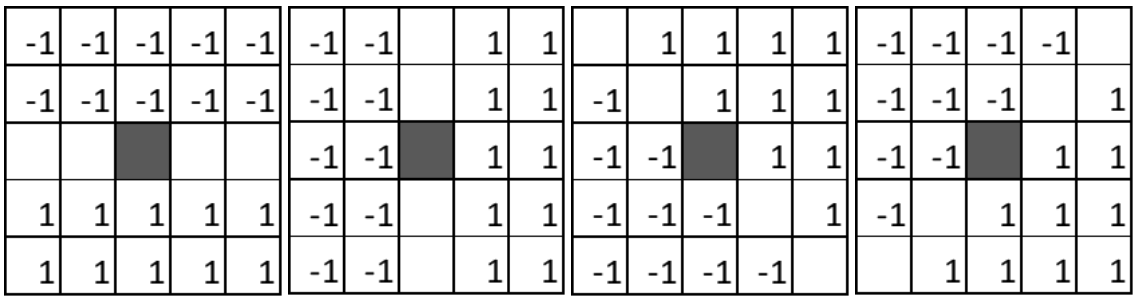


Figure 3.16: General computation masks of the centered on the missing pixel, for the four orientations of the gradient. (a) Vertical gradient. (b) Horizontal gradient. (c) and (d) two diagonal gradients.

$$Grad_{vertical} = \sum_{m\in[-2;2]} [|I_{\eta\neq 3}(-2,m) - I_{\eta\neq 3}(+2,m)| + |I_{\eta\neq 3}(-1,m) - I_{\eta\neq 3}(+1,m)|] \quad (3.12)$$

$$Grad_{horizontal} = \sum_{m\in[-2;2]} [|I_{\eta\neq3}(m,-2) - I_{\eta\neq3}(m,+2)| + |I_{\eta\neq3}(m-1) - I_{\eta\neq3}(m,+1)|] \quad (3.13)$$

$$Grad_{diagonal1} = \sum_{m\in[-2;1]} \sum_{k\in[0;1]} [|I_{\eta\neq3}(m+1+k,m-1-k) - I_{\eta\neq3}(m-1-k,m+1+k)| + |I_{\eta\neq3}(m+1+k,m-k) - I_{\eta\neq3}(m-k,m+1+k)|] \quad (3.14)$$

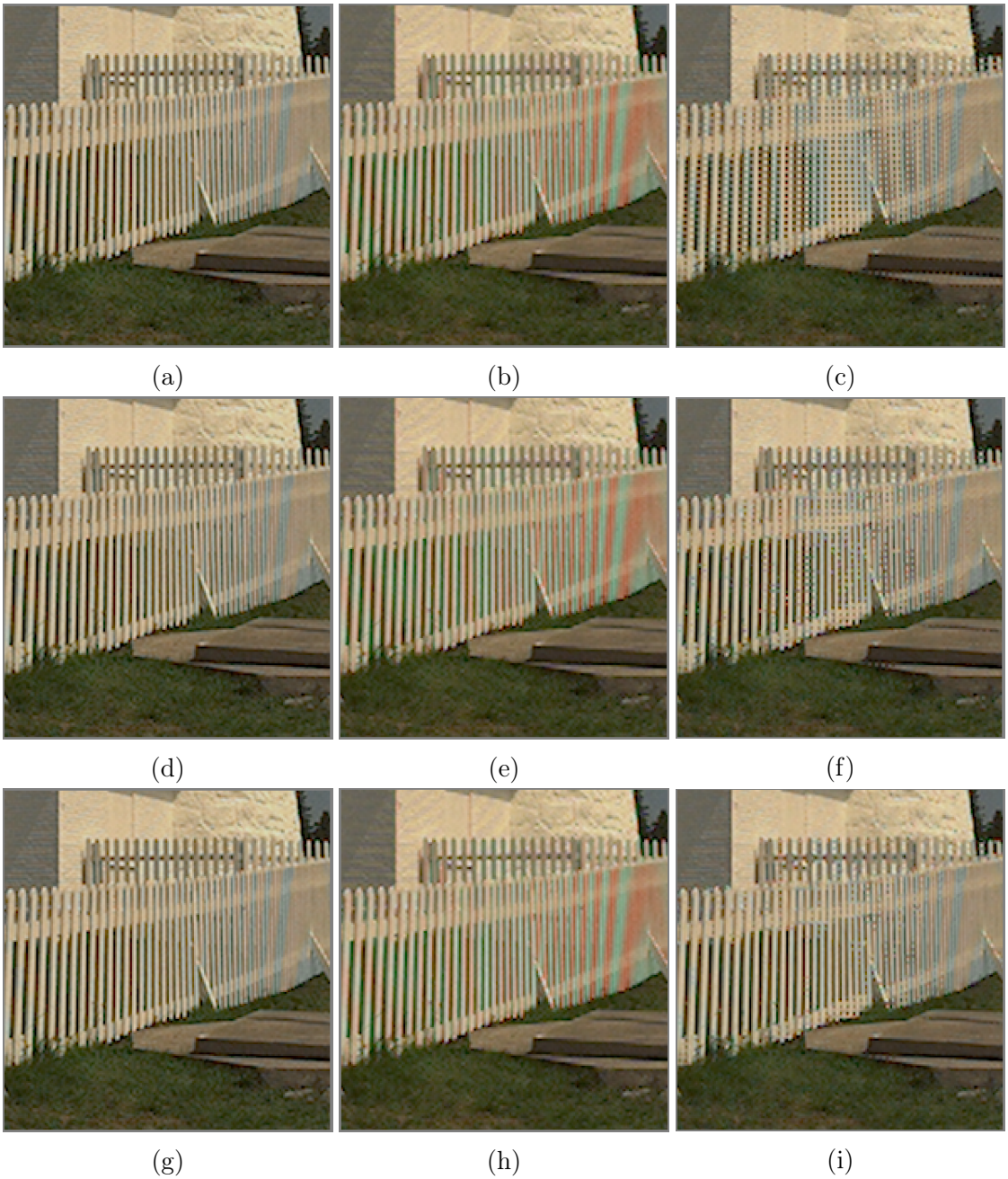
$$Grad_{diagonal2} = \sum_{m\in[-2;1]} \sum_{k\in[0;1]} [|I_{\eta\neq3}(m+1+k,1+k-m) - I_{\eta\neq3}(m-1-k,-m-1-k)| + |I_{\eta\neq3}(m+1+k,k-m) - I_{\eta\neq3}(m-k,-m-1-k)|] \quad (3.15)$$

Where, $I(i,j)$ represents the intensity value of a pixel (i,j) and η corresponds to the color channel of a given pixel. Namely, $\eta = 0$ corresponds to red (R) channel, $\eta = 1$ corresponds to the green (G) channel, $\eta = 2$ corresponds to the blue (B) channel, and $\eta = 3$ correspond to the Z channel. According to the Z-pixel distribution across the kernel, the mask used to compute a gradient can vary. It depends on the kernel configuration: only the color pixels are used to calculate the gradient. The masks can be adjusted according to the current RGB-Z architecture. The Z-pixel values are not taken into account, so the differences including a Z-pixel are not calculated. The number of differences calculated for each orientation can be different. The four gradients are normalized according to their number of differences used to compute them. The steps to achieve the interpolation are similar to the previous sections: the weights are computed using equation 3.1, they are normalized with equation 3.6 and finally the missing color pixel $I_{\eta\neq3}(p,q)$ is interpolated using a weighted averaged summation of the same color channel pixels. For example to interpolate a missing red pixel (respectively η is adapted following the color channel reconstructed), we use:

$$I_{\eta=0}(p,q) = \sum_{(g,h)\in\xi} \omega'(g,h) I_{\eta=0}(g,h) \quad (3.16)$$

On the Figure 3.17 are represented two simulated RGB-Z images (Mask1 and Mask2) reconstructed with the presented edge-directed algorithm. They are also compared to the images reconstructed with the bilateral and EDI methods pre-

sented above. A reduction of structural artifacts is again observed on the Mask2. The pattern of the RGB-Z architecture remains however clearly visible. About the Mask1, there is no noticeable improvement between the different solutions proposed. Aliasing and false color artifacts are still present.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.17: Color images reconstructed from simulated RGB-Z data using the bilateral solution (a-c), the EDI solution using central symmetry (d-f), and the EDI using axial symmetry (g-i). RGB-Z data are simulated using both Mask1 (section 3.2.2.1) and Mask2 (section 3.2.2.2). The first column (a, d, g) represents the reference images. The second column (b, e, h) corresponds to the reconstructed images using the Mask1 and the third column (c, f, i) corresponds to the reconstructions based on the Mask2.

3.4.3 Semi-gradients

The reconstruction methods using EDI presented in the previous sections still have some remaining artifacts. These artifacts are mainly present along the thin edges. A study is first performed for the Mask1 in order to understand the cause of these artifacts and thus to propose a solution that corrects them. In a second step, the proposed reconstruction solution for the Mask1 is adapted to the Mask2.

3.4.3.1 Artifacts study for the Mask1

In order to better understand the artifacts present in an image simulated with the Mask1, it is necessary to look back at the RGB-Z matrix and more precisely at the computational kernel. On the Figure 3.18, we can see that the amount of green information available to reconstruct the missing pixel is limited. Indeed, there are only four green pixels in the considered neighborhood. Moreover, the distribution of these pixels does not allow a purely horizontal or purely vertical interpolation. This effect is even more visible when the reconstructed pixel is located along a narrow edge. A visual example is given on the Figure 3.19. On this example we can see some false colors artifacts along the vertical edges of the fork. There is a reconstruction error despite the fact that the gradients correctly identify the vertical edge. This error is due to the ill-suited distribution of green pixels in the neighborhood of the missing pixel.

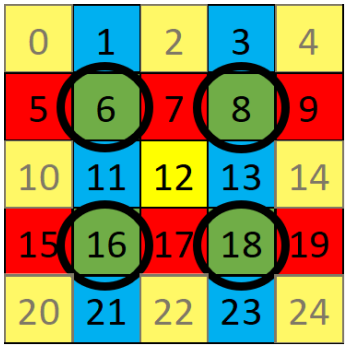


Figure 3.18: The 5x5 computational kernel for the Mask1 centered on the missing pixel (pixel number 12). Black circle show the four green pixels used to interpolate the central missing pixel.

In this case, the four green pixels belong to two different textures of the image. The two left pixels belong to the white fork area and the two right pixels belong to the background area. Moreover, they are not on the vertical axis of the kernel which corresponds, in this case, to the priority axis of interpolation. This is why, during the interpolation the weight distribution is homogeneous between the four

pixels (they all are equal to 1/4). In order to solve this issue, we propose to estimate to which region of the kernel the missing pixel most likely belongs.

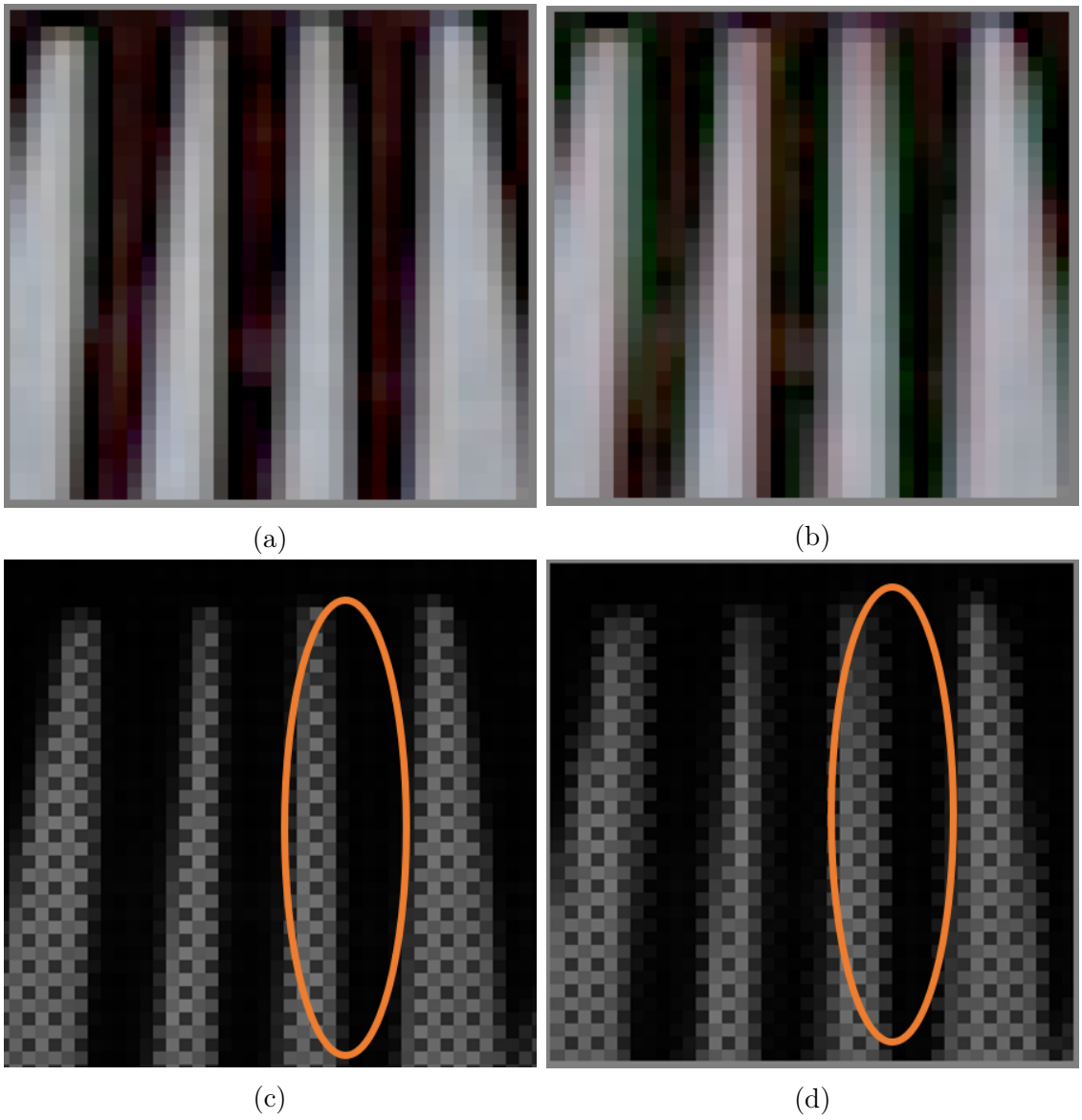


Figure 3.19: Example of reconstructed Bayer images from the Mask1 and their corresponding demosaiced image. (a) Reference color image. (b) Reconstructed color image using EDI section 3.4.2.2. (c) Reference CFA that corresponds to the reference color image. (d) Reconstructed CFA using algorithm presented in section 3.4.2.2. The orange circles show the pixel reconstruction errors.

3.4.3.2 Proposed solution for the Mask1

In order to estimate to which texture the missing pixel belongs among the different potential areas present in a given kernel, we propose to implement a new operator that we call *semi-gradient*. As we do not have color information directly available at the location of the missing pixel, we study its closest neighbors. The objective

is to study the local variations of the texture inside the computational kernel. The developed operator corresponds to the calculation of a local gradient in the kernel, while the gradient studies the texture variations through the kernel. This is why it is called *semi-gradient*.

The SG operator is specifically designed to work on a Bayer based architecture. In the same vein as the calculation of the gradient presented in section 3.4.2.2, the SG is calculated by considering horizontal and vertical symmetry axes of the kernel. The idea is to compare the differences between the pixels close to the missing pixel and pixels at the kernel borders. If the tested pixels are close, in terms of pixel intensity, to a specific area of the kernel, then the weights of the pixels in this area are valued. Two arrays of eight values are computed corresponding to eight vertical and eight horizontal differences (equations 3.17). They are computed using masks shown in Figure 3.20.

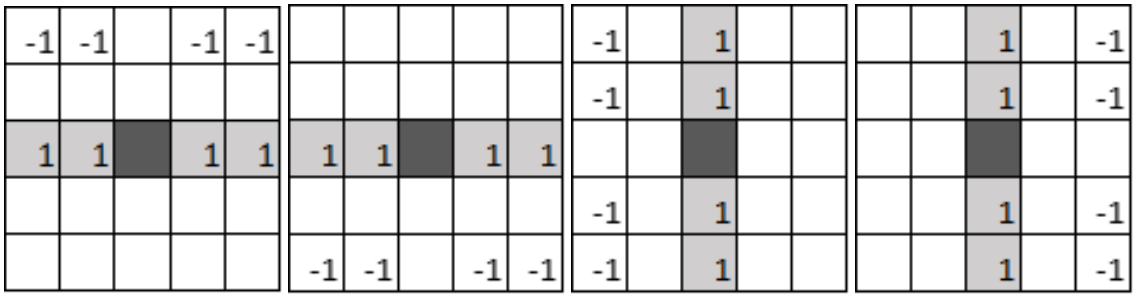


Figure 3.20: Differences computation masks used to compute SGs. Close pixels are marked in Grey. The missing color pixel is the darkest. (a) Comparison between close pixels and the top of the kernel. (b) Comparison between close pixels and the bottom of the kernel. (c) Comparison between close pixels and the left part of the kernel. (d) Comparison between close pixel and the right part of the kernel. (a) and (b) correspond to the vertical differences. (c) and (d) correspond to the horizontal differences

$$diff_{vert}[2j] = |I_{\eta_1}(0, j-2) - I_{\eta_2}(-2, j-2)|,\ with\ \eta_1 = \eta_2 \neq \eta_3 \tag{3.17a}$$

$$diff_{vert}[2j+1] = |I_{\eta_1}(0, j-2) - I_{\eta_2}(2, j-2)|,\ with\ \eta_1 = \eta_2 \neq \eta_3 \tag{3.17b}$$

$$diff_{hori}[2j] = |I_{\eta_1}(j-2, 0) - I_{\eta_2}(j-2, -2)|,\ with\ \eta_1 = \eta_2 \neq \eta_3 \tag{3.17c}$$

$$diff_{hori}[2j+1] = |I_{\eta_1}(j-2, 0) - I_{\eta_2}(j-2, 2)|,\ with\ \eta_1 = \eta_2 \neq \eta_3 \tag{3.17d}$$

The index $j \in [0, 4]$ is the index of both arrays. Horizontal and vertical differences are associated to form the SGs. A total of eight SGs SG are computed (equations 3.18), two for each orientation. To facilitate the use of SGs, we attribute cardinal

points to them: North, South, East, West.

$$SG_N = \sum_{j\in[0;2;6;8]} diff_{vert}[j] \tag{3.18a}$$

$$SG_S = \sum_{j\in[1;3;7;9]} diff_{vert}[j] \tag{3.18b}$$

$$SG_W = \sum_{j\in[0;2;6;8]} diff_{hori}[j] \tag{3.18c}$$

$$SG_E = \sum_{j\in[1;3;7;9]} diff_{hori}[j] \tag{3.18d}$$

$$SG_{NW} = \sum_{j\in[0;2]} diff_{vert}[j] + diff_{hori}[j] \tag{3.18e}$$

$$SG_{SE} = \sum_{j\in[7;9]} diff_{vert}[j] + diff_{hori}[j] \tag{3.18f}$$

$$SG_{NE} = \sum_{j\in[6;8]} diff_{vert}[j] + \sum_{j\in[1;3]} diff_{hori}[j] \tag{3.18g}$$

$$SG_{SW} = \sum_{j\in[1;3]} diff_{vert}[j] + \sum_{j\in[4;6]} diff_{hori}[j] \tag{3.18h}$$

They are then discriminated using 3.11. Finally the weights are computed using only the SGs. There are only four weighted pixels used for the interpolation, while there are eight SGs. Each weight is therefore the linear combination of three SGs. For example, the weight of the pixel number 6, that is located to the top right area of the kernel (Figure 3.18), is computed by the linear combination of SG_N, SG_W and SG_{NW}. The four weights are computed with equations 3.19.

$$\omega_6 = SG_N + SG_W + SG_{NW} \tag{3.19a}$$

$$\omega_8 = SG_N + SG_E + SG_{NE} \tag{3.19b}$$

$$\omega_{16} = SG_S + SG_W + SG_{SW} \tag{3.19c}$$

$$\omega_{18} = SG_S + SG_E + SG_{SE} \tag{3.19d}$$

It is interesting to note that in the case of Mask1, gradients are no longer used to calculate weights. Only SGs are considered. The particular distribution of green pixels useful for interpolation does not allow direct interpolation along a direction. However, the SGs study the belonging of the central pixel and its neighborhood compared to the rest of the kernel. They thus allow to promote some areas in the kernel. A practical example of a pixel reconstruction is given in appendix A.2. An example of an image reconstructed using the SG method is given in Figure 3.21.c and its corresponding CFA Figure 3.21.f . The edges of the fork are sharper and

there is also a reduction of false color artifacts along the edges. The flowchart of the algorithm presented in this section is described in Figure 3.22. An other example of reconstructed CFA using the semi-gradient method compared to the EDI method is given in appendix A.3.

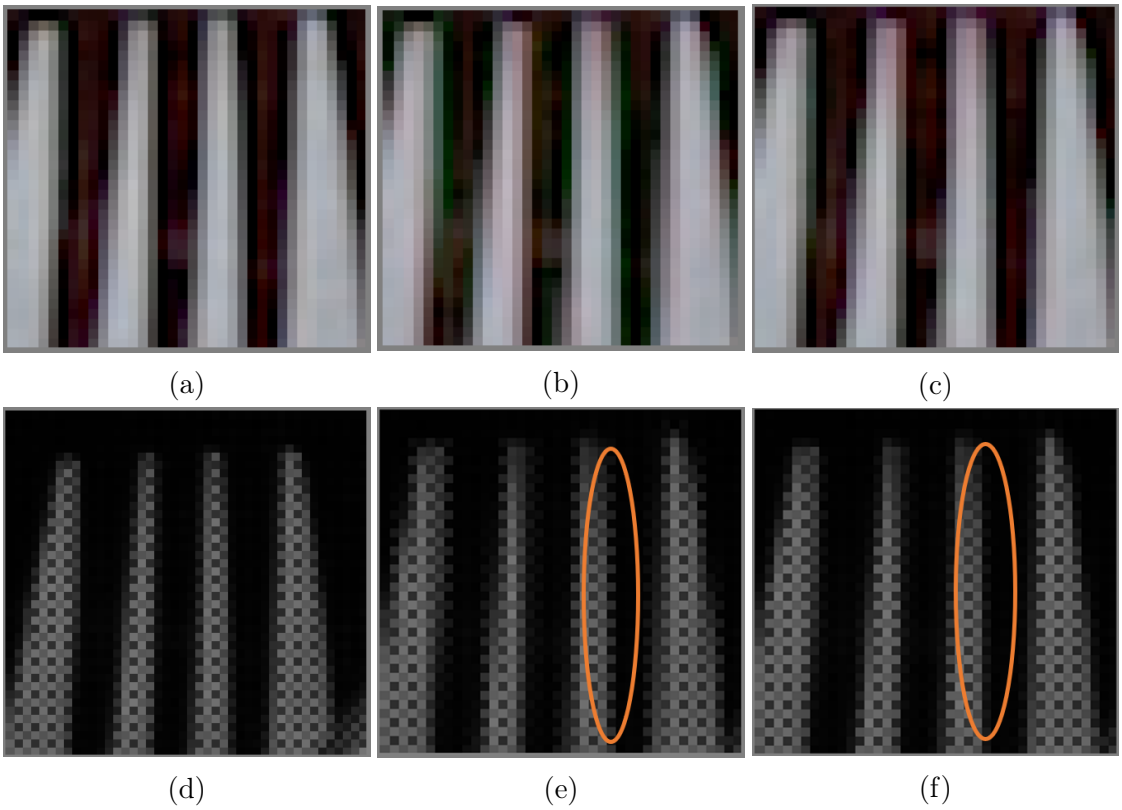


Figure 3.21: Example of reconstructed Bayer images from the Mask1 and their corresponding demosaiced image. (a) Reference color image. (b) Reconstructed color image using EDI section 3.4.2.2. (c) Reconstructed color image using SGs. (d) Reference CFA that corresponds to the reference color image. (e) Reconstructed CFA using algorithm presented in section 3.4.2.2. (f) Reconstructed CFA using SGs. The orange circles show the improvement of the pixel reconstruction between the two solutions.

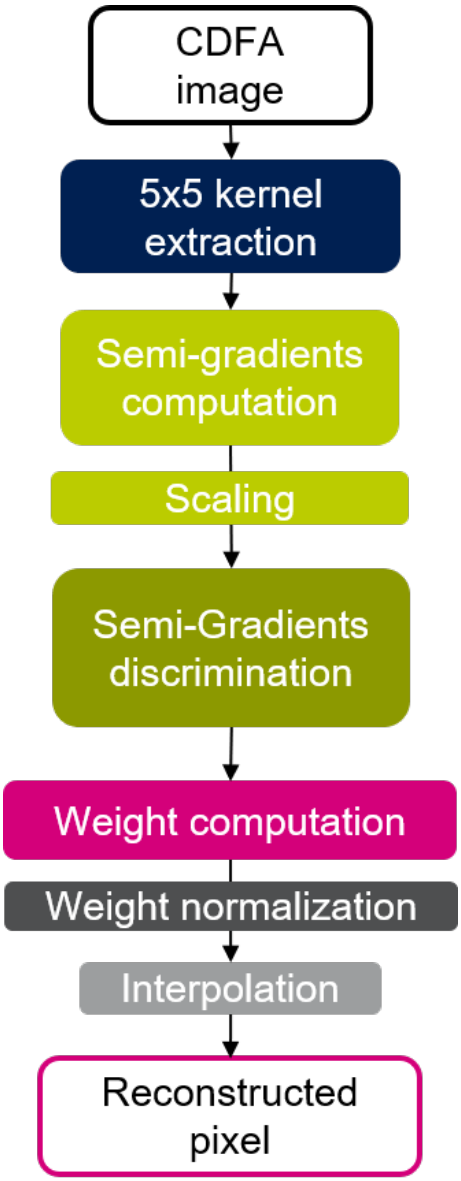


Figure 3.22: Flowchart of a pixel reconstruction with the solution using the SG for the Mask1.

3.4.3.3 Method adapted to the Mask2

The idea of semi-gradients, which study the close neighbors of the missing pixel in order to determine to which texture it belongs, is adapted to the case of the Mask2. The configuration of Mask2 is however different from Mask1. Indeed, the amount of information available in the 5x5 neighborhood of the missing pixel is not the same. There is more information in the case of Mask2. Moreover, this information is spread over all the orientations. The four different computational kernels used to reconstruct the missing color information are shown in the Figure 3.23. For the cases of red and blue pixel reconstruction, eight pixels are available to interpolate the missing pixel. While in both cases of green pixel reconstruction, eleven pixels are available in the neighborhood. The algorithm implemented for the Mask2 continues to use the gradients, but it also uses SGs to correct some artifacts.

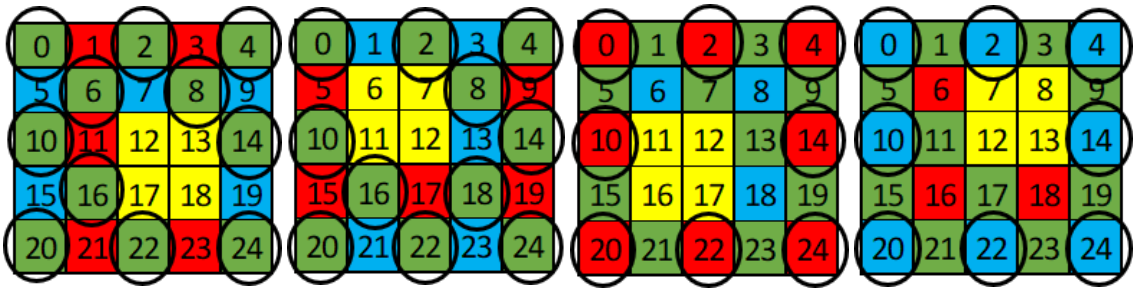


Figure 3.23: Representation of the four kernel computation of the Mask2. The dark circles represent for each case the pixels used to interpolate the missing one. (a) First green pixel reconstruction kernel. (b) Second green pixel reconstruction kernel. (c) Red pixel reconstruction kernel. (d) Blue pixel reconstruction kernel.

The SGs are used in two different ways to reconstruct the missing pixel in the case of mask2. They are firstly used to enhance the gradient computation in order to detect narrow edges (1-2 pixels width). This step is called gradient improvement. They are also used during the weights computation step to promote weights on the preferred side along an edge. The aim is the same as for the Mask1. It is to evaluate how close neighbors of the missing pixel are similar to pixels in the eight directions. These evaluations allow refining the identification of structures and identifying in which direction there is a continuity of the texture.

The SGs are calculated using the same equations (Equations 3.17 and 3.18) that were presented in the case of Mask1. However, the number of calculated differences may vary, the position of the Z-pixel in the kernel is not constant according to the type of reconstructed pixel (Figure 3.23). For the gradient improvement step, the SGs are directly added to the gradients. Two SGs corresponding to a given direction are added to the gradient corresponding to the same direction. For example, the

north and south SGs are added to the vertical gradient. The improved gradient IG are defined with equations 3.20. During the computation, each IG is scaled according to the number of terms used to calculate each of its components individually. Depending on the position of the Z-pixel, the number of components N to calculate each quantity (gradient and two semi-gradients) can vary, so it is necessary to scale them in order to make comparisons.

$$IG_{vertical} = \frac{Grad_{vertical} + SG_N + SG_S}{N_{Grad_{vertical}} + N_{SG_N} + N_{SG_S}} \tag{3.20a}$$

$$IG_{horizontal} = \frac{Grad_{horizontal} + SG_E + SG_W}{N_{Grad_{horizontal}} + N_{SG_E} + N_{SG_W}} \tag{3.20b}$$

$$IG_{diagonal1} = \frac{Grad_{diagonal1} + SG_{SW} + SG_{NE}}{N_{Grad_{diagonal1}} + N_{SG_{SW}} + N_{SG_{NE}}} \tag{3.20c}$$

$$IG_{diagonal2} = \frac{Grad_{diagonal2} + SG_{SE} + SG_{NW}}{N_{Grad_{diagonal2}} + N_{SG_{SE}} + N_{SG_{NW}}} \tag{3.20d}$$

On the other hand, the SGs are also individually normalized using the number of differences N used to compute them.

$$SG_{scaled} = \frac{SG}{N} \tag{3.21}$$

The SGs and the improved gradients are separately discriminated using the gaussian function 3.11. σ corresponds in both cases to the maximum between the eight improved gradients or the maximum between the four SGs.

Finally the weights are computed as a weighted-sum between two values: the discriminated semi-gradients (DSG) and the discriminated improved gradient (DIG). The weights of the pixels along the edge are mainly defined by the DIG and, conversely, the weights of the pixels located on either side of the edge are mainly defined by the DSG. As we have no information about the missing pixel, the $DSGs$ allow considering local texture on the side of the edge. We compare a given DIG to its orthogonal $DIG_{\perp}$ to determine the edge and then adapt the weight computation.

For example, to compute the weight of the pixel located at the position 2 on the Figure 3.23, we compare $DIG_{vertical}$ with $DIG_{horizontal}$ to identify if the weight needs to be mainly based on the $DIG_{vertical}$ (i.e. there is an edge along the vertical axis) or the DSG_N (i.e. there is an edge along the horizontal axis but the texture of the missing pixel is continue with the North area of the kernel). This allows smoothing the interpolation according to the neighborhood textures and to have a sharper edge. The comparison between the two orthogonal gradients for a given

orientation is defined by:

$$\delta_{DIG/DIG_\perp} = \frac{DIG}{DIG + DIG_\perp} \tag{3.22}$$

Contrary to Mask1, the pixel distribution of Mask2 allows a direct interpolation in the eight directions. The pixel weights are therefore divided into eight distinct areas (Figure 3.24). For a better understanding these areas are represented by the four cardinal points. The weights are computed for each area with the equations 3.24 and apply to the pixels during the interpolation (3.16) according to their distribution between the eight areas.

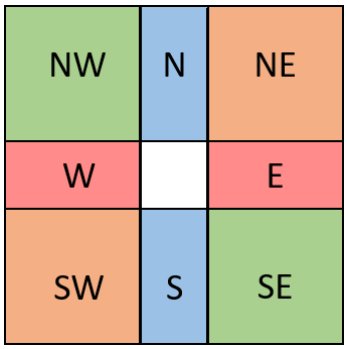
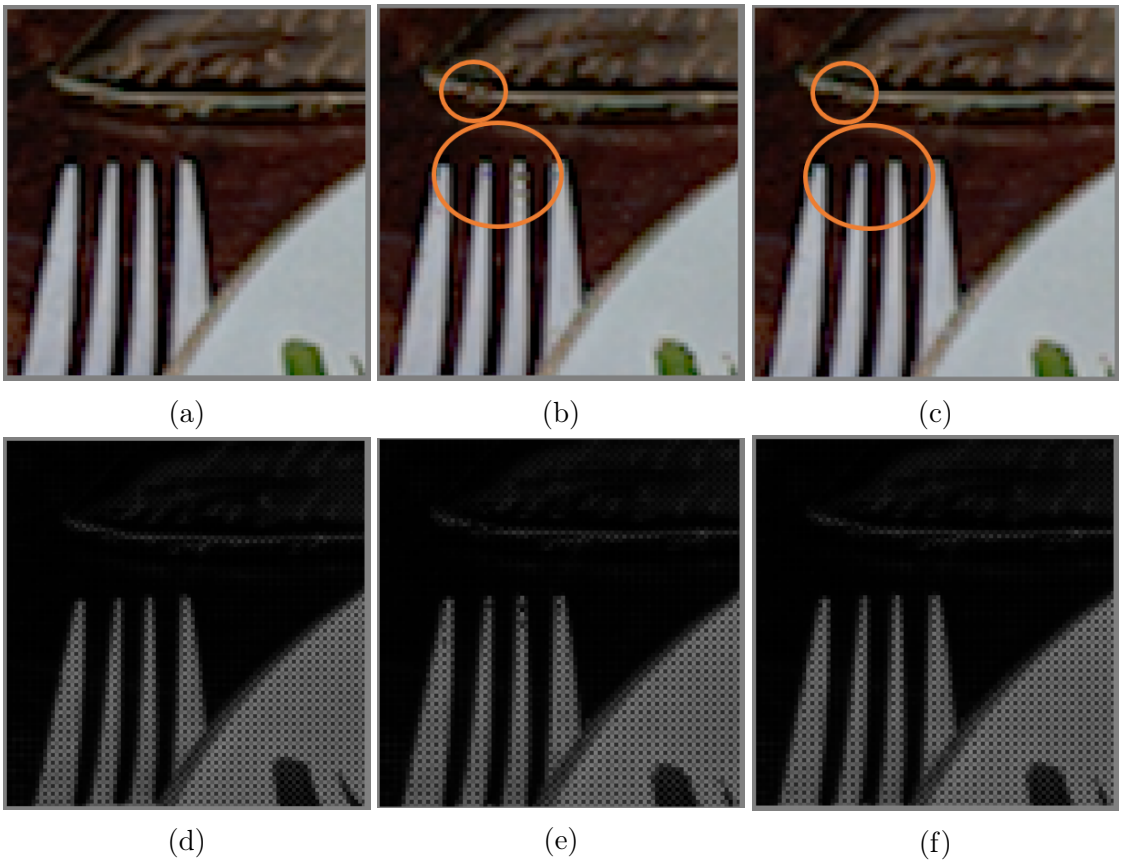


Figure 3.24: Representation of the eight areas. The calculation of weights is based on this cartography.

$$\omega_N = \delta_{vertical/horizontal} \times DIG_{vertical} + (1-\delta) \times DSG_N \tag{3.23a}$$
$$\omega_S = \delta_{vertical/horizontal} \times DIG_{vertical} + (1-\delta) \times DSG_S \tag{3.23b}$$
$$\omega_W = \delta_{horizontal/vertical} \times DIG_{horizontal} + (1-\delta) \times DSG_W \tag{3.23c}$$
$$\omega_E = \delta_{horizontal/vertical} \times DIG_{horizontal} + (1-\delta) \times DSG_E \tag{3.23d}$$
$$\omega_{NW} = \delta_{diagonal1/diagonal2} \times DIG_{diagonal} + (1-\delta) \times DSG_{NW} \tag{3.23e}$$
$$\omega_{SE} = \delta_{diagonal1/diagonal2} \times DIG_{diagonal} + (1-\delta) \times DSG_{SE} \tag{3.23f}$$
$$\omega_{NE} = \delta_{diagonal2/diagonal1} \times DIG_{diagona2} + (1-\delta) \times DSG_{NE} \tag{3.23g}$$
$$\omega_{SW} = \delta_{diagonal2/diagonal1} \times DIG_{diagona2} + (1-\delta) \times DSG_{SW} \tag{3.23h}$$

An example of a reconstructed image using the SG is given in Figure 3.25. It is compared to an image reconstructed with the EDI approach introduced in section 3.4.2. There is a reduction of structural artifacts, especially at the narrow edges. The semi-gradients are used in the case of Mask2 to improve the detection of fine edges, and the reconstruction of missing pixels at the location of these fine edges.

The flowchart of the algorithm is shown in the Figure 3.26 and an other example of reconstructed CFA using the SG for the Mask2 is given in appendix A.4.



(a) (b) (c)

(d) (e) (f)

Figure 3.25: Example of reconstructed Bayer images from the Mask2 and their corresponding demosaiced image. (a) Reference color image. (b) Reconstructed color image using EDI section 3.4.2.2. (c) Reconstructed color image using SGs. (d) Reference CFA that corresponds to the reference color image. (e) Reconstructed CFA using algorithm presented in section 3.4.2.2. (f) Reconstructed CFA using SGs. The orange circles show the reconstruction improvements between EDI and SG.

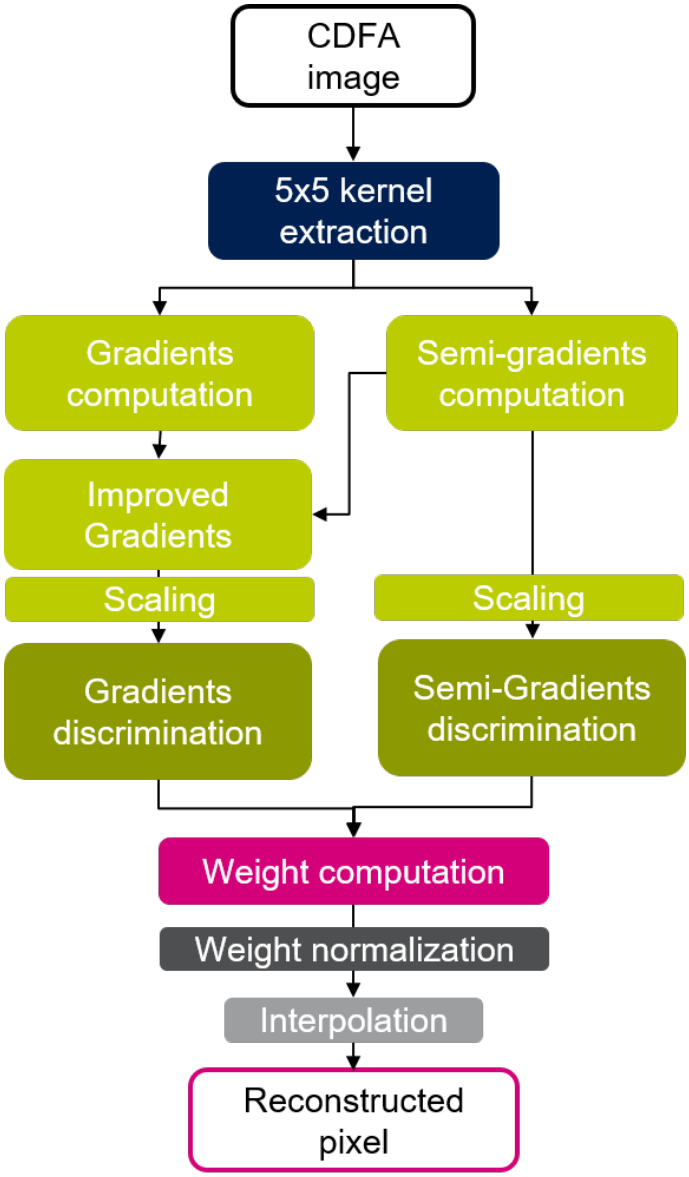


Figure 3.26: Flowchart of a pixel reconstruction with the solution using the SG for the Mask2.

3.4.4 Joint reconstruction (RGB and Z)

A last method of missing information reconstruction has been designed. This method exploits the information from depth pixels while using the semi-gradient method presented in the previous sections. The information used from a Z-pixel at the missing pixel location corresponds to the amplitude information calculated from the four shutters measures (equation 1.5). Concerning the Z-pixels distribution across the pixel matrix, it is interesting to note that the Mask1 and Mask2 are similar. Regarding to the color pixel size, the Z-pixel in Mask2 is twice as large as the Z-pixel in Mask1. However, by considering a kernel twice as large we get the same Z-pixel pattern for both masks. The analogy is represented on the Figure 3.27. The calculation method used to integrate the Z-pixel information is therefore the same for Mask1 and Mask2. In both cases, we consider a 5x5 kernel at the Z pixel level represented in Figure 3.27.b. Following the same idea as the SGs, we study the

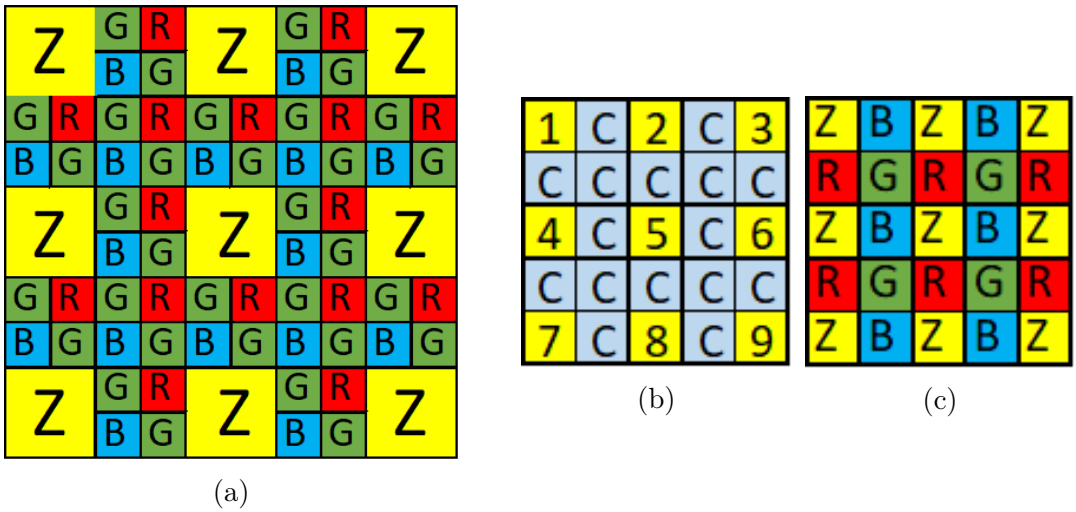


Figure 3.27: Representation of the RGB-Z matrices Mask1 and Mask2 regarding the Z-pixel distribution. (a) 10x10 computational kernel used for the joint reconstruction of the Mask2. (b) Representation of the same kernel at the Z pixel level. It corresponds to a 5x5 Z pixels kernel. Pixels noted "C" corresponds to a cluster of four color pixels. (c) 5x5 computational kernel of the Mask1.

similarity between the intensity levels of the central pixel and its neighborhood. The configuration of the matrix allows to study the eight directions, so eight differences are computed. The same system of cardinal points as presented before is used, and

the pixels indices refer to the Figure 3.27.b.

$$Zdiff_N = |I_5 - I_2| \tag{3.24a}$$
$$Zdiff_{NE} = |I_5 - I_3| \tag{3.24b}$$
$$Zdiff_E = |I_5 - I_6| \tag{3.24c}$$
$$Zdiff_{SE} = |I_5 - I_9| \tag{3.24d}$$
$$Zdiff_S = |I_5 - I_8| \tag{3.24e}$$
$$Zdiff_{SW} = |I_5 - I_7| \tag{3.24f}$$
$$Zdiff_W = |I_5 - I_4| \tag{3.24g}$$
$$Zdiff_{NW} = |I_5 - I_1| \tag{3.24h}$$

These differences are then directly added to the semi-gradient (equations 3.18). In both cases, the differences are used to study the belonging of the missing pixel in each of the eight directions. The last steps of the reconstruction method are the same regarding the proposed method of Mask1 (section 3.4.3.2) and Mask2 (section 3.4.3.3). An example of a joint reconstruction is given in the Figure 3.28. The Middleburry dataset [24] is used for the joint reconstruction. This database provides a depth map corresponding to each color image. It is difficult to notice visual differences between joint reconstruction and the semi-gradient method.

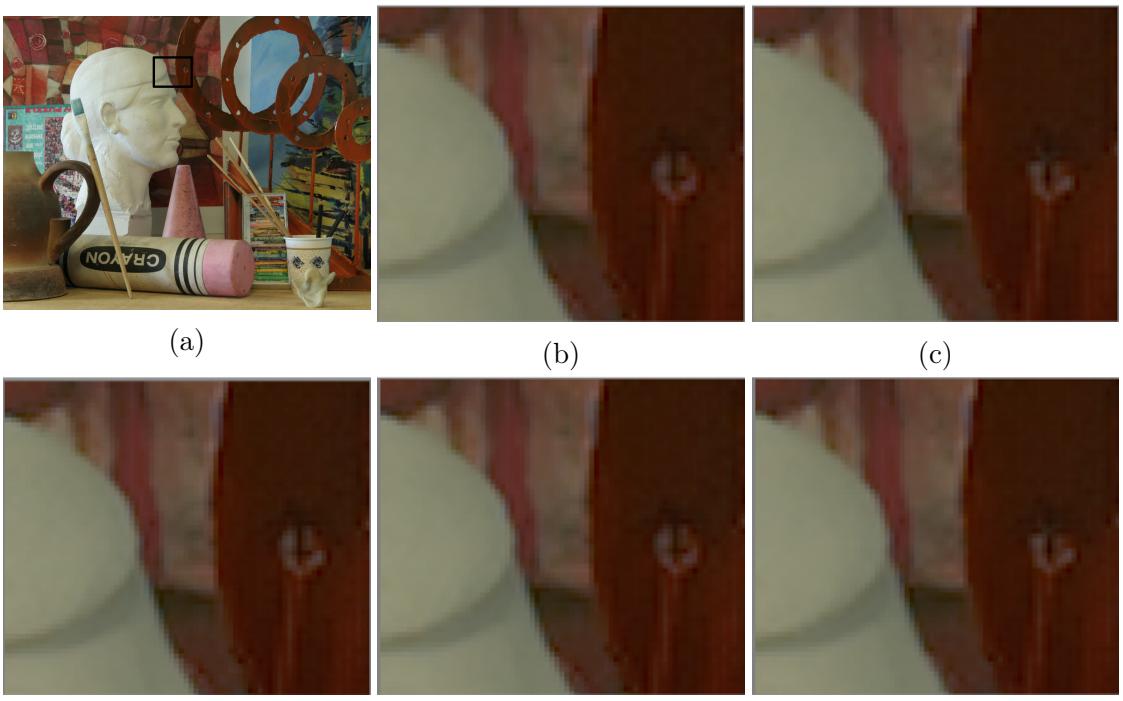


Figure 3.28: Example of reconstructed image using joint reconstruction compared to the reference and the images reconstructed with the method using the SG. (a) Original full color image from Middlebury dataset [23][24]. (b) Image reconstructed using joint reconstruction method from the Mask1. (c) Image reconstructed using joint reconstruction method from the Mask2. (d) Reference image. (e) Image reconstructed using SG method from the Mask1. (f) Image reconstructed using SG method from the Mask2.

3.5 Conclusion

In this chapter different RGB-Z matrices have first been presented. These matrices were designed according to constraints concerning in particular the size of the Z-pixels. Among them, two are based on the Bayer pattern and two others are not. A CFA-independent demosaicing algorithm has been implemented and evaluated. The results of this solution are not satisfactory. This is why we have focused our work on the two RGB-Z architectures based on the Bayer model.

Several reconstruction methods have been implemented to reconstruct the missing pixels at the Z-pixel location. A functional study of both RGB-Z matrices has led to the development of a solution that reduces the artifacts in the reconstructed image. This solution introduces a new operator called Semi-Gradient, which allows a better reconstruction of fine structures.

Finally, a joint reconstruction solution using the information from both the depth and color pixels has been proposed. However, this solution does not give significantly different results than the solution using only the SGs, without any information from depth pixels.

Now that we have studied solutions for reconstructing the missing information for our proposed RGB-Z architectures, we will present the simulation environment that was used, and the results obtained for each solution in more details. Then we will propose a first material implementation of the solution using semi-gradients.

Chapter 4

Methodology and experimental results

4.1 Introduction

This chapter presents the results of the evaluation of the different reconstruction methods presented in the previous chapter. The test environment and the databases used are first presented.

The evaluation of the algorithms was performed in simulation. This allows to compare the reconstructed images with reference images, also generated in simulation. Unfortunately the first real data acquired with a RGB-Z test chip arrived only at the very end of the thesis. However, a first glimpse of this real data is also presented. Then, the reconstruction performances of the different methods applied on the RGB-Z architectures are discussed. Finally a first hardware implementation is performed using the Vivado HLS tools from Xilinx.

4.2 Test environment

The test environment used to evaluate the different algorithms is composed of several elements. A simulation chain, which comprises firstly a sensor model to generate simulated raw RGB-Z images and secondly an ISP model used to reconstruct the full color images. Several databases with different features are used to evaluate the quality of the reconstructed images.

4.2.1 Processing chain

The RGB-Z sensor model used to simulate the images that would have resulted from a real RGB-Z sensor is based on two existing sensor models. The first one, called color sensor model, simulates color pixels, while the second one, called i-ToF sensor

model, simulates both depth pixels and the IR active illumination source. A RGB-Z ISP model is then used to reconstruct the color image from the CDFA raw image. The color sensor model and the i-ToF sensor model are firstly presented to introduce the RGB-Z sensor model. Then, the RGB-Z ISP model based on two ISP models is presented. In the same way as the RGB-Z sensor model, the first ISP model is used to reconstructed the depth map and the second one is used to reconstruct the full color image. The whole RGB-Z simulation chain is detailed in the Figure 4.1.

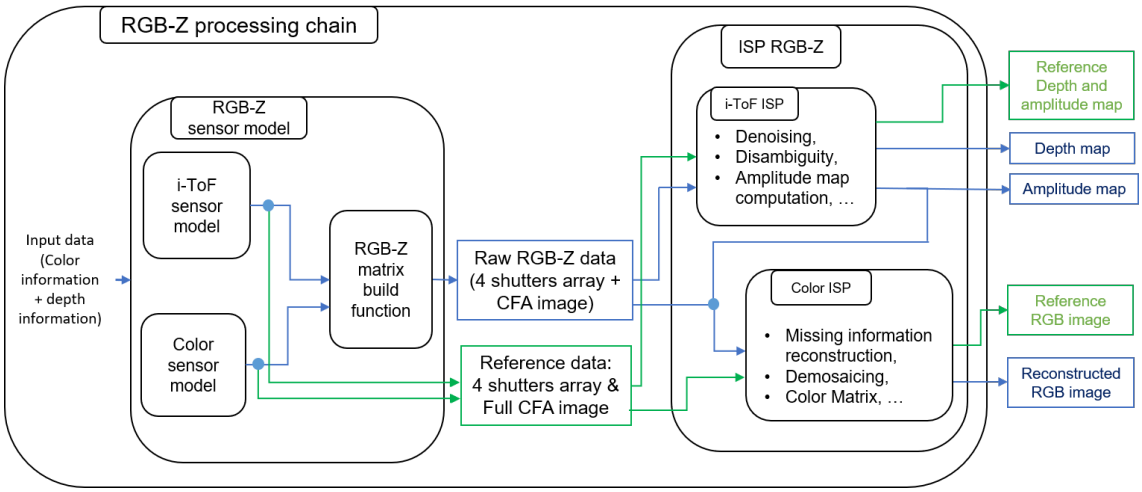


Figure 4.1: Complete RGB-Z chain. The RGB-Z sensor model generates the pixel arrays; the ISP RGB-Z part reconstructs the color image and depth information. The blue path represents RGB-Z data. The green path represents the reference data.

Reference Bayer/RGB and Z images are also generated. This corresponds to the green path in the Figure 4.1. The reference images are used to evaluate the reconstructed images. The step of implementation of the RGB-Z pattern is not applied. In this way the reconstructed missing pixels in the RGB-Z matrix can be compared to the reference ones in the Bayer matrix, which is considered as our ground truth.

4.2.1.1 Color Sensor Model

The color sensor model is used to simulate the acquisition of a CFA image. It is inspired from [102]. It takes as input a sRGB image or a multispectral image set coded on 8 or 16 bits per channel. The output is a raw, noisy image patterned with the selected CFA coded on 11 bits. A noise model [102] is applied to simulate the photon-shot noise and the fix pattern noise. Different characteristics of the pixel and the pixel matrix are taken into account in the model (Section 1.2.1).

The process of the sensor model is as follows. We first compute an average level of generated electrons per channel according to internal and external parameters. Internal parameters of the sensor are, for example, the lens transmission, the pixel size, the integration time and the color and IR filters characteristics. External parameters are represented by the illuminant. The input image is then converted into a electrons map by multiplying the number of generated electrons for each channel with the pixel value of the image, to simulate the scene reflectance at each spatial location. The electron map is an array of floating values that represents a probability distribution of the number of charges accumulated for each pixel. The electrons map is then augmented by the application of different noise sources such as photo shot noise, dark current, or fixed pattern noise before being converted into the raw CFA image output coded on 11 bits.

4.2.1.2 i-ToF Sensor Model

The sensor model that is used to simulate the acquisition of an i-ToF sensor takes as input a depth map and the maximum theoretical distance corresponding to this map. The simulated Z-pixel is composed of 4 shutters (taps) that allow the acquisition of photons out of phase with a phase delay of 90 degrees from each other. In the same way as the Color Sensor Model, the i-ToF Sensor Model takes into account internal and external parameters to compute a multidimensional array that represents the probability distribution of generated electrons for each shutter. Information about the active laser source, such as wavelength, modulation frequency and power are external parameters of this model. It is also necessary to have information about the IR reflectance of the objects in the scene. To my knowledge, this is a data that is not available in depth image databases. This is why several approximations are made to simulate a reflectance map in the infrared. A first simple idea is to consider a constant infra-red reflectance for the whole scene. A second one is to say that the reflectance in infrared and in red are close enough to be confused. This is at least partly false. A known example are the leaves of green plants. They reflect a lot in the infrared and little in the red. Finally, we can also make the assumption that the luminance is close to the reflectance in the infrared. As the example of the leaf for red reflectance, this is also certainly false in some cases.

In the same way as the color sensor model, an averaged level of generated electrons is computed based on the active illumination and the ambient illuminant chosen. The four shutters of each pixel are then sequentially accumulated according to the modulation frequency of the active source. The electrons maps of the four shutters are then augmented by the application of different noise sources such as photon shot noise, dark current, or fixed pattern noise. Finally they are converted

into an output array coded on 11 bits.

4.2.1.3 RGB-Z Sensor Model

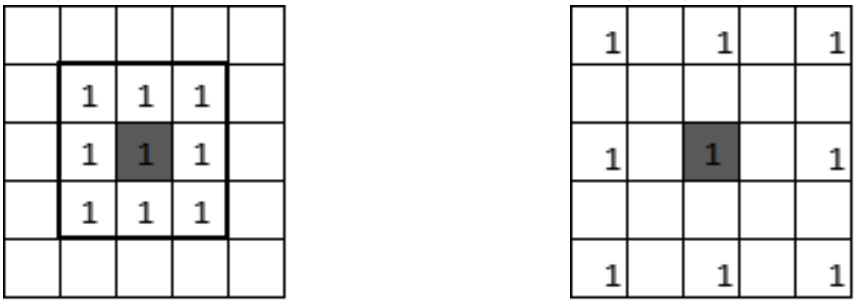
A RGB-Z sensor model that combines the two previous sensor models is used to generate the raw RGB-Z data. Its schematic diagram is represented in figure 4.1. Both color and i-ToF sensor models are used in parallel to simulate a raw RGB-Z image. However it is necessary to make several acquisitions for the i-ToF sensor model in order to generate a depth map. Indeed, a set of three frequencies is used in order to avoid phase wrapping phenomena and to improve the maximum measurement distance (section 1.3.3). Once the CFA image and the shutter array of electrons are generated, they are mixed together to form a RGB-Z matrix. The color information at the Z-pixel location is therefore removed and vice versa. It is called the *RGB-Z matrix build function*. That is during this step that the studied RGB-Z patterns are realized.

In addition, in the case of a RGB-Z sensor, there is the active IR source which allows to make depth measurements. It is important to take into account this potential pollution in the color pixels, despite the IR cut filter on the color pixels.

4.2.1.4 RGB-Z ISP model

Once the RGB-Z data are simulated, the ISP models reconstruct depth and amplitude information from the Z-pixels on one side, and the full color image from color pixels on the other side. Some modifications are made to both ISP chains in order to correctly process RGB-Z data.

Concerning the ISP i-ToF, the filter kernel used for the denoising step is adapted to the RGB-Z matrix. The original filter was a 3x3 average filter. It is replaced by an average filter that takes into account the architecture of the matrix (Figure 3.27). It is now computed with a 5x5 kernel with the distribution represented in Figure 4.2.



(a) (b)

Figure 4.2: Filter kernel used for noise reduction during ISP i-ToF step. (a) Original kernel. (b) Kernel used for the RGB-Z matrix.

The other functions of the ISP i-ToF are not modified: the depth calculation is done only on the depth pixels. The missing depth information at the color pixels location is not reconstructed (section 3.2.1.2). In the joint reconstruction context (section 3.4.4), we use amplitude information from the Z-pixel data. Amplitude information must then be reconstructed first using the i-ToF ISP functions, in order to use it during the missing color information reconstruction.

The color ISP is also modified in order to reconstruct missing color information. The algorithms presented in the previous chapter are included in the processing chain. However, this reconstruction step should not be placed anywhere in the chain. Indeed, it is imperative to reconstruct the missing information before the demosaicing step. In the same way, as we use an automatic white balance, it is preferable to have a complete color pixel matrix available for this algorithm. On the other hand, in order to avoid using defect pixels for the color pixel reconstruction, we apply the defect correction algorithms before the reconstruction of the missing color pixels. The modified ISP chain is shown in Figure 4.3.

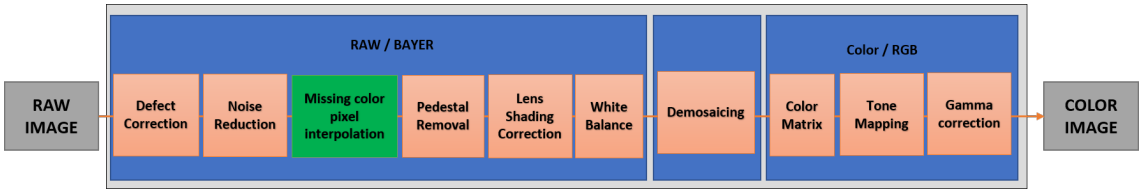


Figure 4.3: Typical ISP processing chain adapted to a RGB-Z sensor.

4.2.2 Databases

In this section we present the different databases used to evaluate the missing information reconstruction algorithms implemented. It is important to evaluate the algorithms on many images with various type of content. It allows to reduce the effect that a particular image could have. To this intent, we use databases with varying characteristics. For example, different resolutions are studied, different frequency contents.

Kodak [11], McMaster [25] and HDR+ [26] burst are used to study color reconstruction without the use of depth pixel information. These three databases have different frequency content and resolutions. CAVE [27] and Okutomi [20] allow the use of multispectral data, this allows to study the impact of the infrared on the colorimetry. Finally Middlebury [23][24] allows to study the methods of joint reconstruction. They are summarized in the Table 4.1.

4.2.2.1 Kodak and McMaster datasets

The Kodak database [11] is wildly used in demosaicing papers. It contains twenty-four color images, of spatial size 768×512, which have been digitized by scanner after being captured on film. This dataset was criticized for having statistics very different from the ones of natural images [25]. This led to the creation of the McMaster dataset, which contains eighteen 500×500 crops of 2310×1814 images. The high resolution of the original McMaster images is an advantage of McMaster over Kodak, as most image sensors have a high resolution too. In comparison, they have more saturated colors than Kodak and contain many sharp structures with abrupt color transitions. This can be seen in the images given as examples in Figure 4.4 and 4.5.

Figure 4.4: Four examples of color images from Kodak database [11].

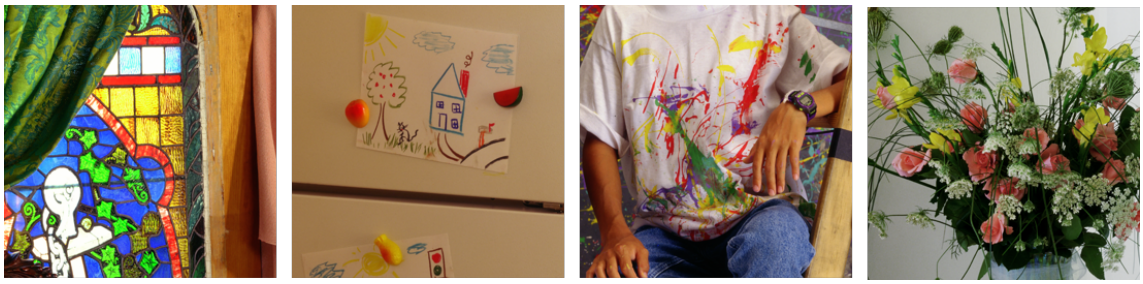


Figure 4.5: Four examples of color images from McMaster database [25].

4.2.2.2 HDR+ burst dataset

In addition to the McMaster and Kodak databases, the HDR+ burst dataset [26] constituted of real images acquired by CFA sensors is used. It consists of 3640 images acquired with different CMOS imaging sensors from Android mobiles. This database is built to study HDR reconstruction techniques. The images are generally 12-13 Mpixels, depending on the type of camera used for capture. Not all the images in the database are used in this work. A subjective selection of twelve images is made (appendix A.5). In addition, the selected images are downsampled by a

factor 2 horizontally and vertically in order to improve the computational time. Furthermore, by downsampling the images, we also reduce the distance between structures, thus, we increase the frequency content in the image. Since the most complex structures to reconstruct are located in high frequencies areas, it is also interesting for the assessment of the reconstruction algorithms.

Figure 4.6: Three examples of color images from HDR+ burst database [26].

4.2.2.3 CAVE and Okutomi datasets

Two multi-spectral databases are used: the CAVE database [27] and the Okutomi [20] database. It allows to study the influence of the IR on the colorimetry. In both cases, the image size is 512×512. The CAVE dataset consists of 31-band multispectral images of thirty-two scenes. The 31-band images are acquired at every 10 nm from 400 nm to 700 nm. In the CAVE dataset, many images are dominated by smooth objects or backgrounds.

Figure 4.7: Three examples of color images from CAVE database [27].

The Okutomi dataset [20] consists of 59-band multispectral images of forty scenes. The 59-band images are acquired from 420 nm to 1 000 nm at every 10 nm. We know the infrared reflectance of the scene for wavelengths up to 1 000 nm. The hyperspectral images are converted into a form of spectral reflectance by a calibration process using a colorchecker whose spectral reflectance property is known.

Infrared reflectance information is available with this database, however there is still no depth information or disparity map available. In the same way as the McMaster dataset, the Okutomi images are cropped from high resolution images. The selected regions contain rich textures and colored objects, which is more challenging for missing information reconstruction applications.

Figure 4.8: Three examples of color images from Okutomi database [20].

4.2.2.4 Middlebury dataset

The 2005 Middlebury database [23][24] provides a set of six[1] scenes. For each scene, a disparity map and the corresponding color image are provided. However the infrared reflectance information is not available. The original images are about 1300×1100 pixels. However, in the same way as the HDR+ burst data set, the images from the Middlebury data set are also downsampled in order to have an higher frequency content. Reducing the resolution of images can also reduce the number of pixels between two edges, and thus artificially create narrow edges.

4.3 Reconstruction evaluation

In this section, we present the results obtained with the algorithms presented in Chapter 3. Reconstructed images are compared to the corresponding reference image at two different levels of the processing chain presented in Section 4.2.1. Both reference and reconstructed images are generated using the same processing chain. In this way, only the reconstruction step of the missing information is evaluated. The parameters of the simulation of the raw RGB-Z and raw RGB images are identical, only the generation of the RGB-Z pattern differs. In the same way for the reconstruction of the two color images, only the reconstruction step of the missing

[1] 9 different scenes are available, but only 6 different scenes have a disparity map.

Figure 4.9: Four examples of color images and their associated disparity map from Middlebury database [23][24].

Database	Year	Total Images	Number of images used	Image resolution (used)	Multi-spectral acquisition	Depth map
Kodak [11]	1991	24	24	768×512	No	No
McMaster [25]	2011	18	18	500×500	No	No
HDR+ burst [26]	2016	3640	12	4000×3000 (2000×1500)	No	No
CAVE [27]	2008	32	32	512×512	Yes	No
Okutomi [20]	2015	16	16	500×500	Yes	No
Middleburry [23][24]	2007	6	6	1330×1110 (463×370)	No	Yes

Table 4.1: Overview of databases used in this work.

information is added compared to the reference image. Characteristics such as noise level, integration time, illuminant, pixels quantum efficiency, color and IR filters characteristics are common to both images. Only the RGB-Z matrix construction function and the missing information reconstruction algorithm are bypassed for the reference image. The first level of assessment is applied on the reconstructed CFA images. The PSNR is computed on the reconstructed color pixels only. The second level of assessment is then applied at the end of the ISP chain, on the reconstructed full color image. The C-PSNR, the SSIM and M-SSIM, and the Zipper metric presented in the chapter 2.2 are used to quantify the reconstruction performance. Figure 4.10 illustrates the two ISP chains executed on the reference image and the reconstructed RGB-Z image.

The experimental results are presented according to the RGB-Z pattern considered. The results of the reconstruction methods applied to the Mask1 are first presented, then the results of the images reconstruction of the Mask2 are analyzed.

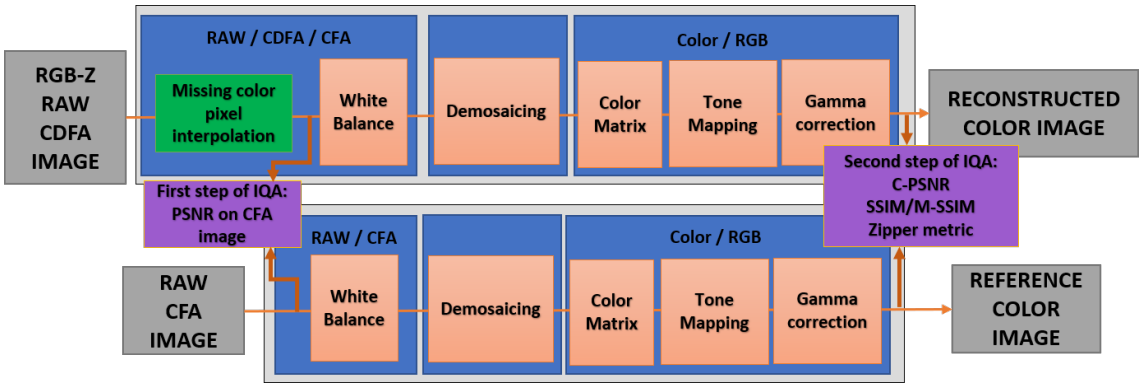


Figure 4.10: Diagram of two simplified ISPs, one for the reconstruction of the RGB-Z image (top) and the other for the reconstruction of the color image (bottom). In purple are indicated the two steps of IQA.

The joint reconstruction is also studied, for the databases containing depth information, in order to be able to properly simulate the Z-pixels.

In the following sections, the color pixel reconstruction algorithms are abbreviated. The bilateral based algorithm (Section 3.4.1) is indicated as *BI*. The edge directed algorithm calculating the gradients by a central symmetry around the missing pixel (Section 3.4.2.1) is indicated as *EDICentral*. The edge directed algorithm calculating the gradients along the kernel axes s (Section 3.4.2.2) is indicated as *EDILine*. Finally, the solution using the Semi-gradients (Section 3.4.3) is indicated as *SG*.

4.3.1 Quality assessment of the Mask1

The methodology presented in the previous sections is applied to evaluate the reconstruction quality for the Mask1 (Figure 4.11).

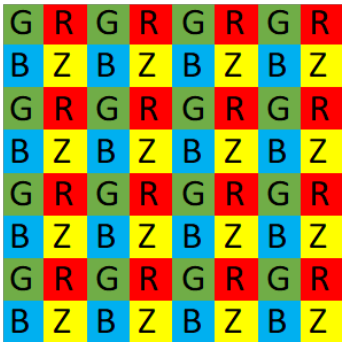


Figure 4.11: Representation of the Mask1: RGB-Z matrix based on the Bayer pattern built using the 1x1-Z-pixel.

The results for the Kodak, McMaster and HDR+ burst databases are presented

in the Figure 4.13. The details of the results for each computed image are given in appendix A.6, appendix A.7 and A.10 respectively. We notice that the standard deviation for a given method is relatively high compared to the differences between the four methods. This is mainly due to the variations of the content between the images in the same database. However, it is interesting to note a general trend: the solution using the semi-gradients *SG* has better performances than the *EDILine*, which has better performances than the *EDICentral*. We can also notice that *BI* and *EDICentral* solutions have very similar results. This is mainly due to the fact that the distribution of green pixels useful for interpolation does not allow direct interpolation in all directions. Therefore, despite the use of edge directed methods, it is common that the weights distribution is homogeneous between the four pixels. In this case, the EDI interpolation is similar to the *BI* interpolation. The semi-gradients are designed to compensate this limitation, to improve the weights computation and, consequently, the reconstruction of the missing pixel. On average, the semi-gradient *SG* method improves the PSNR of the reconstructed pixels in the CFA image by 1 dB compared to *EDILine*. Similarly, the overall C-PSNR on the color image is improved when the reconstruction of the missing pixels is performed by the semi-gradient method.

4.3.2 Quality assessment of the Mask2

The same evaluation as in the previous section is performed for Mask2 (Figure 4.12). The results are summarized in the Figure 4.13 for the Kodak, McMaster and HDR+ burst databases. The results are detailed in the appendix A.8, the appendix A.9 and the appendix A.11.

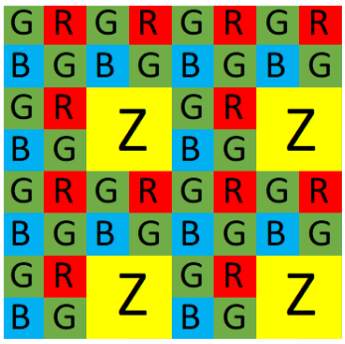


Figure 4.12: Representation of the Mask2: RGB-Z matrix based on the Bayer pattern built using the 2x2-Z-pixel.

In the same way as the Mask1, the general trend is similar: the semi-gradient based method shows the better performances. However, we can note that the metrics scores applied on the color images obtained for Mask2 are in average lower than the

scores obtained for Mask1, while the PSNR computed on the reconstructed pixels using the CFA images are closer. This suggests that is inherently more difficult to fully reconstruct a color image from a RGB-Z CDFA image based on the Mask2 compared to the Mask1. The semi-gradient based algorithm obtains, on average, a PSNR score of 30.60 dB when used to reconstruct a RGB-Z architecture based on the Mask1, while it obtains a score of 30.20 dB for the Mask2. Those results are relatively close compared to the difference in term of C-PSNR between the two Masks. On average the C-PSNR score of the semi-gradient based algorithm for the Mask1 is 36.61 dB whereas it is 33.95 dB for the Mask2.

One possible explanation concerns the demosaicing step. In the case of 2x2 Z-Pixel mask, the missing information represents a cluster of 2x2 color pixels at the Z-pixel location. An incorrect reconstruction of the four color pixels corresponding to a Z-pixel in the 2x2 Z-Pixel mask introduces an error that could strongly impact the demosaicing step. The Z-pixel pattern becomes visible. This effect is particularly visible on the images reconstructed with the *BI* algorithm.

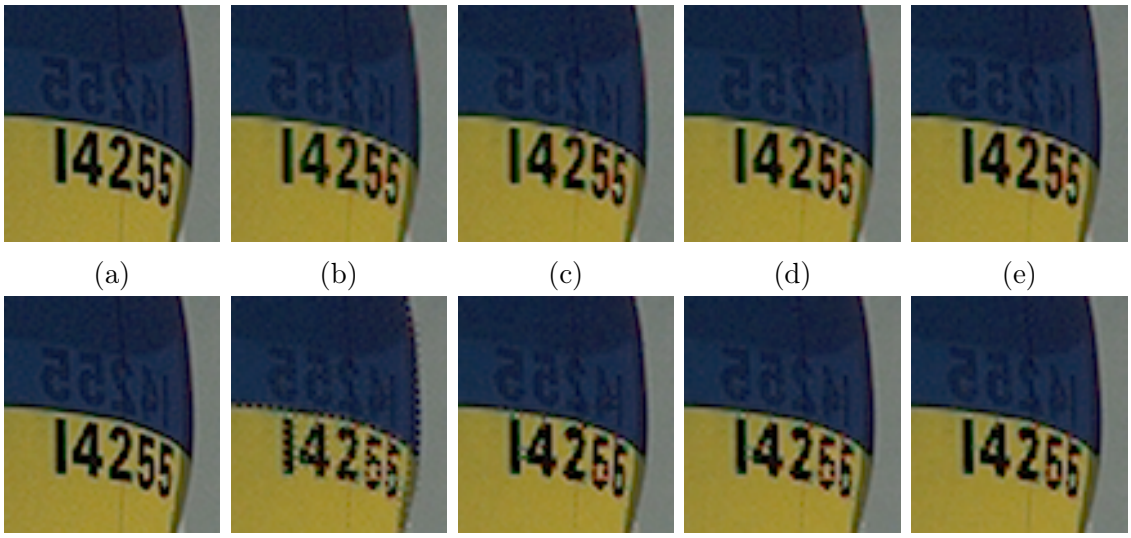
An example is given in the Figure 4.14. The corresponding metrics results processed are given in the Table 4.2. The Z-pixel pattern is apparent when we compare the two reconstructed images using the Bilateral interpolation (Figure 4.14.b and Figure 4.14.f). Indeed, the artifacts corresponding to the missing information at the Z-pixels location are directly identifiable on the edges of the image in the case of Mask2. They remain visible with the *EDICentral*, *EDIline* and the *SG* solutions, but they are significantly reduced. The objective evaluation of the different methods has the same trend as the subjective analysis of the reconstructed images. In both cases, it is reported that the solution using semi-gradients reduces the artifacts in the final reconstructed image.

4.3.3 Joint reconstruction evaluation

The joint reconstruction method presented in Chapter 3.4.4 is implemented and compared to semi-gradient based methods using only color information. The objective is to evaluate the impact of using the Z-pixel information to reconstruct the missing color information. For this study, we use the Middlebury database [23][24] (section 4.2.2.4) in order to have depth information and thus to be able to simulate the Z pixels as well as possible. However, we do not have information about the reflectance in the infrared. In order to overcome this issue, we have firstly made the assumption that the IR reflectance is constant in the scene. We set it to 0.4, which means that 40% of the photons in the infrared domain that hit an object are reflected. In Figure 4.15 are illustrated a simulated color image and its associated Z-pixel amplitude map and reconstructed depth map. The simulated amplitude

Figure 4.13: Averaged performances of reconstruction methods in terms of (a) PSNR applied on the CFA image and (b) C-PSNR, (c) SSIM, (d) M-SSIM, and (e) Zipper Metric applied on the color image. Results are computed on the Kodak database [11], the McMaster database [25] and the HDR+ burst [26] for the Mask1 and the Mask2.

map with homogeneous IR reflectance does not have as much structural detail as the color image. As the amplitude map depends only on the depth in this case, the areas located at the same distance are therefore flat. We then made a second assumption that allow to have an heterogeneous map for IR reflectance. We use the



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

(f) (g) (h) (i) (j)

Figure 4.14: Crop of reconstructed KodaK9 images from the Mask1 and Mask2 using the four reconstruction methods. (a) and (f) Reference image. (b) Bilateral interpolation *BI* on Mask1. (c) EDI central on Mask1. (d) EDI Line on Mask1. (e) Semi-gradient based interpolation *(SG)* on Mask1. (g) Bilateral interpolation *BI* on Mask2. (h) EDI central on Mask2. (i) EDI Line on Mask2. (j) Semi-gradient based interpolation *(SG)* on Mask2.

Mask1	BI	EDICentral	EDILine	SG
PSNR	24.55	24.46	24.97	**26.9**
C-PSNR	30.64	30.81	31.02	**32.99**
SSIM	0.9595	0.9607	0.9636	**0.9751**
M-SSIM	0.9861	0.9861	0.9872	**0.9918**
Zipper metric	12.493	12.087	11.504	**8.591**
Mask2	BI	EDICentral	EDILine	SG
PSNR	22.93	24.83	25.13	**26.13**
C-PSNR	26.70	29.12	29.13	**30.31**
SSIM	0.9163	0.9565	0.9600	**0.9652**
M-SSIM	0.9676	0.9832	0.9835	**0.9868**
Zipper metric	9.810	5.420	5.434	**5.041**

Table 4.2: Results of the metrics for the kodak9 image using the four reconstruction algorithms applied on the Mask1 and the Mask2. The corresponding images are shown in figure 4.14.

red channel of the color image and consider that it corresponds to the IR reflectance. We choose the R channel because compared to the blue or green channel, the red spectrum is closer to the infrared spectrum. This assumption is not true, however, in an experimental setting it allows to obtain an amplitude map with more details, even if these details do not correspond to real IR amplitude map.

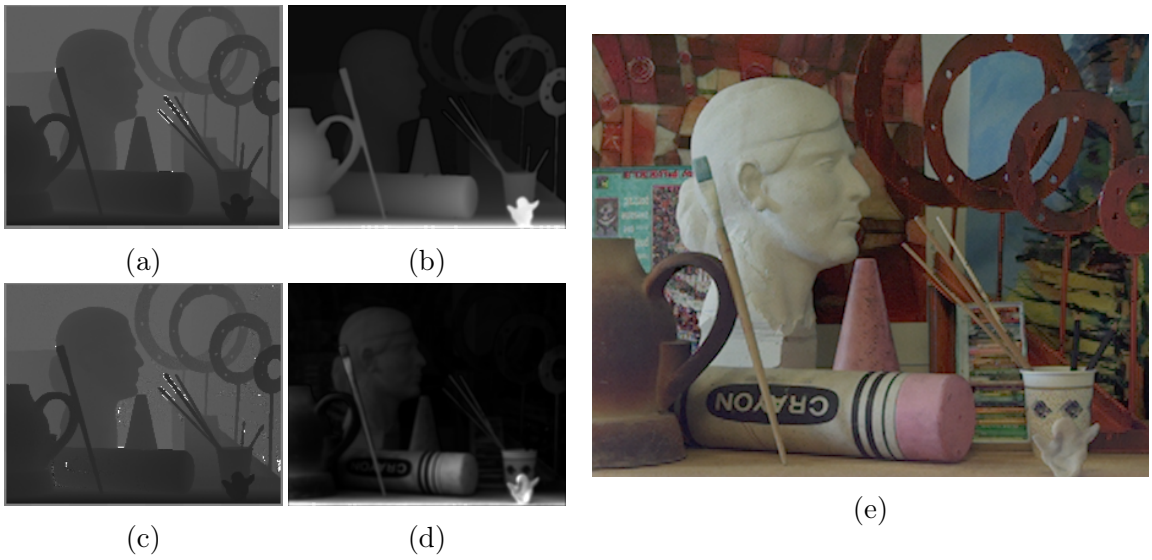
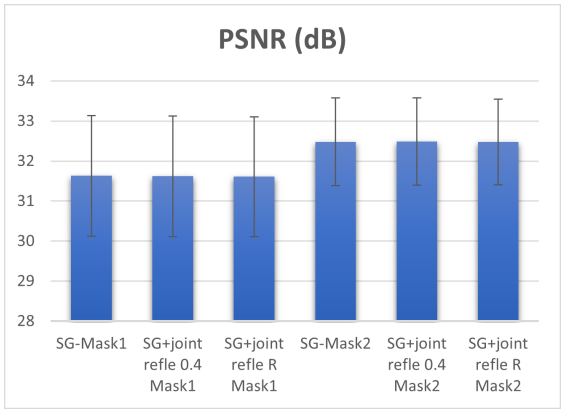


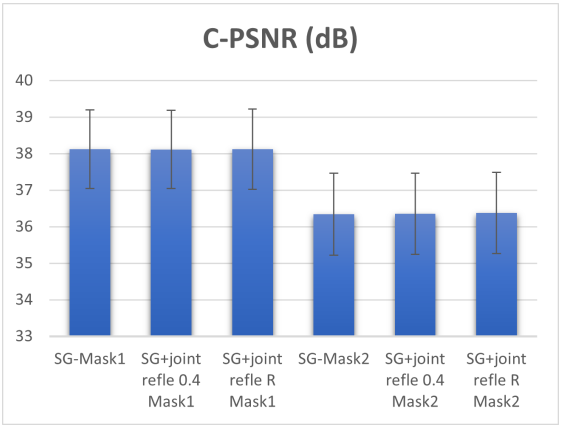
Figure 4.15: Example of simulated image from Middlebury dataset [23][24]. (a) Depth map obtained with a homogeneous IR reflectance set to 0.4. (b) Amplitude map obtained with a homogeneous IR reflectance set to 0.4. (c) Depth map obtained with an IR reflectance map corresponding to the R channel. (d) Amplitude map obtained with an IR reflectance map corresponding to the R channel. (e) Corresponding color image. In this example, the simulation is made using a Mask1. Amplitude and depth maps are downsampled in order to only show the Z-pixels.

The results obtained on the Middlebury database are presented in the figure 4.16. We can not conclude on the interest of using the joint reconstruction as it is defined in section 3.4.4. Despite the use of the red channel as a reflectance map, and thus the improvement of the details in the amplitude map, the reconstruction results are not significantly different from the semi-gradient method working without information from the Z-pixels.

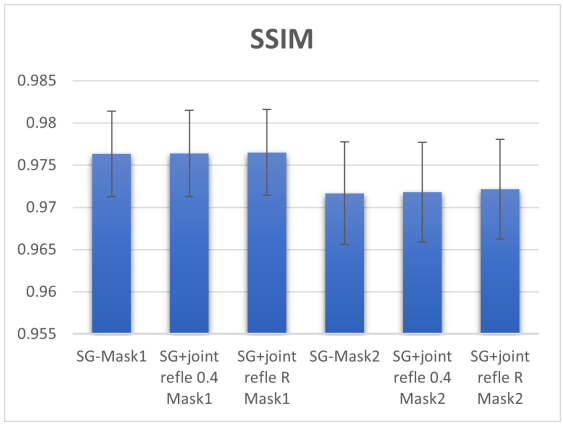
It is important to note that the database used for this study contains few images. Moreover, it is possible that the content of these images does not allow to highlight the interest of the joint reconstruction. A more detailed study, with more specific data, such as high frequency content in both the depth map and the color image, might highlight the benefits of this approach.



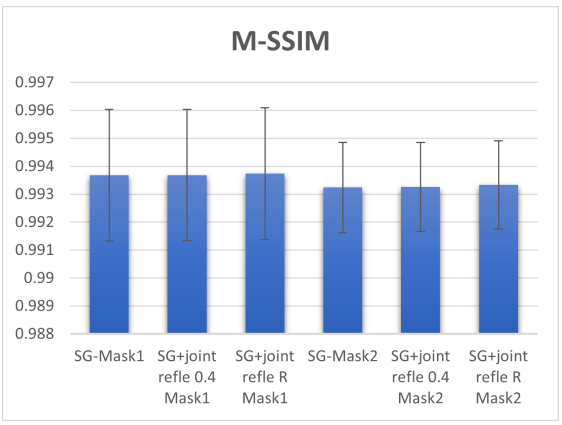
(a)



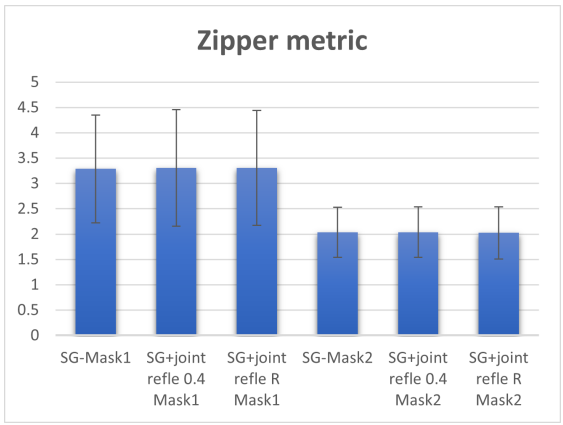
(b)



(c)



(d)



(e)

Figure 4.16: Averaged performances of joint reconstruction compared to semi-gradient based methods for both Mask1 and Mask2 in terms of (a) PSNR applied on the CFA image and (b) C-PSNR, (c) SSIM, (d) M-SSIM, and (e) Zipper Metric applied on the color image. Results are computed on the Middlebury dataset [23][24]. Results per image are detailed in appendix A.12.

4.3.4 Examples of remaining artifacts

We have seen in the previous sections that the use of semi-gradients allows to improve the quality of reconstructed images in the case of the studied RGB-Z architectures. However, even with the semi-gradients, some structures are not correctly reconstructed.

4.3.4.1 Mask1

Regarding the Mask1, for the same reasons explained in the section 3.4.3.1, it is difficult to reconstruct vertical and horizontal edges due to the absence of green pixel along these axes. Semi-gradients allow to overcome these limitations for some cases by analyzing the close neighborhood of the missing pixel in order to determine to which texture it belongs. This method works as long as the missing pixel is close to a neighboring texture. However, there are situations with sharp narrow edges with a one pixel width. In these situations, it may happen that the green pixels available for interpolating the missing pixel do not match the value of the missing pixel. A weighted sum of these pixels cannot then correctly reconstruct the missing pixel. The Figure 4.17 shows an example of a narrow edge that can not be properly reconstructed using semi-gradients.

In this example, we can notice that the semi-gradient based interpolation promotes the use of the green pixels located on the left side of the kernel. Indeed, the red pixels values neighboring the missing pixel are closer to the red pixels values located on the left side of the kernel. An example of erroneous reconstruction along very thin edges is given in Figure 4.18.

4.3.4.2 Mask2

Corners are structures that are difficult to reconstruct in the Mask2 architecture. In particular, there are reconstruction errors when a Z-pixel is precisely located on the corner. In a such case, there are often only two textures present in the 5x5 computation kernel: a first one that corresponds to the corner and the second one that corresponds to the background. An example is given on the Figure 4.19. The color pixels present on the bottom left edges of the calculation kernel belong to the background texture, and the missing color pixels belong to the corner texture that is located in the top right area. Background pixels are shaded on the figure. The two red pixels located on the northwest/southeast diagonal belong to the background texture. The gradient and semi-gradients along this axis are small and imply an interpolation along this one, while the interpolation should be along the southwest/northeast axis with a privileged interpolation along the north east corner of the kernel thanks to the semi-gradient. A known information from the Z

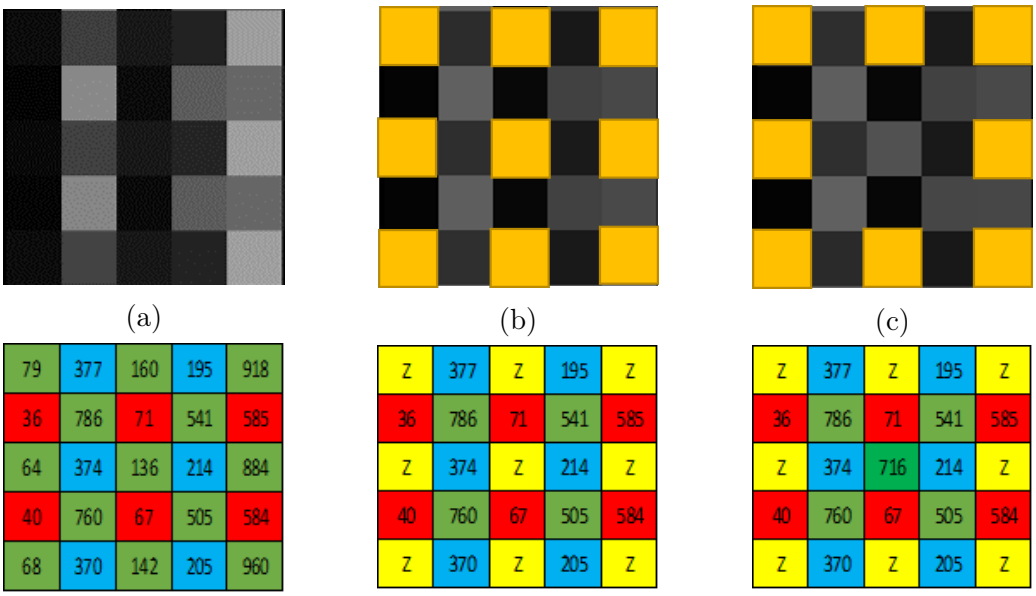


Figure 4.17: Example of a narrow edge reconstruction error for Mask1 using the semi-gradient method. a) CFA reference and its corresponding values (d). (b) CDFA and its corresponding values (e). (c) CDFA after the interpolation of the missing pixel and its corresponding values (f). The real value of the missing pixel is 136 while the four green pixels in the neighborhood are all greater than 500. It is impossible to correctly interpolate the missing pixel value using a weighted sum of the four neighboring green pixels. The result of the interpolation is 716. The pixel values are coded on 11 bits. The yellow squares indicate the Z-pixels.

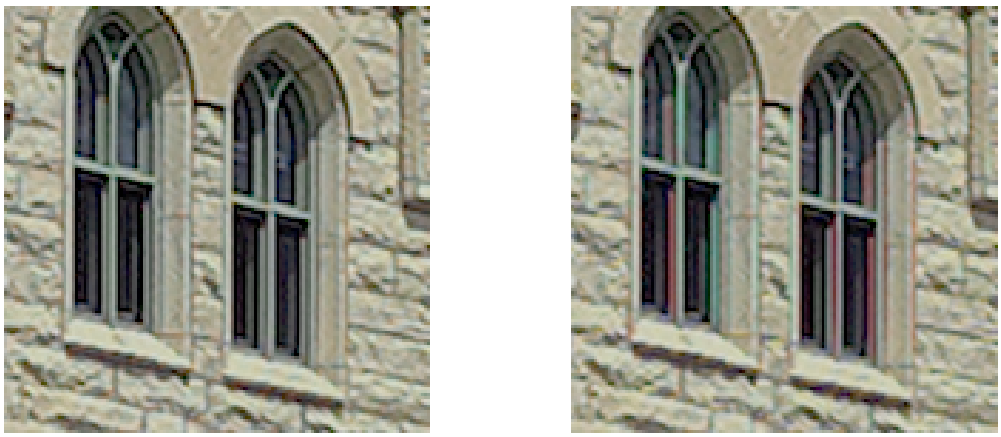


Figure 4.18: Example of reconstruction errors along very thin edges. (a) Reference image. (b) Reconstructed image of the Mask1 using semi-gradient method. The red and green vertical stripes along the window are characteristic of too thin edges. The image is a crop from the HDR+ burst database [26].

pixel at the missing pixel location could help to detect the corner properly and thus improve its reconstruction. Studied on real data, or with a relevant database, joint reconstruction may be an opportunity of improvement in the case of reconstruction

based on the Mask2. An example of a reconstruction error of a corner is given in the Figure 4.20.

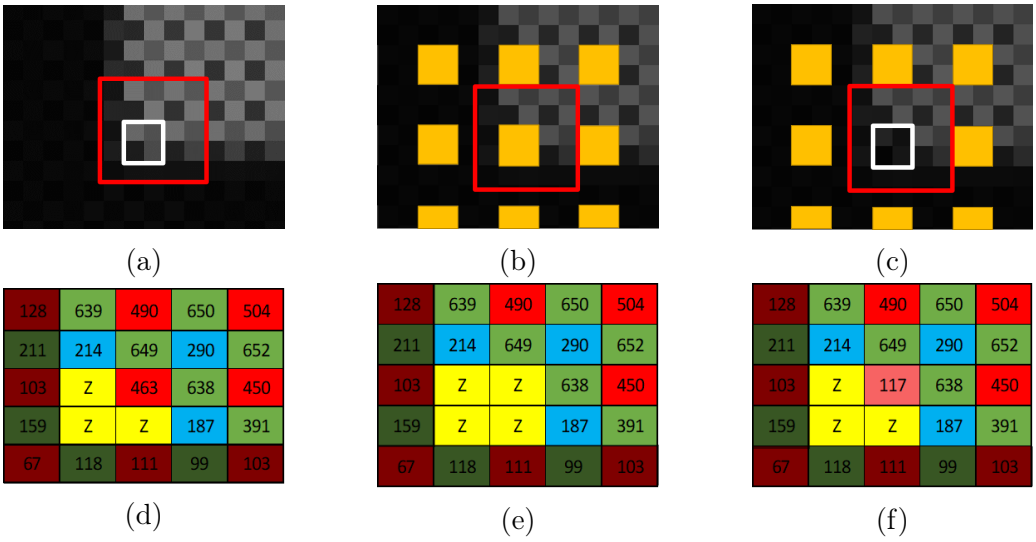


Figure 4.19: Example corner reconstruction error for Mask2 using the semi-gradient method. The yellow squares indicate the Z-pixels, the white squares indicate the location of the four reconstructed color pixels, and the red square indicates the computational kernel to reconstruct the missing red pixel show as example(a) CFA reference and its corresponding values (d). (b) CDFA and its corresponding values (e). (c) CDFA after the interpolation of the missing pixel and its corresponding values (f). The shaded area represents the texture corresponding to the background. The pixel values are coded on 11 bits. The interpolation is made using the semi-gradients method.

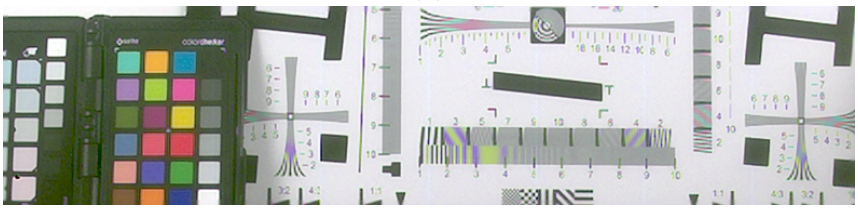
Figure 4.20: Example of reconstruction errors of a corner for Mask2 architecture. (a) Reference image. (b) Reconstructed image using semi-gradient method. The image is a crop from the CAVE database [27].

4.3.5 First real RGB-Z data

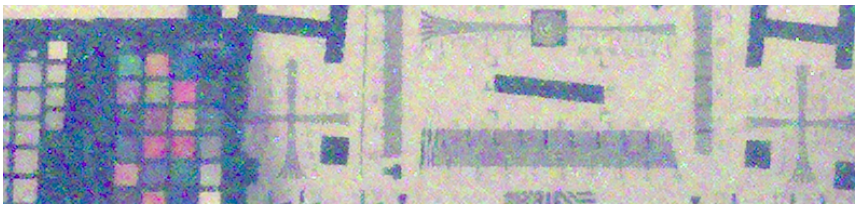
A first sample of four images captured with a first RGB-Z test chip were provided during the last weeks of the thesis. The raw sensor data was provided to us by the CEA[2]. These images are based on the Mask1 but use a pixel size of 3.6 µm^2. The images are shown in the Figure 4.21. However, No depth information is acquired by the Z-pixels yet in these first tests. Only color information is provided. The joint reconstruction could thus not be evaluated on this samples. Moreover, IR pollution could not be studied because the illuminant used for the captures of these first samples is a white LED that does not contain IR.

The RGB-Z ISP implemented for the simulations is used to reconstruct these images. In absence of ground truth it is difficult to make an objective analysis of the result.

(a)



(b)



(c)

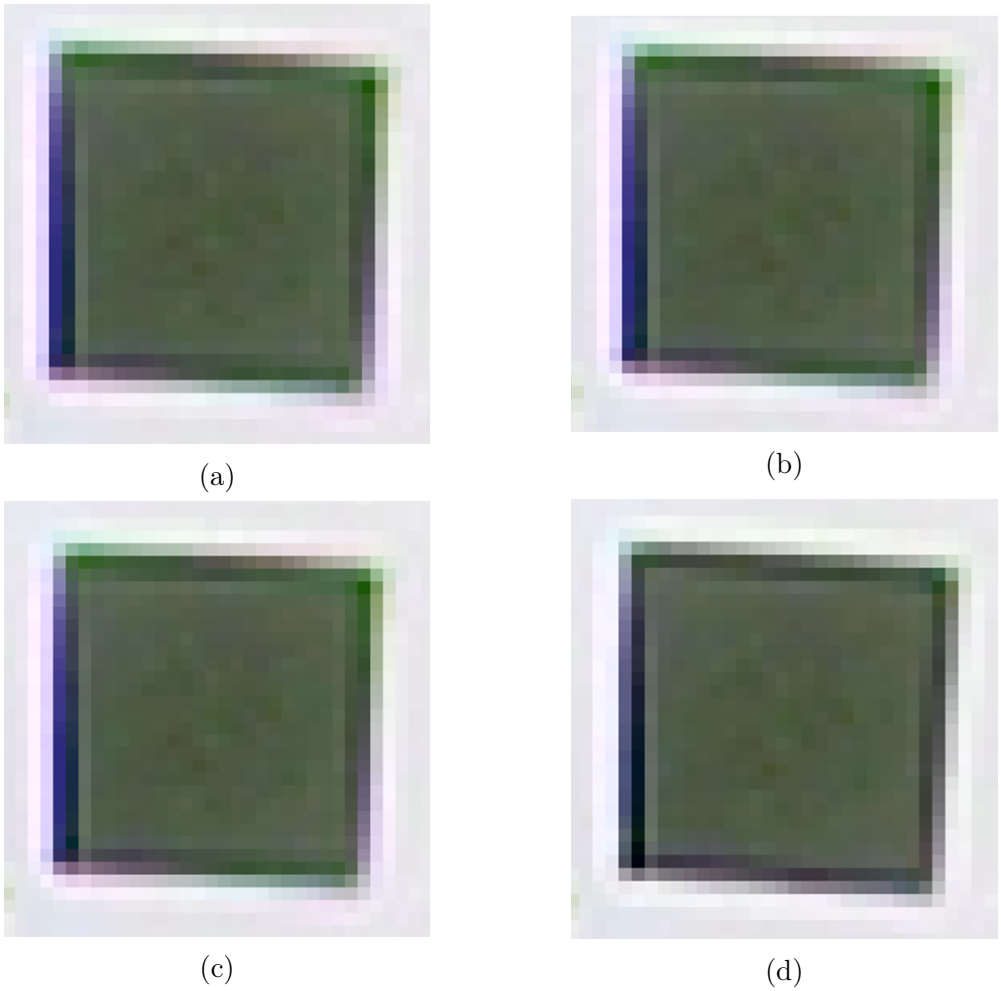
(d)

Figure 4.21: The four real data reconstructed with the solution using the semi-gradients. The image (c) is acquired in very low light conditions.

The focus of the images is not optimal, which makes the images slightly blurred.

[2]https://www.leti-cea.fr/cea-tech/leti/Pages/Accueil.aspx

This blur does not allow to observe correctly the reconstruction of the contours. However, on Figure 4.22 we can note, by subjective analysis, a color artifacts reduction and some contours are sharper with the semi-gradient solution.



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

Figure 4.22: Comparison of reconstruction of different methods. (a) BI. (b) EDI central. (c) EDI Line. (d) Semi-gradient based interpolation. Crop from image (b) of the Figure 4.21.

4.4 Complexity study

In this section we propose to study the hardware implementation of the missing information reconstruction algorithm proposed in section 3.4.3. In order to integrate the proposed processing within an ISP chain, it is essential to have an estimate of the complexity, latency and resources of the algorithm. A first prototyping has been done in this sense. We choose to implement the solution proposed for Mask2 because it is more complex than the solution used for Mask1. Indeed, in the case of Mask2, gradients are also used. This first solution allowed to identify weaknesses, especially in terms of latency. First suggestions for acceleration are considered.

First, we briefly present the High Level Synthesis (HLS) tool, which is used to describe the hardware solution. Then we describe the fixed point implementation of some functions of the algorithm. Finally, we present the first simulation results and optimization suggestions.

4.4.1 HLS

The HLS tools allow to interpret algorithmic descriptions of high-level programming languages (C, C++ or SystemC) in order to create Register-Transfer Level (RTL) implementations in a low-level Hardware Description Language (HDL) (VHDL or verilog). HLS reduces the implementation efforts and debugging of the HDL design while allowing flexibility in implementation through the use of directives. In our case, we use the VivadoHLS tool provided by Xilinx[3]. VivadoHLS compiles C input languages into RTL, which can then be synthesized and implemented on an FPGA target. The design flow of a material design with VivadeoHLS is done in three steps:

- C simulation to validate that the C functions are correct according to the test bench.
- C Synthesis. C/C++ source code is synthesized into a RTL implementation.
- C/RTL co-simulation to verify that the RTL is functionally identical to the C source.

An overview of the VivadoHLS input files and output files are shown in Figure 4.23. They are listed :

- Inputs:
  - C/C++ programs (algorithmic functions.)
  - A C-test bench and test files.
  - Constraints that include the period and the clock uncertainty.
  - Pragmas are used to optimize the design. They reduce latency, improve throughput performance, and reduce area and device resource usage of the resulting RTL code.
- Outputs:
  - The RTL implementation written in HDL (verilog / VHDL).

[3]https://www.xilinx.com

- IP Package, corresponds to the RTL design that can be used by other tools in the Xilinx design flow.

- Synthesis reports are automatically generated in order to analyze the hardware design in terms of execution time and FPGA area.

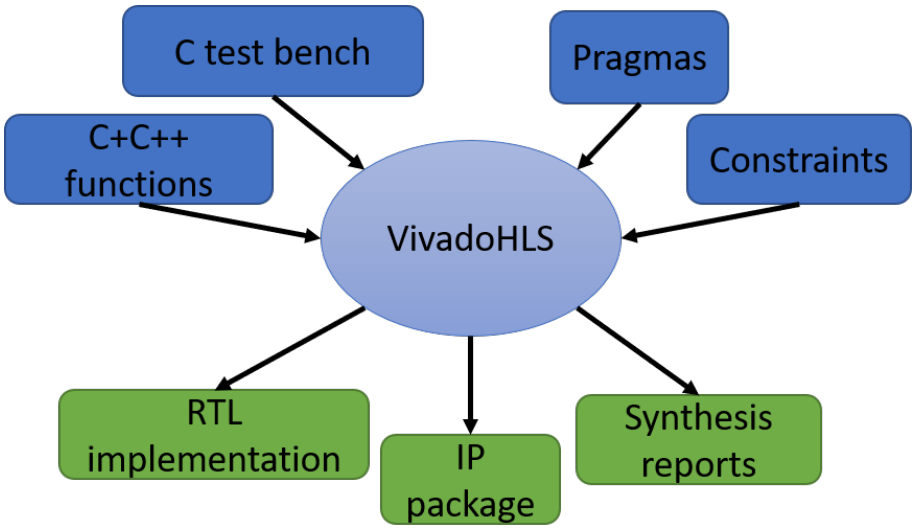


Figure 4.23: VivadoHLS overview.

There are 24 different pragmas[4] that can be applied to the C code to obtain an efficient hardware implementation. There is always a trade-off between resources used and processing speed. The use of pragmas makes it possible to adjust this compromise according to the constraints of the system. Among the possible optimization, we can mention some pragmas, such as:

– UNROLL pragma transforms loops by creating multiples copies of the loop body in the RTL) design, which allows some or all loop iterations to occur in parallel (Figure 4.24). In contrast, unrolled loops require more hardware resources and higher area usage.

– PIPELINE pragma reduces the initiation interval for a function or loop by allowing the concurrent execution of operations. Tasks are implemented in a concurrent manner when the PIPELINE pragma is used (Figure 4.25).

– ARRAY_PARTITION pragma divides an array into smaller arrays or individual elements . This increases the amount of read and write ports for a structure. It is then possible to concurrently access the elements of an array. Arrays can be partially or totally partitioned (Figre 4.26). However, it requires more memory instances or registers.

[4]available list at https://www.xilinx.com/html_docs/xilinx2019_1/sdaccel_doc/hls-pragmas-okr1504034364623.html

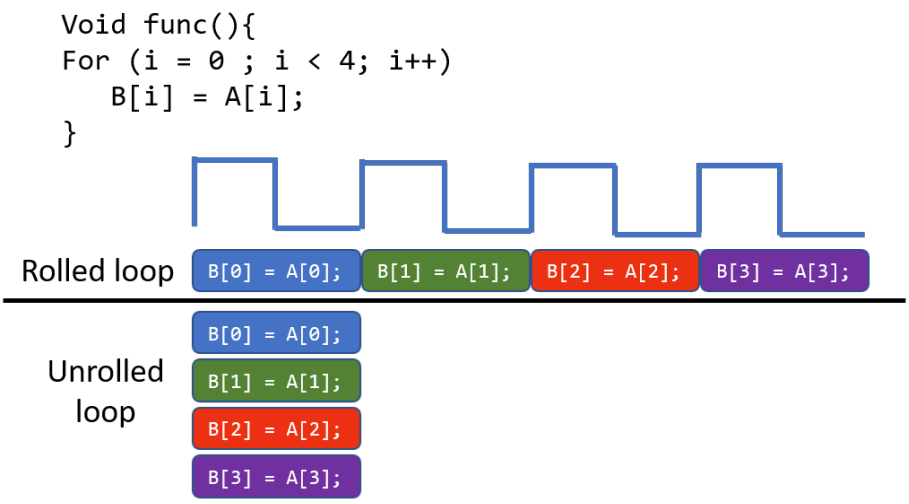


Figure 4.24: Example of an unrolled optimized loop.

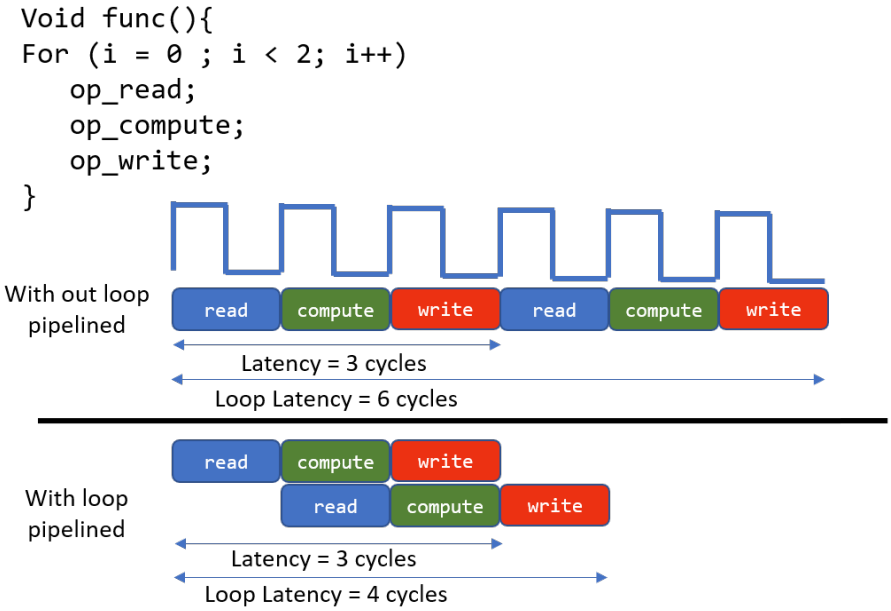


Figure 4.25: Example of a pipelined optimized loop.

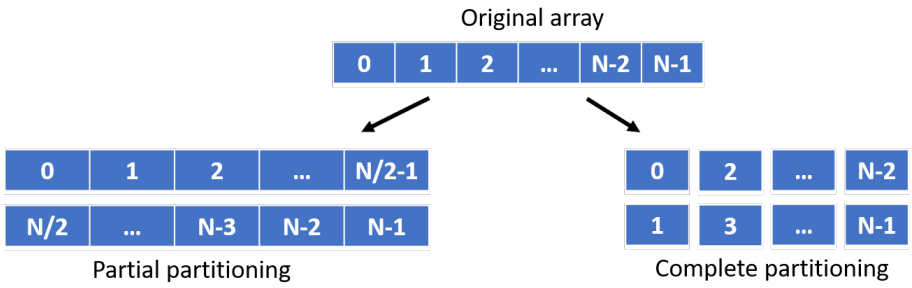


Figure 4.26: Examples of partial and complete partition of an array.

4.4.2 Fixed-point implementation

We realize a hardware architecture that allows to perform the missing information reconstruction function presented above. In a hardware architecture, floating-point computations are complex and expensive to implement. For this reason, they must be transformed into fixed-point computations. We impose a constraint for the fixed-point implementation, which is that the per-pixel error between the floating point and fixed-point implementations must be at most 1. This choice is a trade-off between implementation complexity and consistency of results between the different implementations. It is therefore necessary to define an adapted fixed-point buffer in order to have a sufficient accuracy throughout the calculation chain.

The input CDFA RGB-Z image is coded on 11 bits. The gradient and semi-gradient calculations are the sum of the absolute differences between the pixels. In the worst case, a total of 14 differences are summed. A minimum of $11 + 14 = 28$ bits buffer is therefore necessary to compute the gradients without loss. We use a standard 32-bit integer buffer to store these values. Then the gradients and semi-gradients are averaged. Each value is divided by the number of sums used to calculate it. The result is then stored in a 32-bit fixed point buffer whose integer part corresponds to 12 bits in order to store the range of normalized values. The rest of the 32-bit buffer is used for the decimal part to have the most accurate precision: it corresponds to 20 bits. The smallest possible coded value is therefore $\frac{1}{2}^{20}$. The equation 4.1 represents the fixed-point division used to compute the normalized gradients and semi-gradients.

$$gradient_{averaged} = (uint_{32})(((uint_{64})gradient << BIT\_SHIFT)/Nb_{sum}) \quad (4.1)$$

Where BIT_SHIFT represents the number of bits used to code the decimal part. In our case $BIT\_SHIFT = 20$. The two types $uint_{32}$ and $uint_{64}$ corresponds respectively to the standard 32-bit unsigned integer type and a 64-bit unsigned integer type. We use unsigned type, because we consider the absolute values of the differences, and not the differences themselves. Moreover, in this case, the rounding corresponds to a floor function (i.e. the nearest lower or equal integer value).

The main difficulty was to implement the discrimination function which uses the exponential function. Vivado does not propose an implementation of the exponential function in fixed point in these libraries. We used a method based on the following relation:

$$exp(a + b) = exp(a) * exp(b) \quad (4.2)$$

However, a first step allows to avoid unnecessary exponential calculations (4.3) for gradient (or semi-gradient) values which are too large compared to the sigma used. In this case sigma corresponds to the minimum of the four gradients (or eight semi-gradients) considered.

$$\gamma(ori) = \exp\left(-\frac{Grad^2_{averaged}}{2\sigma^2} \right) \tag{4.3}$$

The parameter of the exponential function is decomposed as follows:

$$-\frac{Grad^2_{averaged}}{2\sigma^2} = -1 \times \theta \times \frac{\theta}{2} \tag{4.4}$$

Where, $\theta = \frac{Grad_{averaged}}{\sigma}$. The term θ is firstly studied. Indeed, it is useless to compute the exponential function if the result cannot be coded on a fixed point whose decimal part corresponds to 20 bits. We must then solve the following equation:

$$exp\left(-1 \times \theta \times \frac{\theta}{2}\right) = (\frac{1}{2})^{20} \tag{4.5}$$

It corresponds to :

$$-\theta^2 = 2log\left((\frac{1}{2})^{20}\right) \tag{4.6}$$

The two solutions of the polynomial are:

$$\theta_s = \pm\sqrt{2 * log\left((\frac{1}{2})^{20}\right)} \simeq \pm 5.2655 \tag{4.7}$$

To facilitate the implementation, we study the value of θ. If it is greater than or equal to 6, then we do not compute the exponential, and the result is set to 0. In the case where θ is strictly inferior to 6, then we compute the exponential. The value of θ is in the interval $[0, 6[$, so according to equation 4.4, the integer part of the values that the parameter of the exponential takes is in the interval $]-18, 0]$.

The calculation of the exponential is therefore decomposed: $exp(a+b) = exp(a) * exp(b)$. We calculate on one side the integer part a and on the other side the decimal part b. A Lookup Table (LUT) containing the 19 results of the exponential calculation is used to directly read the result for the integer part.

On the other hand, the decimal part b of the exponential is computed using the same decomposition as before (4.2). We use a LUT that contains the 20 calculated values of the exponential corresponding to each bit of the decimal part. We loop through the bits of the decimal part, and multiply the values read in the LUT corresponding to each bit equal to 1. The following example helps to understand

the computing process of the exponential function implemented here. To compute the $exp(2.625)$, we first decompose:

$$exp(2.625) = exp(2) \times exp(0.625) \tag{4.8a}$$
$$= exp(2) \times exp(0.5) \times exp(0.125) \tag{4.8b}$$
$$= exp(2) \times exp((\frac{1}{2})^1) \times exp((\frac{1}{2})^3) \tag{4.8c}$$

We loop on the bits of the decimal part in order to identify which bits are at 1. In this example, we identify that bits number 1 and 3 are set to 1. These three values are then read in the two corresponding LUTs for the integer part and the decimal part. A total of three multiplications are computed in this example.

The presented method has been implemented with Vivado HLS in order to study its resources, performance and latency. The validation of the accuracy constraint has been verified experimentally on several examples. However, no error propagation model has been made. The results of the FPGA simulation are presented in the next section.

4.4.3 Simulation on FPGA target

In this section several simulations of the algorithmare performed on an FPGA target. The target considered is a Zynq UltraScale+ FPGA xczu9eg[5]. Its characteristics are illustrated in the Table 4.3. We chose to prototype our solution on an oversized FPGA target to study its latency behavior. However, the final objective would be to have a design on an Application Specific Integrated Circuit (ASIC) according to its performances study in this section.

Two solutions are generated with Vivado HLS. A first solution which is not optimized and corresponds to the C code. And a second optimized solution where pragmas are added to the C code to have a more efficient hardware implementation. However optimizing the latency, increases the used area of the FPGA. The optimization concerns the partition of some arrays in order to be able to make parallel memory accesses and thus to be able to pipeline some functions and loops. The loop of the kernel extraction is unrolled: the 25 values are loaded in parallel. The functions for the calculation of gradients and semi-gradients as well as the weights computation and the interpolation are pipelined. Moreover, inside each function,

[5]https://www.xilinx.com/support/documentation/data_sheets/ds891-zynq-ultrascale-plus-overview.pdf

loops are also pipelined. In this way the latency of each function is decreased and therefore the global latency. The resources used by the two solutions are presented in table 4.3.

	LUT	LUTRAM	FF	DSP	BRAM
available resources	274080	144000	548160	2520	912
Non-optimized Solution	6081 (2.22%)	100 (0.07%)	5851 (1.07%)	61 (2.42%)	9 (0.99%)
Optimized Solution	101732 (37.12%)	5486 (3.81%)	88757 (16.19%)	408 (16.19%)	2.5 (0.27%)

Table 4.3: Available resource of the Kintex UltraScale FPGA xczu9eg and resource consumption for both optimized and non-optimized solutions for VGA image resolution. In brackets are indicated the percentages of total available resources. LUT stands for LookUp Table. This element performs logic operations. FF stands for Flip-Flop, it is a register element that stores the result of the LUTs. BRAM is dedicated a Random-Access Memory Block, while LUTRAM is a LUT used as a RAM. DSP stand for Digital Signal Processing. DSP blocks provide several basic arithmetic primitives: multiplication, accumulation and addition.

Both solutions are tested using a test bench that streams a RGB-Z image as input, and the reconstructed CFA image as output of the algorithm. The reconstructed image is then compared with a reference image generated with the floating point implementation. For each solution, two simulations are performed. The first one is done using a thumbnail of 82×100 pixels. And the second one is done using a VGA image. The results of the two simulations are summarized in the Table 4.4. A 100Mhz clock is used for the simulations.

Solution	Image resolution	Number of cycles	Power estimation		
			static (W)	dynamic (W)	total (W)
Non-optimized Solution	640×480	210477354	0.622	0.103	0.725
Optimized Solution	640×480	1846326	0.643	2.54	3.183

Table 4.4: Results of the FPGA simulations for a VGA resolution image.

The detail of the number of cycles per function is given in the table 4.5. Two cases are to be taken into account. The worst (max) case and the best (min) case. The worst case corresponds to the most complex path in terms of operations to reconstruct a pixel. And the best case corresponds to the least complex path. These differences are due to the presence of IF conditions in the loops to calculate, in particular, the gradients. With pragmas, HLS organizes operations so that it

can start a new iteration more often. This requires a lot more resources, but the latency is both shorter and consistent. For the worst case, a maximum of 2796 cycles is required to reconstruct a missing pixel with the non-optimized solution. In the best case, 1387 cycles are needed to reconstruct a missing pixel. For the optimized solution, the number of cycles needed to reconstruct a missing pixel is 224. That represents up to 10 times fewer cycles for the reconstruction of a missing pixel.

	Non-optimized solution		Optimized solution	
	min	max	min	max
Kernel extraction	61	61	3	3
Gradients and semi-gradients computation	946	2280	109	109
Weights computation	131	206	55	55
Interpolation	249	249	57	57
Total	1387	2796	224	224

Table 4.5: Detail of number of cycles per function.

To generate an IP with vivado HLS it is necessary to validate the co-simulation step. The duration of the co-simulation step depends on the size of the image used in the test bench. For example, for a 82×100 pixels thumbnail, the co-simulation step takes few minutes to complete. For the VGA image, it is already necessary to consider several hours. This is why, no higher resolution images were simulated. An extrapolation of the results obtained with the two simulations allows however to make an estimation of the processing time necessary for different image resolutions. Table 4.6 shows the processing times for different image resolutions (simulated and extrapolated data). The same 100Mhz clock is considered to compute the frame rate.

Despite the first optimization step, the estimated frame rate for high resolution images (3Mpixels and more) does not correspond to the camera standards which are often between 30 and 60 frames per second. A possible optimization solution could be to process the pixel reconstruction in parallel. A solution could be to process the pixels four by four in parallel. Indeed, the missing pixels are spatially grouped in clusters of 4 pixels in the matrix. A second possibility could be to reduce the complexity of the processing. However this second solution may have an impact on the reconstructed image quality.

Image resolution	Solutions	Number of cycles	Simulation time (s)	Frame rate (img/s)
82×100	Non-optimized	5012990	0.05013	19.948
	Optimized	49447	0.00049	2022.388
VGA	Non-optimized	210477354	2.10477	0.475
	Optimized	1846326	0.01846	54.162
3Mpixels	Non-optimized	2060893544	20.60894	0.049
	Optimized	18029064	0.18029	5.547
8Mpixels	Non-optimized	5496752482	54.96752	0.018
	Optimized	48077224	0.48077	2.080

Table 4.6: Frame rate according to the resolution of the images. Simulated and extrapolated data.

4.5 Conclusion

In this chapter, we have seen the simulation chain, and the processing chain used to evaluate our reconstruction algorithms. The results of the image quality evaluation were also presented. The proposed solution using semi-gradients presents, on average, better results than the EDI solutions. These results are confirmed by subjective evaluation of the reconstructed images. Finally, a first hardware implementation in fixed point has been done with the Vivado HLS tool in order to study the latency of the algorithmic solution using the semi-gradients. The results of the proposed implementation do not correspond to the cameras standards. Latency and processing time remain an issue for the moment. However, solutions of parallelisation of the processing are considered in order to reach a suitable image rate, especially for high resolution images.

The reconstruction quality has been significantly improved, with the solution using the semi-gradients, bringing us closer to the quality of native RGB sensor. However, there are still some artifacts, especially when reconstructing the corners for the Mask2 and the narrow edges for the Mask1.

Chapter 5

Conclusion and Future Work

The work presented in this thesis focuses on the missing information reconstruction chain for a monolithic mixed RGB-Z sensor. Such a monolithic sensor would provide an elegant, integrated, and compact solution. Another advantage of such a sensor is the intrinsic spatial co-location of depth and color data. The objective was to study different RGB-Z matrix architectures and to propose solutions for the reconstruction of the missing information in order to have performances similar to a classical RGB sensor. We were primarily interested in the reconstruction of color pixels. For example, in the context of a sensor dedicated to the cell phone market, it is essential to display high quality color images. The reconstruction of color information is therefore a key issue in this context.

In a first step, four different architectures of RGB-Z matrix based on two different Z-pixels were proposed. A first Z-pixel whose dimensions are identical to those of the pixels of color (i.e. $3.2\,\mu m^2$ for both pixels). And a second Z-pixel whose dimensions are twice as large as those of a color pixel (i.e. $3.2\,\mu m^2$ for the Z-pixels and $1.6\,\mu m^2$ for the color pixels). Two architectures are based on the Bayer CFA and two architectures are based on a non-standard CFA model. However, the non-Bayer matrices were quickly rejected.

Indeed, the demosaicing step represents the reconstruction of 2/3 of the total color information of the final image, whereas the reconstruction of the missing color information in the CFA image represents 1/12 of the total information. We did not have an efficient demosaicing algorithm for these CFA, but we had a demosaicing algorithm dedicated to the Bayer pattern already validated by STMicroelectronics. Moreover this demosaicing solution is already validated by STMicroelectronics. This is why we finally studied more thoroughly the two RGB-Z architectures based on the Bayer model. The first one uses the 1x1 Z-pixel and the second one uses the 2x2 Z-pixel.

In a second step, we implemented several algorithms for the reconstruction of the missing color information. These algorithms allow obtaining a CFA image from a CDFA image. Three methods inspired by the state of the art have been adapted to our RGB-Z matrices and then studied. One bilateral interpolation and two edge-directed interpolations have been implemented. These three solutions present different results but have the common drawback of introducing significant structural and false color artifacts in the reconstructed color image.

A solution using a new operator we call semi-gradient has been implemented for the two proposed RGB-Z architectures. The semi-gradient operator studies the close neighborhood of the missing pixel in order to determine to which texture it belongs within the computational window. The objectives of this methods are to reduce artifacts such as zipper effect, and reconstruct cleaner edges. Moreover, a joint reconstruction solution between color information and information from depth pixels has been studied and implemented.

A series of simulations on different databases has shown an significant improvement of the reconstructed images quality with the solution using the semi-gradients compared to the solutions inspired by the state of the art. In addition, the reconstruction of narrow structures is better. The databases used have different image resolutions and different frequency contents. CDFA image were generated using a sensor model that simulate both color and depth sensor in order to obtain raw RGB-Z images. The comparison of the algorithms was done using different metrics (PSNR, SSIM, Zipper Metric) applied on the reconstructed CFA image, as well as on the color image.

Although the quality is significantly improved with the semi-gradient method, the reconstructed images still show some artifacts specific to each RGB-Z architecture. For the Mask1, these artifacts concern mainly features of a width of the order of a pixel. It is possible in this case that there are no pixels with similar values to the missing one inside the filtering kernel. A correct reconstruction is then impossible. In the case of Mask2, the reconstruction of the corners may present artifacts. This can be observed when the Z-pixel is located at the limit of the corner. The use of the joint reconstruction might be a solution to overcome this problem, however this point is not straightforward and could not be demonstrated during this thesis. However, depending on the use cases, these remaining quality problems, which are very localized, could be acceptable. For example, for a given screen, the higher the resolution of the image, the less fine details are visible and thus the less remaining

artifacts are visible.

Finally a study of the complexity of the solution using the semi-gradient in the case of mask2 was conducted. The mask2 solution is the most complex, because it uses both gradients and semi-gradients, that's why we chose it for the complexity study. For this, we used the HLS tools from Xilinx to generate a hardware architecture of the algorithm. This architecture has been simulated on an FPGA target with a 100Mhz clock in order to quantify the latency of the system. The objective is to be able to integrate the solution within an ISP processing chain. A constraint is then to be able to perform the processing at a frequency of 30 frames per second. A first optimization step allows to reach the 50 frames per second frequency in simulation for VGA resolution images. However, a higher image resolution does not allow to have a fast enough processing to respect the constraint of 30 frames per second. For 3M pixel images, the frame rate drops to 5 frames per second.

One of the difficulties encountered during this work is related to the lack of database including scenes with information of color, depth and reflectance in the infrared. Such a database would have allowed us to fully test the joint reconstruction solution. However, the creation of such a database would be a very complex task, and was not feasible in the context of this thesis. Furthermore, the lack of real data during the thesis limited us to simulation studies of the reconstruction algorithms. The first real image samples capture with a RGB-Z matrix corresponding to Mask1 were available in the last weeks of the thesis. These data do not present any information for the Z-pixels. This is why, only the solutions not exploiting the information from the depth pixels could be applied. Finally, IR pollution has not been studied on these samples.

A first perspective is therefore the exploitation of real data for the two solutions of RGB-Z architecture presented in this work. As well as the exploration of a joint reconstruction with real data when these will be available. A second perspective is to set up a hardware architecture allowing to parallelize the processing of the pixels to be reconstructed. Such solution is already considered but could not be implemented yet.

Beyond the solutions proposed in this work, an interesting perspective would be to consider a reconstruction chain where the steps of demosaicing and reconstruction of the missing information due to the Z-pixels are done together. The amount of known information does not vary between the two steps. A solution allowing to directly reconstruct the 3 color channels for each pixel from the CDFA image

could thus allow to decrease the latency. Moreover, this would avoid using poorly reconstructed information during the demosaicing step.
Future projects concerning RGB-Z sensors could also explore other RGB-Z patterns. The study of given application cases could allow to propose new patterns with a reduced Z-pixel density. That could be relevant, especially for the mask1 because the remaining artifacts are due to the absence of green pixel on the horizontal and vertical axes during the interpolation. An example of a potential sparse pattern is given in Figure 5.1. However, these solutions depend on the quality of the depth information required according the use case.

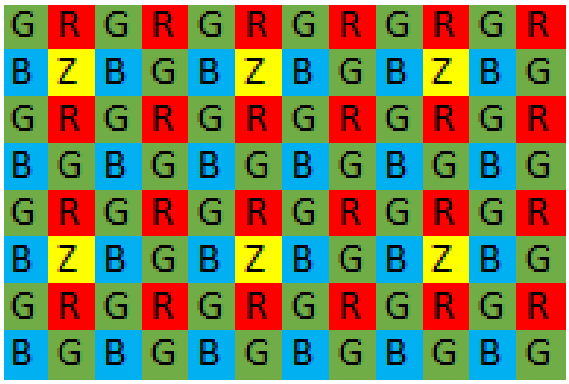


Figure 5.1: Proposition of sparse RGB-Z pattern based on the 1x1 Z-pixel.

List of publications related to this work (April 2021)

Conference:

V. Rebiere, A. Drouot, B. Granado, A. Bourge, A. Pinna, "Semi-Gradient for Color Pixel Reconstruction in a RGBZ CMOS Sensor", IEEE Sensors, 2020. DOI: 10.1109/SENSORS47125.2020.9278887.

V. Rebiere, A. Drouot, B. Granado, A. Bourge, A. Pinna, "Color pixel reconstruction for a monolithic RGB-Z CMOS Imager", Journal of Signal Processing Systems (JSP), 2021.

Patent:

A patent application is due to be filled soon.

Appendix A

Appendices

A.1 Metric tables of CFA-independent algorithm compared to Mozart

	Mozart	CFAInd + Bayer	CFAInd + Non-Bayer Mask1	CFAInd + Non--Bayer Mask2
Kodak1	**20.69**	20.35	20.17	20.19
Kodak2	22.42	22.52	24.89	**24.93**
Kodak3	21.64	21.62	21.62	**21.65**
Kodak4	21.26	21.33	21.71	**21.73**
Kodak5	**22.04**	21.92	21.58	21.63
Kodak6	**19.19**	19.04	18.65	18.65
Kodak7	21.13	21.13	21.19	**21.21**
Kodak8	**18.90**	18.73	18.17	18.16
Kodak9	**19.53**	19.46	18.98	18.99
Kodak10	**20.19**	20.12	19.53	19.53
Kodak11	**21.90**	21.82	21.37	21.38
Kodak12	**18.03**	17.98	17.70	17.71
Kodak13	**20.01**	19.91	19.59	19.62
Kodak14	21.54	21.54	21.62	**21.66**
Kodak15	19.55	**19.62**	19.12	19.13
Kodak16	**21.43**	21.31	20.81	20.82
Kodak17	**22.85**	22.80	22.44	22.45
Kodak18	**23.45**	23.33	23.36	23.39
Kodak19	**20.21**	20.03	19.67	19.70
Kodak20	**16.79**	16.78	15.87	15.87
Kodak21	**20.20**	20.12	19.37	19.39
Kodak22	**20.77**	20.69	20.46	20.50
Kodak23	**20.96**	20.93	20.89	20.95
Kodak24	**20.48**	20.39	19.81	19.86
average	**20.63**	20.56	20.36	20.38
standard deviation	1.52	1.53	1.91	1.92

Table A.1: Details of the performances of demosaicing methods in terms of PSNR, computed on the Kodak database [11].

	Mozart	CFAInd + Bayer	CFAInd + Non-Bayer Mask1	CFAInd + Non--Bayer Mask2
Kodak1	**0.901**	0.833	0.822	0.825
Kodak2	0.893	**0.895**	0.860	0.864
Kodak3	**0.945**	0.928	0.904	0.910
Kodak4	**0.912**	0.903	0.880	0.885
Kodak5	**0.918**	0.868	0.865	0.867
Kodak6	**0.916**	0.858	0.840	0.844
Kodak7	**0.949**	0.935	0.908	0.913
Kodak8	**0.897**	0.841	0.831	0.833
Kodak9	**0.942**	0.919	0.883	0.890
Kodak10	**0.942**	0.915	0.890	0.895
Kodak11	**0.906**	0.863	0.855	0.858
Kodak12	**0.940**	0.909	0.858	0.867
Kodak13	**0.884**	0.806	0.801	0.802
Kodak14	**0.915**	0.864	0.856	0.858
Kodak15	**0.905**	0.900	0.873	0.880
Kodak16	**0.925**	0.882	0.861	0.866
Kodak17	**0.914**	0.883	0.882	0.885
Kodak18	**0.916**	0.866	0.864	0.865
Kodak19	**0.921**	0.881	0.859	0.864
Kodak20	**0.937**	0.920	0.903	0.908
Kodak21	**0.928**	0.891	0.864	0.875
Kodak22	**0.927**	0.877	0.852	0.862
Kodak23	**0.949**	0.940	0.902	0.918
Kodak24	**0.929**	0.876	0.859	0.868
average	**0.921**	0.886	0.865	0.871
standard deviation	0.018	0.033	0.026	0.0285

Table A.2: Details of the performances of demosaicing methods in terms of SSIM, computed on the Kodak database [11].

A.2 Practical examples of pixel reconstruction for the Mask1

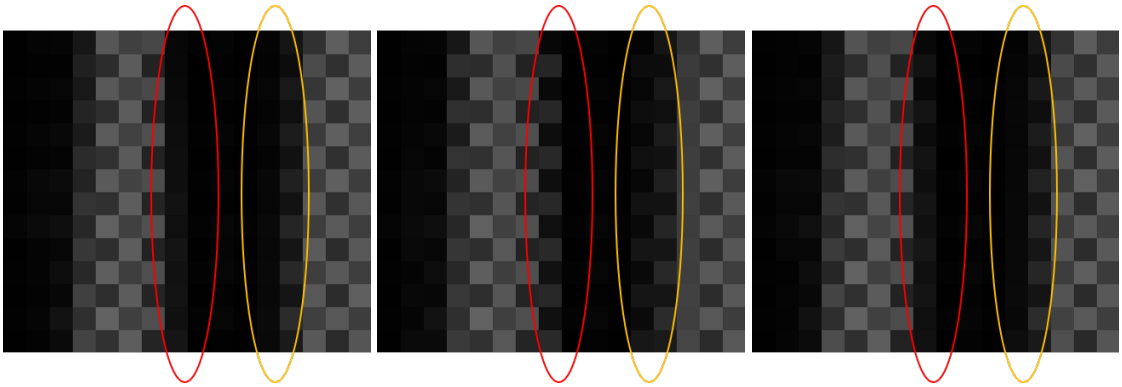
In this example (Figure A.1), the missing pixel is at the level of a texture change. The left part of the kernel is darker, while the right part is lighter. The gradients allow to determine the presence of a vertical edge in this context, but the lack of green information along the vertical axis does not allow to interpolate the missing pixel properly.

In this example the pixel values are coded on 11bits. The actual missing pixel value is 1481. With the edge-directed interpolation method, without semi-gradient, the reconstructed value is 1126. Using the semi-gradient method presented in section 3.4.3, the reconstructed value is then 1485. In this example, we have an error of almost 24% in the first case, while we have an error of less than 1% with the semi-gradients.

Z	395	Z	1924	Z
150	389	1383	1836	1161
Z	477	Missing pixel	1971	Z
157	476	1420	1805	1262
Z	593	Z	2006	Z

Figure A.1: Example of a computation kernel of the Mask1. The semi-gradients can identify that the missing pixel are closest to the right side in term of pixel intensity.

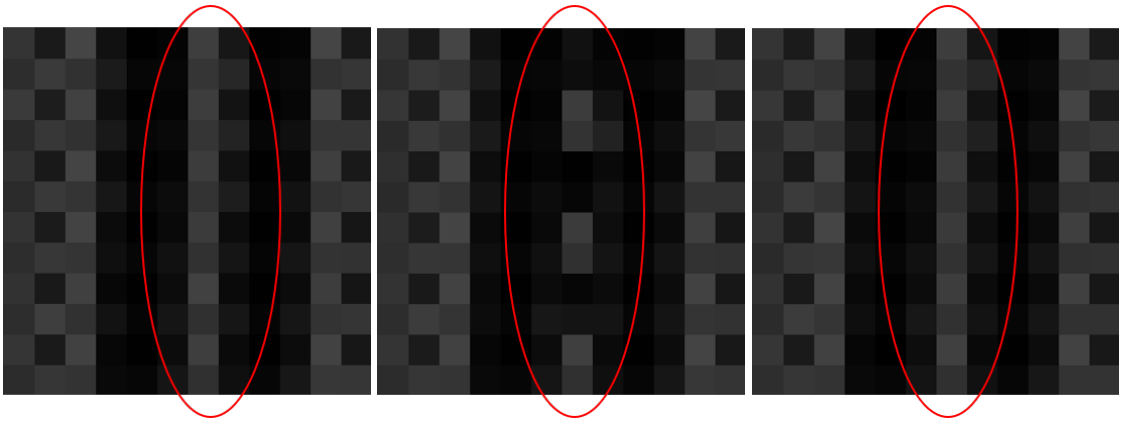
A.3 Examples of Bayer CFA reconstructed from the Mask1



(a) (b) (c)

Figure A.2: Example of Bayer CFA after reconstruction of the missing pixels in a context of a Mask1. (a) Reference image. (b) EDI reconstruction algorithm. (c) Reconstruction algorithm using semi-gradients. In red and yellow, examples of edge pixel reconstruction enhancements.

A.4 Examples of Bayer CFA reconstructed from the Mask2



(a) (b) (c)

Figure A.3: Example of Bayer CFA after reconstruction of the missing pixels in a context of a Mask2. (a) Reference image. (b) EDI algorithm. (c) Reconstruction algorithm using semi-gradients. The red circles indicates the evolution of the reconstructed pixels along the edge.

A.5 Images used from the HDR+ Burst dataset

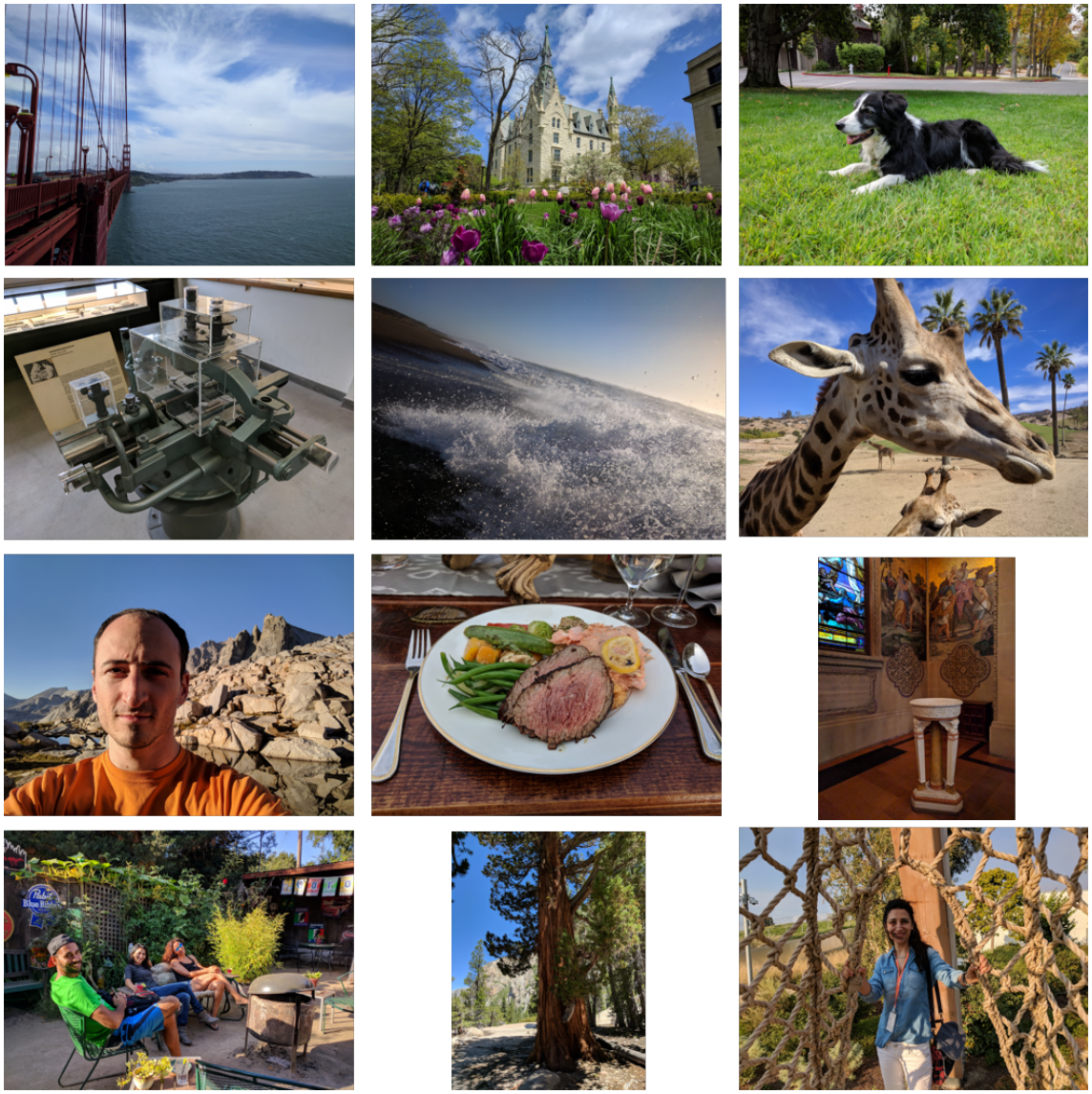


Figure A.4: The twelve images used from the HDR+ Burst data set [26].

A.6 Metrics results of color pixel reconstruction algorithms applied on the Kodak database for the Mask1

	BI	EDICentral	EDILine	SG
Kodak1	24.3	24.06	24.29	**25.17**
Kodak2	31.86	31.65	32.11	**32.53**
Kodak3	32.48	32.56	32.95	**33.7**
Kodak4	32.35	32.56	32.98	**33.47**
Kodak5	24.62	24.98	25.84	**26.69**
Kodak6	25.81	25.53	25.88	**26.47**
Kodak7	31.35	32.00	32.3	**33.53**
Kodak8	21.49	21.38	21.61	**22.62**
Kodak9	30.1	30.19	30.47	**31.99**
Kodak10	30.49	30.52	30.92	**31.98**
Kodak11	27.57	27.34	27.85	**28.63**
Kodak12	30.68	30.72	30.99	**32.27**
Kodak13	22.62	22.26	22.86	**23.22**
Kodak14	27.47	27.6	28.01	**28.82**
Kodak15	29.02	29.17	29.46	**31.35**
Kodak16	29.61	29.4	29.72	**30.12**
Kodak17	30.34	30.5	30.98	**31.82**
Kodak18	26.59	26.3	26.85	**27.4**
Kodak19	26.02	25.99	26.17	**26.92**
Kodak20	29.69	30.15	30.58	**31.6**
Kodak21	26.84	26.71	27.08	**27.87**
Kodak22	28.81	28.61	28.97	**29.64**
Kodak23	33.12	33.68	34.00	**34.47**
Kodak24	25.43	25.06	25.54	**25.96**
average	28.28	28.29	28.68	**29.51**
standard deviation	3.20	3.36	3.34	3.41

Table A.3: Details of the performances of the reconstruction methods on the Mask1 in term of PSNR (dB), computed between the reconstructed pixels in the CFA image and the reference pixels in the reference CFA image on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	31.43	31.25	31.44	**32.19**
Kodak2	37.59	37.62	37.84	**38.13**
Kodak3	38.01	38.35	38.61	**39.09**
Kodak4	39.06	39.60	39.81	**40.20**
Kodak5	30.72	31.56	32.12	**32.78**
Kodak6	32.99	32.82	33.11	**33.52**
Kodak7	36.77	37.76	37.89	**38.95**
Kodak8	28.44	28.39	28.57	**29.53**
Kodak9	37.15	37.48	37.67	**39.05**
Kodak10	37.13	37.52	37.75	**38.56**
Kodak11	34.17	34.13	34.47	**35.15**
Kodak12	37.85	38.24	38.48	**39.31**
Kodak13	28.81	28.64	29.03	**29.28**
Kodak14	33.86	34.12	34.42	**35.07**
Kodak15	37.25	37.77	38.16	**38.81**
Kodak16	36.81	36.72	36.92	**37.25**
Kodak17	36.68	37.11	37.40	**37.84**
Kodak18	32.52	32.42	32.77	**33.23**
Kodak19	32.74	32.75	32.88	**33.66**
Kodak20	35.84	36.46	36.78	**37.70**
Kodak21	33.53	33.52	33.83	**34.53**
Kodak22	35.26	35.25	35.47	**35.99**
Kodak23	38.32	38.97	39.16	**39.48**
Kodak24	32.12	31.97	32.32	**32.67**
average	34.79	35.02	35.29	**35.92**
standard deviation	3.06	3.25	3.21	3.24

Table A.4: Details of the performances of the reconstruction methods on the Mask1 in term of of C-PSNR (dB), computed between the reconstructed color images and the reference color images on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	0.9328	0.9288	0.9326	**0.9411**
Kodak2	0.9726	0.9717	0.9732	**0.9746**
Kodak3	0.9756	0.9751	0.9768	**0.9781**
Kodak4	0.9770	0.9766	0.9780	**0.9792**
Kodak5	0.9506	0.9577	0.9618	**0.9672**
Kodak6	0.9424	0.9396	0.9429	**0.9465**
Kodak7	0.9802	0.9823	0.9831	**0.9857**
Kodak8	0.9340	0.9333	0.9362	**0.9447**
Kodak9	0.9732	0.9728	0.9744	**0.9772**
Kodak10	0.9725	0.9733	0.9747	**0.9779**
Kodak11	0.9511	0.9499	0.9528	**0.9576**
Kodak12	0.9683	0.9671	0.9689	**0.9708**
Kodak13	0.9200	0.9155	0.9222	**0.9267**
Kodak14	0.9550	0.9550	0.9575	**0.9619**
Kodak15	0.9725	0.9720	0.9741	**0.9753**
Kodak16	0.9550	0.9533	0.9554	**0.9575**
Kodak17	0.9733	0.9743	0.9758	**0.9778**
Kodak18	0.9544	0.9529	0.9565	**0.9606**
Kodak19	0.9552	0.9533	0.9558	**0.9593**
Kodak20	0.9711	0.9713	0.9734	**0.9757**
Kodak21	0.9616	0.9606	0.9631	**0.9670**
Kodak22	0.9605	0.9582	0.9611	**0.9644**
Kodak23	0.9823	0.9820	0.9832	**0.9840**
Kodak24	0.9568	0.9562	0.9590	**0.9636**
average	0.9603	0.9597	0.9622	**0.9656**
standard deviation	0.0162	0.0173	0.0162	0.0147

Table A.5: Details of the performances of the reconstruction methods on the Mask1 in term of of SSIM, computed between the reconstructed color images and the reference color images on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	0.9787	0.9774	0.9787	**0.9813**
Kodak2	0.9891	0.9886	0.9894	**0.9899**
Kodak3	0.9939	0.9936	0.9942	**0.9944**
Kodak4	0.9934	0.9930	0.9937	**0.9939**
Kodak5	0.9891	0.9899	0.9913	**0.9925**
Kodak6	0.9793	0.9778	0.9793	**0.9802**
Kodak7	0.9953	0.9957	0.9960	**0.9968**
Kodak8	0.9801	0.9798	0.9808	**0.9835**
Kodak9	0.9925	0.9922	0.9927	**0.9937**
Kodak10	0.9929	0.9929	0.9934	**0.9942**
Kodak11	0.9855	0.9848	0.9859	**0.9871**
Kodak12	0.9891	0.9886	0.9892	**0.9896**
Kodak13	0.9770	0.9750	0.9777	**0.9789**
Kodak14	0.9891	0.9890	0.9898	**0.9908**
Kodak15	0.9935	0.9932	0.9938	**0.9940**
Kodak16	0.9830	0.9822	0.9832	**0.9837**
Kodak17	0.9938	0.9939	0.9944	**0.9948**
Kodak18	0.9886	0.9880	0.9891	**0.9903**
Kodak19	0.9841	0.9832	0.9841	**0.9847**
Kodak20	0.9928	0.9927	0.9933	**0.9938**
Kodak21	0.9899	0.9896	0.9904	**0.9915**
Kodak22	0.9885	0.9875	0.9887	**0.9897**
Kodak23	0.9957	0.9957	0.9960	**0.9962**
Kodak24	0.9888	0.9883	0.9893	**0.9904**
average	0.9885	0.9880	0.9889	**0.9898**
standard deviation	0.0055	0.0060	0.0056	0.0052

Table A.6: Details of the performances of the reconstruction methods on the Mask1 in term of of M-SSIM, computed between the reconstructed color images and the reference color images on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	10.082	11.172	10.563	**8.531**
Kodak2	1.704	1.929	1.728	**1.621**
Kodak3	1.030	0.988	0.818	**0.533**
Kodak4	1.146	1.286	1.047	**0.962**
Kodak5	12.651	11.883	10.569	**9.268**
Kodak6	3.310	3.956	3.486	**3.122**
Kodak7	2.565	2.311	2.165	**1.520**
Kodak8	21.138	21.660	21.049	**17.362**
Kodak9	3.659	3.589	3.472	**2.391**
Kodak10	2.856	2.707	2.598	**2.019**
Kodak11	5.666	5.941	5.507	**4.558**
Kodak12	1.619	1.555	1.436	**1.016**
Kodak13	10.161	11.785	10.513	**9.613**
Kodak14	4.741	5.124	4.602	**3.953**
Kodak15	2.946	2.785	2.476	**1.989**
Kodak16	2.133	2.552	2.194	**2.009**
Kodak17	3.219	3.061	2.801	**2.204**
Kodak18	6.931	7.558	6.757	**5.747**
Kodak19	8.317	8.715	8.334	**6.829**
Kodak20	3.797	3.672	3.357	**2.590**
Kodak21	5.975	6.573	5.957	**5.138**
Kodak22	4.262	4.715	4.364	**3.739**
Kodak23	1.164	1.126	1.025	**0.851**
Kodak24	8.316	8.795	8.094	**6.560**
average	5.391	5.643	5.205	**4.339**
standard deviation	4.641	4.825	4.601	3.886

Table A.7: Details of the performances of the reconstruction methods on the Mask1 in term of of Zipper metric (%), computed between the reconstructed color images and the reference color images on the Kodak database [11].

A.7 Metrics results of color pixel reconstruction algorithms applied on the McMaster database for the Mask1

	BI	EDICentral	EDILine	SG
McM1	24.44	24.72	25.08	**25.57**
McM2	28.69	28.88	29.31	**30.69**
McM3	25.19	25.4	25.92	**26.57**
McM4	27.62	28.6	28.92	**30.24**
McM5	30.07	30.6	30.74	**31.49**
McM6	33.41	33.69	33.98	**35.02**
McM7	28.96	28.66	29.17	**29.95**
McM8	28.43	28.39	29.00	**30.07**
McM9	30.96	31.51	31.86	**33.19**
McM10	33.14	33.71	33.81	**34.7**
McM11	33.56	33.61	34.19	**34.84**
McM12	29.54	28.92	29.95	**31.34**
McM13	34.8	35.22	35.18	**36.47**
McM14	34.56	35.09	35.41	**36.22**
McM15	34.41	35.02	35.45	**36.19**
McM16	26.54	26.61	27.21	**27.93**
McM17	29.98	30.46	30.91	**31.35**
McM18	26.62	26.59	26.93	**29.21**
average	30.05	30.32	30.72	**31.72**
standard deviation	3.32	3.44	3.35	3.33

Table A.8: Details of the performances of the reconstruction methods on the Mask1 in term of PSNR (dB), computed between the reconstructed pixels in the CFA image and the reference pixels in the reference CFA image on the McMaster database [25].

	BI	EDICentral	EDILine	SG
McM1	29.45	29.71	29.99	**30.34**
McM2	35.08	35.43	35.71	**37.05**
McM3	30.80	31.19	31.56	**32.07**
McM4	32.70	34.07	34.27	**35.40**
McM5	36.67	37.49	37.51	**38.14**
McM6	39.77	40.31	40.53	**41.50**
McM7	36.12	36.06	36.48	**37.31**
McM8	34.84	35.07	35.46	**36.48**
McM9	36.43	37.12	37.34	**38.44**
McM10	38.64	39.09	39.14	**39.72**
McM11	37.94	38.04	38.39	**38.85**
McM12	35.95	35.87	36.31	**37.29**
McM13	40.87	41.38	41.36	**42.43**
McM14	39.60	40.20	40.40	**40.91**
McM15	39.67	40.52	40.70	**41.24**
McM16	29.94	29.91	30.35	**30.91**
McM17	34.93	35.28	35.59	**35.96**
McM18	31.50	31.45	31.70	**33.15**
average	35.61	36.01	36.27	**37.07**
standard deviation	3.55	3.67	3.59	3.61

Table A.9: Details of the performances of the reconstruction methods on the Mask1 in term of of C-PSNR (dB), computed between the reconstructed color images and the reference color images on the McMaster database [25]

	BI	EDICentral	EDILine	SG
McM1	0.9437	0.9443	0.9473	**0.9508**
McM2	0.9686	0.9686	0.9709	**0.9745**
McM3	0.9546	0.9572	0.9611	**0.9656**
McM4	0.9796	0.9829	0.9841	**0.9871**
McM5	0.9741	0.9751	0.9763	**0.9788**
McM6	0.9820	0.9823	0.9832	**0.9854**
McM7	0.9593	0.9571	0.9605	**0.9644**
McM8	0.9704	0.9705	0.9733	**0.9763**
McM9	0.9780	0.9792	0.9803	**0.9832**
McM10	0.9817	0.9823	0.9831	**0.9848**
McM11	0.9751	0.9748	0.9770	**0.9787**
McM12	0.9759	0.9753	0.9776	**0.9818**
McM13	0.9852	0.9850	0.9857	**0.9871**
McM14	0.9817	0.9820	0.9831	**0.9841**
McM15	0.9823	0.9827	0.9840	**0.9850**
McM16	0.9401	0.9400	0.9460	**0.9523**
McM17	0.9691	0.9721	0.9739	**0.9758**
McM18	0.9481	0.9458	0.9496	**0.9575**
average	0.9694	0.9698	0.9721	**0.9752**
standard deviation	0.0142	0.0146	0.0134	0.0120

Table A.10: Details of the performances of the reconstruction methods on the Mask1 in term of of SSIM, computed between the reconstructed color images and the reference color images on the McMaster database [25].

	BI	EDICentral	EDILine	SG
McM1	0.9868	0.9870	0.9878	**0.9885**
McM2	0.9934	0.9934	0.9939	**0.9946**
McM3	0.9895	0.9898	0.9910	**0.9920**
McM4	0.9956	0.9963	0.9966	**0.9972**
McM5	0.9940	0.9943	0.9946	**0.9950**
McM6	0.9964	0.9964	0.9966	**0.9971**
McM7	0.9907	0.9901	0.9910	**0.9918**
McM8	0.9938	0.9938	0.9943	**0.9950**
McM9	0.9953	0.9954	0.9958	**0.9964**
McM10	0.9964	0.9965	0.9966	**0.9970**
McM11	0.9950	0.9949	0.9954	**0.9957**
McM12	0.9941	0.9934	0.9943	**0.9951**
McM13	0.9957	0.9958	0.9960	**0.9964**
McM14	0.9962	0.9963	0.9965	**0.9967**
McM15	0.9961	0.9962	0.9964	**0.9967**
McM16	0.9885	0.9884	0.9897	**0.9911**
McM17	0.9935	0.9940	0.9945	**0.9947**
McM18	0.9883	0.9875	0.9886	**0.9901**
average	0.9933	0.9933	0.9939	**0.9945**
standard deviation	0.0031	0.0033	0.0029	0.0026

Table A.11: Details of the performances of the reconstruction methods on the Mask1 in term of of M-SSIM, computed between the reconstructed color images and the reference color images on the McMaster database [25].

	BI	EDICentral	EDILine	SG
McM1	9.466	9.543	9.101	**7.969**
McM2	5.386	5.252	4.988	**3.939**
McM3	8.778	8.480	7.815	**6.347**
McM4	5.151	4.753	4.420	**3.135**
McM5	3.055	2.985	2.823	**2.332**
McM6	2.122	1.913	1.784	**1.259**
McM7	7.927	8.340	7.749	**6.421**
McM8	6.727	6.299	5.844	**4.555**
McM9	2.406	2.357	2.088	**1.747**
McM10	1.212	1.094	0.968	**0.862**
McM11	1.422	1.461	1.221	**1.046**
McM12	4.015	4.568	3.956	**3.160**
McM13	1.256	1.184	1.191	**0.912**
McM14	1.032	0.982	0.881	**0.692**
McM15	0.968	0.812	0.674	**0.543**
McM16	2.759	3.268	2.586	**2.127**
McM17	3.316	2.858	2.606	**2.422**
McM18	4.764	4.920	4.490	**3.035**
average	3.987	3.948	3.622	**2.917**
standard deviation	2.747	2.767	2.616	2.184

Table A.12: Details of the performances of the reconstruction methods on the Mask1 in term of of Zipper metric (%), computed between the reconstructed color images and the reference color images on the McMaster database [25].

A.8 Metrics results of color pixel reconstruction algorithms applied on the Kodak database for the Mask2

	BI	EDICentral	EDILine	SG
Kodak1	23.62	25.39	25.94	**26.74**
Kodak2	31.19	32.01	32.52	**32.79**
Kodak3	31.39	32.40	32.75	**33.15**
Kodak4	30.88	31.88	32.16	**32.23**
Kodak5	23.45	24.40	25.06	**25.11**
Kodak6	25.58	26.21	26.90	**27.72**
Kodak7	29.34	31.38	31.62	**31.98**
Kodak8	21.14	23.00	23.85	**25.22**
Kodak9	28.80	31.32	31.67	**32.47**
Kodak10	29.47	30.29	31.02	**31.50**
Kodak11	26.97	27.86	28.61	**29.08**
Kodak12	30.06	32.09	32.59	**33.29**
Kodak13	22.33	22.26	22.86	**23.02**
Kodak14	26.42	27.35	27.70	**28.00**
Kodak15	29.03	31.32	31.94	**32.02**
Kodak16	29.80	30.20	30.93	**31.79**
Kodak17	29.32	30.35	30.92	**30.95**
Kodak18	25.83	26.03	26.57	**26.86**
Kodak19	25.37	27.52	28.66	**30.14**
Kodak20	28.62	30.13	30.56	**31.02**
Kodak21	26.27	26.82	27.52	**28.17**
Kodak22	28.16	28.84	29.68	**30.05**
Kodak23	31.12	32.10	32.68	**32.91**
Kodak24	24.84	24.67	25.31	**25.52**
average	27.46	28.58	29.17	**29.66**
standard deviation	2.96	3.15	3.07	3.02

Table A.13: Details of the performances of the reconstruction methods on the Mask2 in term of PSNR (dB), computed between the reconstructed pixels in the CFA image and the reference pixels in the reference CFA image on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	28.01	30.12	30.47	**31.30**
Kodak2	32.93	33.76	34.42	**34.74**
Kodak3	34.83	36.17	36.50	**36.81**
Kodak4	33.57	34.20	34.57	**34.71**
Kodak5	27.04	28.63	29.07	**29.18**
Kodak6	30.11	30.94	31.41	**32.27**
Kodak7	32.23	34.86	34.93	**35.38**
Kodak8	25.29	27.71	28.29	**29.59**
Kodak9	33.09	36.20	36.49	**37.27**
Kodak10	33.78	34.92	35.45	**35.89**
Kodak11	30.80	32.18	32.78	**33.25**
Kodak12	34.66	36.78	37.18	**37.84**
Kodak13	26.38	26.66	27.10	**27.29**
Kodak14	30.04	31.27	31.57	**31.93**
Kodak15	32.77	34.32	34.95	**35.25**
Kodak16	34.12	34.91	35.46	**36.39**
Kodak17	33.35	34.51	35.01	**35.05**
Kodak18	29.44	30.14	30.50	**30.81**
Kodak19	29.37	32.06	32.97	**34.22**
Kodak20	32.41	34.45	34.70	**35.19**
Kodak21	30.03	31.06	31.57	**32.22**
Kodak22	32.06	33.11	33.77	**34.15**
Kodak23	33.97	35.36	35.75	**35.93**
Kodak24	29.13	29.42	29.92	**30.10**
average	31.23	32.66	33.12	**33.61**
standard deviation	2.73	2.84	2.81	2.81

Table A.14: Details of the performances of the reconstruction methods on the Mask2 in term of C-PSNR (dB), computed between the reconstructed color images and the reference color images on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	0.8996	0.9266	0.9318	**0.9398**
Kodak2	0.9218	0.9301	0.9376	**0.9415**
Kodak3	0.9614	0.9681	0.9697	**0.9719**
Kodak4	0.9416	0.9438	0.9479	**0.9497**
Kodak5	0.9052	0.9348	0.9393	**0.9418**
Kodak6	0.9284	0.9387	0.9446	**0.9525**
Kodak7	0.9573	0.9740	0.9743	**0.9768**
Kodak8	0.8986	0.9360	0.9430	**0.9541**
Kodak9	0.9603	0.9718	0.9736	**0.9764**
Kodak10	0.9577	0.9684	0.9716	**0.9752**
Kodak11	0.9287	0.9467	0.9520	**0.9572**
Kodak12	0.9585	0.9651	0.9684	**0.9723**
Kodak13	0.8999	0.9057	0.9115	**0.9154**
Kodak14	0.9236	0.9365	0.9401	**0.9446**
Kodak15	0.9454	0.9483	0.9541	**0.9574**
Kodak16	0.9444	0.9524	0.9565	**0.9634**
Kodak17	0.9565	0.9651	0.9672	**0.9679**
Kodak18	0.9275	0.9388	0.9412	**0.9450**
Kodak19	0.9387	0.9532	0.9577	**0.9626**
Kodak20	0.9594	0.9685	0.9700	**0.9714**
Kodak21	0.9415	0.9527	0.9557	**0.9598**
Kodak22	0.9400	0.9474	0.9509	**0.9549**
Kodak23	0.9668	0.9718	0.9735	**0.9744**
Kodak24	0.9333	0.9449	0.9490	**0.9520**
average	0.9373	0.9496	0.9534	**0.9574**
standard deviation	0.0212	0.0173	0.0159	0.0150

Table A.15: Details of the performances of the reconstruction methods on the Mask2 in term of SSIM, computed between the reconstructed color images and the reference color images on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	0.9612	0.9739	0.9752	**0.9789**
Kodak2	0.9734	0.9772	0.9803	**0.9822**
Kodak3	0.9879	0.9902	0.9907	**0.9916**
Kodak4	0.9802	0.9816	0.9829	**0.9840**
Kodak5	0.9711	0.9796	0.9815	**0.9822**
Kodak6	0.9701	0.9751	0.9779	**0.9818**
Kodak7	0.9869	0.9928	0.9929	**0.9937**
Kodak8	0.9675	0.9793	0.9822	**0.9863**
Kodak9	0.9869	0.9916	0.9922	**0.9934**
Kodak10	0.9866	0.9903	0.9915	**0.9928**
Kodak11	0.9766	0.9824	0.9846	**0.9867**
Kodak12	0.9855	0.9881	0.9896	**0.9912**
Kodak13	0.9633	0.9659	0.9686	**0.9700**
Kodak14	0.9764	0.9807	0.9821	**0.9836**
Kodak15	0.9836	0.9844	0.9863	**0.9877**
Kodak16	0.9790	0.9824	0.9843	**0.9876**
Kodak17	0.9870	0.9900	0.9907	**0.9910**
Kodak18	0.9763	0.9805	0.9817	**0.9830**
Kodak19	0.9750	0.9824	0.9851	**0.9876**
Kodak20	0.9881	0.9909	0.9914	**0.9920**
Kodak21	0.9819	0.9857	0.9868	**0.9884**
Kodak22	0.9781	0.9812	0.9833	**0.9847**
Kodak23	0.9902	0.9923	0.9928	**0.9932**
Kodak24	0.9782	0.9820	0.9837	**0.9847**
average	0.9788	0.9834	0.9849	**0.9866**
standard deviation	0.0081	0.0066	0.0060	0.0056

Table A.16: Details of the performances of the reconstruction methods on the Mask2 in term of M-SSIM, computed between the reconstructed color images and the reference color images on the Kodak database [11].

	BI	EDICentral	EDILine	SG
Kodak1	9.188	7.161	5.850	**4.742**
Kodak2	3.279	2.457	1.720	**1.357**
Kodak3	1.531	1.314	1.162	**0.874**
Kodak4	2.639	2.282	1.775	**1.463**
Kodak5	12.835	11.164	10.232	**9.698**
Kodak6	5.744	4.874	4.107	**3.018**
Kodak7	3.264	2.066	1.952	**1.580**
Kodak8	14.026	11.319	9.666	**7.737**
Kodak9	2.655	1.707	1.526	**1.149**
Kodak10	2.419	1.827	1.654	**1.381**
Kodak11	4.832	3.778	3.258	**2.909**
Kodak12	1.082	0.810	0.682	**0.541**
Kodak13	9.341	9.830	8.818	**8.080**
Kodak14	5.937	5.193	4.592	**4.100**
Kodak15	2.691	2.440	2.023	**1.823**
Kodak16	4.399	3.806	3.171	**2.069**
Kodak17	2.867	2.384	2.127	**2.026**
Kodak18	5.380	5.422	4.906	**4.562**
Kodak19	5.106	4.221	3.384	**2.688**
Kodak20	3.101	2.457	2.204	**1.963**
Kodak21	6.557	5.953	5.194	**4.497**
Kodak22	2.685	2.641	2.144	**1.954**
Kodak23	1.469	1.209	1.032	**0.906**
Kodak24	4.947	5.041	4.471	**4.081**
average	4.915	4.223	3.652	**3.133**
standard deviation	3.400	3.025	2.695	2.435

Table A.17: Details of the performances of the reconstruction methods on the Mask2 in term of Zipper metric (%), computed between the reconstructed color images and the reference color images on the Kodak database [11].

A.9 Metrics results of color pixel reconstruction algorithms applied on the McMaster database for the Mask2

	BI	EDICentral	EDILine	SG
McM1	23.47	25.09	25.65	**26.05**
McM2	27.16	29.11	29.64	**30.28**
McM3	23.34	24.23	24.77	**25.1**
McM4	23.7	26.32	26.75	**27.41**
McM5	28.06	29.48	29.75	**30.19**
McM6	30.4	32.02	32.51	**33.17**
McM7	27.97	28.83	29.53	**29.66**
McM8	26.72	27.3	28.23	**28.97**
McM9	28.14	29.76	30.15	**30.56**
McM10	30.6	32.08	32.5	**32.89**
McM11	31.51	32.55	33.17	**33.37**
McM12	28.49	28.84	29.98	**30.76**
McM13	32.83	34.81	35.03	**35.93**
McM14	32.32	34.12	34.41	**34.52**
McM15	32.36	33.5	**34.2**	34.04
McM16	25.21	26.34	26.6	**26.91**
McM17	27.5	28.72	29.21	**29.54**
McM18	24.58	28.98	29.36	**29.79**
average	28.02	29.56	30.08	**30.51**
standard deviation	3.15	3.09	3.05	3.02

Table A.18: Details of the performances of the reconstruction methods on the Mask2 in term of PSNR (dB), computed between the reconstructed pixels in the CFA image and the reference pixels in the reference CFA image on the McMaster database [25].

	BI	EDICentral	EDILine	SG
McM1	26.50	28.52	29.05	**29.48**
McM2	30.39	32.91	33.31	**33.96**
McM3	26.89	28.30	28.64	**28.96**
McM4	26.39	29.63	29.82	**30.38**
McM5	31.76	33.59	33.74	**34.17**
McM6	34.06	36.04	36.43	**37.11**
McM7	31.96	33.72	34.29	**34.47**
McM8	30.82	32.04	32.83	**33.61**
McM9	30.92	32.92	33.23	**33.66**
McM10	33.19	34.84	35.32	**35.62**
McM11	34.36	35.78	36.14	**36.32**
McM12	32.18	33.24	34.13	**34.83**
McM13	36.00	38.32	38.48	**39.39**
McM14	34.91	37.03	37.20	**37.27**
McM15	35.10	36.61	**37.13**	36.92
McM16	27.94	29.10	29.29	**29.64**
McM17	29.72	30.83	31.31	**31.71**
McM18	27.37	31.82	32.15	**32.71**
average	31.14	33.07	33.47	**33.90**
standard deviation	3.14	3.01	3.00	2.99

Table A.19: Details of the performances of the reconstruction methods on the Mask2 in term of C-PSNR (dB), computed between the reconstructed color images and the reference color images on the McMaster database [25]

	BI	EDICentral	EDILine	SG
McM1	0.9061	0.9364	0.9430	**0.9471**
McM2	0.9197	0.9398	0.9454	**0.9498**
McM3	0.9203	0.9410	0.9462	**0.9490**
McM4	0.9437	0.9696	0.9721	**0.9748**
McM5	0.9428	0.9571	0.9593	**0.9625**
McM6	0.9486	0.9643	0.9671	**0.9711**
McM7	0.9308	0.9415	0.9473	**0.9483**
McM8	0.9464	0.9594	0.9641	**0.9689**
McM9	0.9396	0.9585	0.9620	**0.9639**
McM10	0.9401	0.9522	0.9570	**0.9591**
McM11	0.9448	0.9551	0.9586	**0.9594**
McM12	0.9528	0.9629	0.9677	**0.9712**
McM13	0.9743	0.9809	0.9822	**0.9834**
McM14	0.9564	0.9642	0.9663	**0.9661**
McM15	0.9473	0.9525	**0.9571**	0.955
McM16	0.9212	0.9411	0.9442	**0.9477**
McM17	0.8834	0.9078	0.9172	**0.9245**
McM18	0.9083	0.9473	0.9523	**0.9574**
average	0.9348	0.9518	0.9561	**0.9589**
standard deviation	0.0216	0.0161	0.0144	0.0135

Table A.20: Details of the performances of the reconstruction methods on the Mask2 in term of SSIM, computed between the reconstructed color images and the reference color images on the McMaster database [25].

	BI	EDICentral	EDILine	SG
McM1	0.9753	0.9841	0.9859	**0.9874**
McM2	0.9805	0.9853	0.9869	**0.9881**
McM3	0.9754	0.9821	0.9838	**0.9847**
McM4	0.9839	0.9912	0.9919	**0.9927**
McM5	0.9852	0.9890	0.9898	**0.9906**
McM6	0.9878	0.9915	0.9924	**0.9934**
McM7	0.9812	0.9841	0.9860	**0.9862**
McM8	0.9856	0.9880	0.9899	**0.9915**
McM9	0.9847	0.9894	0.9907	**0.9912**
McM10	0.9854	0.9885	0.9900	**0.9906**
McM11	0.9865	0.9891	0.9901	**0.9907**
McM12	0.9858	0.9876	0.9902	**0.9916**
McM13	0.9916	0.9941	0.9946	**0.9952**
McM14	0.9896	0.9921	0.9926	**0.9927**
McM15	0.9872	0.9884	**0.9898**	0.9895
McM16	0.9789	0.9842	0.9854	**0.9865**
McM17	0.9700	0.9763	0.9794	**0.9812**
McM18	0.9762	0.9855	0.9873	**0.9889**
average	0.9828	0.9872	0.9887	**0.9896**
standard deviation	0.0057	0.0042	0.0037	0.0034

Table A.21: Details of the performances of the reconstruction methods on the Mask2 in term of M-SSIM, computed between the reconstructed color images and the reference color images on the McMaster database [25].

	BI	EDICentral	EDILine	SG
McM1	9.332	6.585	5.860	**5.438**
McM2	7.362	4.851	4.369	**3.601**
McM3	8.150	6.980	6.308	**5.760**
McM4	9.020	4.892	4.567	**4.005**
McM5	4.681	3.252	3.122	**2.803**
McM6	3.691	2.414	2.138	**1.765**
McM7	5.003	4.929	4.093	**3.802**
McM8	5.684	4.634	4.183	**3.457**
McM9	5.718	3.484	3.222	**2.834**
McM10	3.793	2.254	1.989	**1.568**
McM11	3.325	1.870	1.581	**1.403**
McM12	3.704	2.688	2.273	**1.972**
McM13	1.489	0.863	0.774	**0.587**
McM14	2.141	1.180	1.065	**1.002**
McM15	1.955	1.219	1.010	**0.988**
McM16	6.731	4.340	3.804	**3.061**
McM17	10.639	7.434	6.381	**5.282**
McM18	9.099	3.370	2.916	**2.473**
average	5.640	3.736	3.314	**2.878**
standard deviation	2.799	1.994	1.772	1.574

Table A.22: Details of the performances of the reconstruction methods on the Mask2 in term of Zipper metric (%), computed between the reconstructed color images and the reference color images on the McMaster database [25].

A.10 Metrics results of color pixel reconstruction algorithms applied on the Hdr+ burst database for the Mask1

	BI	EDICentral	EDILine	SG
HDR+ Burst 1	27.71	27.75	27.85	**28.76**
HDR+ Burst 2	24.38	24.23	24.68	**25.17**
HDR+ Burst 3	27.05	26.88	27.39	**28.03**
HDR+ Burst 4	33.7	34.49	34.72	**36.29**
HDR+ Burst 5	33.46	33.21	33.61	**34.2**
HDR+ Burst 6	32.33	32.04	32.48	**32.92**
HDR+ Burst 7	31.13	31.46	31.8	**32.87**
HDR+ Burst 8	35.13	35.67	35.85	**36.81**
HDR+ Burst 9	35.01	35.26	35.58	**36.18**
HDR+ Burst 10	27.42	27.51	28.04	**28.93**
HDR+ Burst 11	24.73	24.47	25	**25.5**
HDR+ Burst 12	26.06	26.07	26.56	**27.25**
average	29.84	29.92	30.30	31.08
standard deviation	3.86	4.05	3.97	4.11

Table A.23: Details of the performances of the reconstruction methods on the Mask1 in term of PSNR (dB), computed between the reconstructed CFA images and the reference CFA images on the HDR+ burst database [26].

	BI	EDICentral	EDILine	SG
HDR+ Burst 1	36.14	36.21	36.31	**37.20**
HDR+ Burst 2	30.41	30.35	30.68	**31.11**
HDR+ Burst 3	31.40	31.40	31.70	**32.18**
HDR+ Burst 4	40.21	41.25	41.39	**42.76**
HDR+ Burst 5	41.26	41.21	41.46	**41.99**
HDR+ Burst 6	39.22	39.24	39.53	**39.95**
HDR+ Burst 7	37.27	37.68	37.96	**38.88**
HDR+ Burst 8	41.40	42.24	42.33	**43.25**
HDR+ Burst 9	40.79	41.10	41.34	**41.74**
HDR+ Burst 10	33.16	33.39	33.75	**34.45**
HDR+ Burst 11	30.83	30.69	31.03	**31.44**
HDR+ Burst 12	32.04	32.18	32.50	**33.05**
average	36.18	36.41	36.67	**37.33**
standard deviation	4.20	4.41	4.35	4.49

Table A.24: Details of the performances of the reconstruction methods on the Mask1 in term of C-PSNR (dB), computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

	BI	EDICentral	EDILine	SG
HDR+ Burst 1	0.9753	0.9749	0.9759	**0.9782**
HDR+ Burst 2	0.9453	0.9444	0.9488	**0.9537**
HDR+ Burst 3	0.9519	0.9509	0.9547	**0.9598**
HDR+ Burst 4	0.9837	0.9851	0.9858	**0.9876**
HDR+ Burst 5	0.9841	0.9834	0.9844	**0.9857**
HDR+ Burst 6	0.9774	0.9760	0.9778	**0.9794**
HDR+ Burst 7	0.9778	0.9773	0.9792	**0.9814**
HDR+ Burst 8	0.9854	0.9856	0.9864	**0.9878**
HDR+ Burst 9	0.9808	0.9801	0.9814	**0.9824**
HDR+ Burst 10	0.9649	0.9661	0.9686	**0.9733**
HDR+ Burst 11	0.9504	0.9492	0.9532	**0.9580**
HDR+ Burst 12	0.9623	0.9633	0.9659	**0.9702**
average	0.9699	0.9697	0.9718	0.9748
standard deviation	0.0138	0.0141	0.0128	0.0114

Table A.25: Details of the performances of the reconstruction methods on the Mask1 in term of SSIM, computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

HDR+ Burst 1	0.9907	0.9905	0.9908	**0.9913**
HDR+ Burst 2	0.9852	0.9846	0.9859	**0.9872**
HDR+ Burst 3	0.9887	0.9880	0.9891	**0.9904**
HDR+ Burst 4	0.9963	0.9965	0.9967	**0.9971**
HDR+ Burst 5	0.9961	0.9959	0.9961	**0.9965**
HDR+ Burst 6	0.9947	0.9943	0.9948	**0.9951**
HDR+ Burst 7	0.9946	0.9945	0.9950	**0.9954**
HDR+ Burst 8	0.9968	0.9969	0.9971	**0.9973**
HDR+ Burst 9	0.9953	0.9951	0.9955	**0.9957**
HDR+ Burst 10	0.9923	0.9923	0.9930	**0.9940**
HDR+ Burst 11	0.9882	0.9877	0.9888	**0.9900**
HDR+ Burst 12	0.9911	0.9913	0.9920	**0.9931**
average	0.9925	0.9923	0.9929	0.9936
standard deviation	0.0036	0.0038	0.0034	0.0031

Table A.26: Details of the performances of the reconstruction methods on the Mask1 in term of M-SSIM, computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

HDR+ Burst 1	4.717	4.778	4.685	**3.714**
HDR+ Burst 2	10.776	11.617	10.647	**9.287**
HDR+ Burst 3	4.763	4.843	4.435	**3.583**
HDR+ Burst 4	1.862	1.522	1.470	**1.025**
HDR+ Burst 5	1.708	1.750	1.570	**1.282**
HDR+ Burst 6	2.683	2.944	2.676	**2.289**
HDR+ Burst 7	2.173	2.282	1.976	**1.608**
HDR+ Burst 8	1.037	0.897	0.837	**0.654**
HDR+ Burst 9	0.945	1.047	0.883	**0.783**
HDR+ Burst 10	5.841	5.809	5.313	**4.287**
HDR+ Burst 11	11.601	12.566	11.488	**10.236**
HDR+ Burst 12	8.484	9.031	8.233	**7.140**
average	4.716	4.924	4.518	3.824
standard deviation	3.596	3.923	3.596	3.203

Table A.27: Details of the performances of the reconstruction methods on the Mask1 in term of Zipper metric (%), computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

A.11 Metrics results of color pixel reconstruction algorithms applied on the Hdr+ burst database for the Mask2

	BI	EDICentral	EDILine	SG
HDR+ Burst 1	27.84	31.99	33.85	**36.23**
HDR+ Burst 2	23.71	24.05	24.58	**24.83**
HDR+ Burst 3	25.52	25.95	26.45	**26.77**
HDR+ Burst 4	31.66	34.35	34.83	**35.39**
HDR+ Burst 5	31.70	31.96	32.14	**32.47**
HDR+ Burst 6	31.22	31.60	31.91	**32.25**
HDR+ Burst 7	29.40	31.12	31.14	**31.59**
HDR+ Burst 8	32.74	34.87	35.02	**35.55**
HDR+ Burst 9	32.62	34.61	34.81	**35.33**
HDR+ Burst 10	25.97	26.96	27.54	**27.93**
HDR+ Burst 11	23.92	24.14	24.67	**24.93**
HDR+ Burst 12	24.74	25.32	25.69	**26.04**
average	28.42	29.74	30.22	30.78
standard deviation	3.38	4.00	3.98	4.25

Table A.28: Details of the performances of the reconstruction methods on the Mask2 in term of PSNR (dB), computed between the reconstructed CFA images and the reference CFA images on the HDR+ burst database [26].

	BI	EDICentral	EDILine	SG
HDR+ Burst 1	32.36	36.93	38.60	**40.70**
HDR+ Burst 2	27.56	28.26	28.62	**28.89**
HDR+ Burst 3	28.51	29.32	29.62	**29.96**
HDR+ Burst 4	35.24	38.51	38.80	**39.42**
HDR+ Burst 5	36.72	37.21	37.26	**37.61**
HDR+ Burst 6	35.20	36.12	36.35	**36.73**
HDR+ Burst 7	32.67	34.68	34.67	**35.11**
HDR+ Burst 8	35.94	38.63	38.71	**39.29**
HDR+ Burst 9	35.26	37.54	37.71	**38.33**
HDR+ Burst 10	29.54	30.93	31.31	**31.71**
HDR+ Burst 11	27.57	28.21	28.57	**28.85**
HDR+ Burst 12	28.31	29.23	29.47	**29.82**
average	32.07	33.80	34.14	34.70
standard deviation	3.43	4.06	4.10	4.36

Table A.29: Details of the performances of the reconstruction methods on the Mask2 in term of C-PSNR (dB), computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

	BI	EDICentral	EDILine	SG
HDR+ Burst 1	0.9583	0.9743	0.9789	**0.9827**
HDR+ Burst 2	0.9180	0.9311	0.9362	**0.9402**
HDR+ Burst 3	0.9241	0.9362	0.9401	**0.9447**
HDR+ Burst 4	0.9696	0.9810	0.9819	**0.9831**
HDR+ Burst 5	0.9691	0.9725	0.9731	**0.9749**
HDR+ Burst 6	0.9593	0.9636	0.9649	**0.9674**
HDR+ Burst 7	0.9528	0.9638	0.9643	**0.9662**
HDR+ Burst 8	0.9646	0.9757	0.9766	**0.9793**
HDR+ Burst 9	0.9570	0.9657	0.9672	**0.9695**
HDR+ Burst 10	0.9309	0.9519	0.9548	**0.9589**
HDR+ Burst 11	0.9142	0.9293	0.9347	**0.9396**
HDR+ Burst 12	0.9258	0.9419	0.9440	**0.9488**
average	0.9453	0.9572	0.9597	0.9629
standard deviation	0.0201	0.0177	0.0165	0.0155

Table A.30: Details of the performances of the reconstruction methods on the Mask2 in term of SSIM, computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

HDR+ Burst 1	0.9849	0.9911	0.9932	**0.9950**
HDR+ Burst 2	0.9719	0.9768	0.9783	**0.9798**
HDR+ Burst 3	0.9747	0.9788	0.9806	**0.9823**
HDR+ Burst 4	0.9919	0.9950	0.9953	**0.9957**
HDR+ Burst 5	0.9906	0.9917	0.9918	**0.9925**
HDR+ Burst 6	0.9885	0.9900	0.9904	**0.9911**
HDR+ Burst 7	0.9879	0.9908	0.9910	**0.9916**
HDR+ Burst 8	0.9910	0.9941	0.9944	**0.9952**
HDR+ Burst 9	0.9889	0.9915	0.9919	**0.9925**
HDR+ Burst 10	0.9805	0.9863	0.9873	**0.9886**
HDR+ Burst 11	0.9746	0.9790	0.9809	**0.9825**
HDR+ Burst 12	0.9777	0.9830	0.9839	**0.9854**
average	0.9836	0.9873	0.9883	0.9893
standard deviation	0.0070	0.0061	0.0056	0.0053

Table A.31: Details of the performances of the reconstruction methods on the Mask2 in term of M-SSIM, computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

HDR+ Burst 1	3.214	1.677	1.107	**0.670**
HDR+ Burst 2	9.447	9.402	8.358	**7.583**
HDR+ Burst 3	3.729	3.370	3.001	**2.648**
HDR+ Burst 4	2.308	1.188	1.116	**0.885**
HDR+ Burst 5	1.669	1.714	1.637	**1.394**
HDR+ Burst 6	1.701	1.760	1.495	**1.346**
HDR+ Burst 7	2.754	1.915	1.792	**1.523**
HDR+ Burst 8	1.921	1.050	0.890	**0.715**
HDR+ Burst 9	2.290	1.362	1.254	**1.051**
HDR+ Burst 10	6.593	4.932	4.451	**3.895**
HDR+ Burst 11	10.132	9.663	8.906	**8.209**
HDR+ Burst 12	7.211	7.023	6.401	**5.814**
average	4.414	3.755	3.368	2.978
standard deviation	2.963	3.089	2.826	2.636

Table A.32: Details of the performances of the reconstruction methods on the Mask2 in term of Zipper metric (%), computed between the reconstructed color images and the reference color images on the HDR+ burst database [26].

A.12 Metrics results details of joint reconstruction applied on the Middlebury database

	SG	SG + joint reconstruction homogeneous reflectance (0.4)	SG + joint reconstruction reflectance as R channel
Middlebury1	32.84	32.84	32.84
Middlebury2	29.37	29.37	29.38
Middlebury3	31.75	31.73	31.7
Middlebury4	30.22	30.2	30.19
Middlebury5	32.93	32.92	32.93
Middlebury6	32.67	32.65	32.61
average	31.63	31.62	31.61
standard deviation	1.51	1.51	1.50

Table A.33: Details of the performances of the joint reconstruction methods on the Mask1 in term of PSNR (dB), computed between the reconstructed CFA images and the reference CFA images on the Middlebury database [23][24].

	SG	SG + joint reconstruction homogeneous reflectance (0.4)	SG + joint reconstruction reflectance as R channel
Middlebury 1	39.18	39.18	39.23
Middlebury 2	37.06	37.08	37.06
Middlebury 3	37.59	37.59	37.59
Middlebury 4	36.97	36.95	36.92
Middlebury 5	39.46	39.46	39.49
Middlebury 6	38.45	38.42	38.41
average	38.12	38.11	38.12
standard deviation	1.07	1.07	1.10

Table A.34: Details of the performances of the joint reconstruction methods on the Mask1 in term of C-PSNR (dB), computed between the reconstructed color images and the reference color images on the Middlebury database [23][24].

	SG	SG + joint reconstruction homogeneous reflectance (0.4)	SG + joint reconstruction reflectance as R channel
Middlebury 1	0.9824	0.9825	0.9826
Middlebury 2	0.9689	0.9689	0.9690
Middlebury 3	0.9787	0.9788	0.9788
Middlebury 4	0.9718	0.9717	0.9718
Middlebury 5	0.9792	0.9793	0.9792
Middlebury 6	0.9772	0.9771	0.9775
average	0.9763	0.9764	0.9765
standard deviation	0.0051	0.0051	0.0051

Table A.35: Details of the performances of the joint reconstruction methods on the Mask1 in term of SSIM, computed between the reconstructed color images and the reference color images on the Middlebury database [23][24].

	SG	SG + joint reconstruction homogeneous reflectance (0.4)	SG + joint reconstruction reflectance as R channel
Middlebury 1	0.9959	0.9959	0.9960
Middlebury 2	0.9910	0.9910	0.9911
Middlebury 3	0.9955	0.9955	0.9956
Middlebury 4	0.9905	0.9905	0.9906
Middlebury 5	0.9951	0.9951	0.9951
Middlebury 6	0.9940	0.9939	0.9941
average	0.9937	0.9937	0.9937
standard deviation	0.0024	0.0023	0.0024

Table A.36: Details of the performances of the joint reconstruction methods on the Mask1 in term of M-SSIM, computed between the reconstructed color images and the reference color images on the Middlebury database [23][24].

	SG	SG + joint reconstruction homogeneous reflectance (0.4)	SG + joint reconstruction reflectance as R channel
Middlebury 1	2.884	2.864	2.840
Middlebury 2	4.112	4.232	4.235
Middlebury 3	2.402	2.286	2.392
Middlebury 4	4.776	4.929	4.878
Middlebury 5	1.969	1.945	1.909
Middlebury 6	3.577	3.570	3.581
average	3.287	3.304	3.306
standard deviation	1.064	1.153	1.132

Table A.37: Details of the performances of the joint reconstruction methods on the Mask1 in term of Zipper metric (%), computed between the reconstructed color images and the reference color images on the Middlebury database [23][24].

Bibliography

[1] Ahmed Nabil Belbachir. *Smart Cameras*. 2010.

[2] Y. Kagawa and H. Iwamoto. 3D Integration Technologies for the Stacked CMOS Image Sensors. *IEEE 2019 International 3D Systems Integration Conference, 3DIC 2019*, 2019.

[3] Rastislav Lukac and Konstantinos N. Plataniotis. Color filter arrays: Design and performance analysis. *IEEE Transactions on Consumer Electronics*, 51(4):1260–1267, 2005.

[4] John F Hamilton and John F Compton. United States Patent: Image Sensor With Improved Light Sensitivity (Nb US 8,139,130 B2), 2012.

[5] Yusuke Monno, Sunao Kikuchi, Masayuki Tanaka, and Masatoshi Okutomi. A Practical One-Shot Multispectral Imaging System Using a Single Image Sensor. *IEEE Transactions on Image Processing*, 24(10):3048–3059, 2015.

[6] Kim Seong-Jin, J.D.K. Kim, Han Sang-Wook, Kang Byongmin, Lee Keechang, and Kim Chang-Yeong. A 640×480 Image Sensor with Unified Pixel Architecture for 2D/3D Imaging in 0.11µm CMOS. *Symposium on VLSI Circuits Digest of Technical Papers*, pages 92–93, 2011.

[7] Wonjoo Kim, Wang Yibing, Ilia Ovsiannikov, Seunghoon Lee, Yoondong Park, Chilhee Chung, and Eric Fossum. A 1.5Mpixel RGBZ CMOS image sensor for simultaneous color and range image capture. *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, 55(June 2011):392–393, 2012.

[8] Lilong Shi, Ilia Ovsiannikov, Dong-ki Min, Yohwan Noh, Wanghyun Kim, and Sunhwa Jung. Demosaicing for RGBZ Sensor. *SPIE Proceedings*, 8657:1–9, 2013.

[9] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multi-scale Structural Similarity For image Quality Assessment. *IEEE Asilomar Conference on Signals, Systems and Computer*, 2:9–13, 2003.

[10] Bryce Bayer. United States Patent: Colour Imaging Array (Nb 3,971,065), 1976.

[11] Kodak color image dataset. http://r0k.us/graphics/kodak/.

[12] David Cok. Signal processing method and apparatus for producing interpolated chrominance values a sampled color image signal (Nb 4,642,678), 1987.

[13] Bahadir K. Gunturk, John Glotzbach, Yucel Altunbasak, Ronald W. Schafer, and Russell M. Mersereau. Demosaicking: Color filter array interpolation. *IEEE Signal Processing Magazine*, 22(1):44–54, 2005.

[14] Daisuke Kiku, Yusuke Monno, Masayuki Tanaka, and Masatoshi Okutomi. Residual interpolation for color image demosaicking. *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, 2:2304–2308, 2013.

[15] Claude Laroche and Mark Prescott. Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients, 1994.

[16] Ron Kimmel. Demosaicing: Image reconstruction from color CCD samples. *IEEE Transactions on Image Processing*, 1406(9):610–622, 1998.

[17] David Alleysson. 30 Ans De Démosaïçage. 21:561–582, 2005.

[18] Laurent Alacoque. Method for demosaicing a raw digital image, corresponding computer program and imaging or graphic circuit (number EP 2426639 B1), 2015.

[19] Hiroki Yamashita, Daisuke Sugimura, and Takayuki Hamamoto. Enhancing low-light color images using an RGB-NIR single sensor. *2015 Visual Communications and Image Processing, VCIP 2015*, pages 2–5, 2015.

[20] Yusuke Monno, Hayato Teranaka, Kazunori Yoshizaki, Masayuki Tanaka, and Masatoshi Okutomi. Single-Sensor RGB-NIR Imaging: High-Quality System Design and Prototype Implementation. *IEEE Sensors Journal*, 19(2):497–507, 2019.

[21] Hayato Teranaka, Yusuke Monno, Masayuki Tanaka, and Masatoshi Ok. Single-Sensor RGB and NIR Image Acquisition: Toward Optimal Performance by Taking Account of CFA Pattern, Demosaicking, and Color Correction. *Electronic Imaging*, 2016(18):1–6, 2016.

[22] H.S. Malvar, Li-wei He, and R. Cutler. High-quality linear interpolation for demosaicing of Bayer-patterned color images. *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3:iii–485–8, 2004.

[23] Heiko Hirschmüller and Daniel Scharstein. Evaluation of cost functions for stereo matching. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[24] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[25] Lei Zhang, Xiaolin Wu, Antoni Buades, and Xin Li. Color Demosaicking by Local Directional Interpolation and Nonlocal Adaptive Thresholding. *Electronic Imaging*, pages 1–29, 2011.

[26] Andrew Adams Samuel W. Hasinoff, Dillon Sharlet, Ryan Geiss and Marc Levoy Levoy, Jonathan T. Barron, Florian Kainz, Jiawen Chen. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics*, 35, 2016.

[27] Fumihito Yasuma, Tomoo Mitsunaga, Daisuke Iso, and Shree K Nayar. Generalized Assorted Pixel Camera: Post-Capture Control of Resolution, Dynamic Range and Spectrum. Technical report, 2008.

[28] Abbas El Gamal. High Dynamic Range Image Sensors. *International Solid-State Circuits Conference*, 209, 2002.

[29] Alessandro Bevilacqua, Luigi Di Stefano, and Pietro Azzari. People tracking using a Time-of-Flight depth sensor. *Proceedings - IEEE International Conference on Video and Signal Based Surveillance 2006, AVSS 2006*, 2006.

[30] Microsoft. Microsoft Kinect.

[31] K Litomisky. Consumer rgb-d cameras and their applications. Technical report, 2012.

[32] Maggie Tillman. What is Apple Face ID and how does it work?, 2020.

[33] Robert J Gove. CMOS image sensor technology advances for mobile devices. In *High Performance Silicon Imaging*, pages 185–240. Elsevier Ltd., 2 edition, 2020.

[34] Dave Litwiller. CCD vs. CMOS: Facts and fiction. *Photonics Spectra*, 35(1):154–158, 2001.

[35] P Weckler. Operation of p-n Junction Photodetectors in a Photon Flux Integrating Mode. *IEEE Journal of Solid-State Circuits*, (3), 1967.

[36] Peter J W Noble. Self-Scanned Silicon Image Detector Arrays. *IEEE transactions on electron Devices*, ED-15(April):202–209, 1968.

[37] Eric R Fossum. CMOS Image Sensors : Electronic Camera-On-A-Chip. *IEEE TRANSACTIONS ON ELECTRON DEVICES*, 44(10):1689–1698, 1997.

[38] Kuo Liang Chung, Tzu Hsien Chan, and Szu Ni Chen. Effective three-stage demosaicking method for rgbw cfa images using the iterative error-compensation based approach. *Sensors (Switzerland)*, 20(14):1–12, 2020.

[39] Peter Corcoran and Petronel Bigioi. *Consumer Imaging I: Processing Pipeline, Focus, and Exposure*. 2016.

[40] Angelo Bosco, Arcangelo Bruna, Filippo Naccari, and Ivana Guarneri. Hardware/software solution for high precision defect correction in digital image sensors. *Proceedings of the International Symposium on Consumer Electronics, ISCE*, pages 3–6, 2008.

[41] Thomas S. Huang, George J. Yang, and Gregory Y. Tang. A Fast Two-Dimensional Median Filtering Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.

[42] Madhu S. Nair, K. Revathy, and Rao Tatavarti. An improved decision-based algorithm for impulse noise removal. *Proceedings - 1st International Congress on Image and Signal Processing, CISP 2008*, 1:426–431, 2008.

[43] C Tomasi and R Manduchi. Bilateral Filtering for Gray and Color Images. *International Conference on Computer Vision*, 1998.

[44] Angelo Bosco, Bruna Arcangelo, Gaetano Santoro, and Paolo Vivirito. Joint gaussian noise reduction and defects correction in raw digital Iimages. *Nordic Signal Processing Symposium 2004*, 6:109–112, 2004.

[45] Abhishek Jain and Richa Gupta. A survey on defect and noise detection and correction algorithms in image sensors. *Conference Proceeding - 2015 International Conference on Advances in Computer Engineering and Applications, ICACEA 2015*, pages 754–759, 2015.

[46] Fritz Merkle, Tinwai D Lam, and Cottle Road. Imaging System Response Linearization and Shading Correction. *IEEE International Conference on Robotics and Automation*, pages 204–209, 1984.

[47] Yuanjie Zheng, Stephen Lin, Chandra Kambhamettu, Jingyi Yu, and Sing Bing Kang. Single-image vignetting correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2243–2256, 2009.

[48] Varuna De Silva, Viacheslav Chesnokov, and Daniel Larkin. A Novel Adaptive Shading Correction Algorithm for Camera Systems. *IS&T International Symposium on Electronic Imaging*, 2016.

[49] G Zapryanov, D Ivanova, and I Nikolova. Automatic White Balance Algorithms for Digital Still Cameras–a Comparative Study. *Information Technologies and Control*, (January 2012):16–22, 2012.

[50] Ching Chih Weng, Homer Chen, and Chiou Shann Fuh. A novel automatic white balance method for digital still cameras. *Proceedings - IEEE International Symposium on Circuits and Systems*, pages 3801–3804, 2005.

[51] Stephen Wolf. Color Correction Matrix for Digital Still and Video Imaging Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(12):4, 2003.

[52] Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research and Application*, 30(1):21–30, 2005.

[53] F. Drago, K. Myszkowski, T. Annen, and N. Chiba. Adaptive Logarithmic Mapping for Displaying High Contrast Scenes. *Computer Graphics Forum*, 22(3):419–426, 2003.

[54] A Artusi, A O Aky, B Roch, D Michael, Y Chrysanthou, and A Chalmers. Selective Local tone Mapping. *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, pages 2309–2313, 2013.

[55] Charles A. Poynton. 'Gamma' and its disguises: The nonlinear mappings of intensity in perception, CRTs, film, and video. *SMPTE Journal*, 102(12):1099–1108, 1993.

[56] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient Large-Scale Stereo Matching Efficient Large-Scale Stereo Matching. *Asian Conference on Computer Vision*, (May 2014):25–38, 2010.

[57] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 2003.

[58] Edoardo Charbon. Introduction to Time-of-Flight Imaging. *Sensors*, pages 610–613, 2014.

[59] Francescopaolo Mattioli, Della Rocca, Tarek Al Abbas, Neale A W Dutton, and Robert K Henderson. A High Dynamic Range SPAD Pixel for Time of Flight Imaging. *SENSORS, 2017 IEEE*, pages 7–9, 2017.

[60] Neale A.W. Dutton, Istvan Gyongy, Luca Parmesan, Salvatore Gnecchi, Neil Calder, Bruce R. Rae, Sara Pellegrini, Lindsay A. Grant, and Robert K. Henderson. A SPAD-based QVGA image sensor for single-photon counting and quanta imaging. *IEEE Transactions on Electron Devices*, 63(1):189–196, 2016.

[61] Seong Jin Kim, James D.K. Kim, Byongmin Kang, and Keechang Lee. A CMOS image sensor based on unified pixel architecture with time-division multiplexing scheme for color and depth image acquisition. *IEEE Journal of Solid-State Circuits*, 47(11):2834–2845, 2012.

[62] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Horaud. *Time of Flight Cameras : Principles , Methods , and Applications*. 2012.

[63] Taesub Jung, Yonghun Kwon, Sungyoung Seo, Min-sun Keel, Changkeun Lee, Sung-ho Choi, Sae-young Kim, and Sunghyuck Cho. A 4-tap global shutter pixel with enhanced IR sensitivity for VGA time-of-flight CMOS image sensors. pages 1–6, 2020.

[64] Sergi Foix, Guillem Alenya, and Carme Torras. Lock-in Time-of-Flight (ToF) Cameras: A Survey. *IEEE Sensors Journal*, 11(9):1917–1926, 2011.

[65] Eric R. Fossum and Donald B. Hondongwa. A review of the pinned photodiode for CCD and CMOS image sensors. *IEEE Journal of the Electron Devices Society*, 2(3):33–43, 2014.

[66] S. Burak Gokturk, Hakan Yalcin, and Cyrus Bamji. A time-of-flight depth sensor - System description, issues and solutions. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2004-Janua(January), 2004.

[67] David Droeschel, Dirk Holz, and Sven Behnke. Multi-frequency phase unwrapping for time-of-flight cameras. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, (October):1463–1469, 2010.

[68] Changpeng Ti, Ruigang Yang, and James Davis. Single-shot time-of-flight phase unwrapping using two modulation frequencies. *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, pages 667–675, 2016.

[69] Microsoft. Xbox 360 Kinect Sensor Manual. Technical report, 2010.

[70] Víctor Villena-Martínez, Andrés Fuster-Guilló, Jorge Azorín-López, Marcelo Saval-Calvo, Jeronimo Mora-Pascual, Jose Garcia-Rodriguez, and Alberto Garcia-Garcia. A Quantitative Comparison of Calibration Methods for RGB-D Sensors Using Different Technologies. *Sensors*, 17(2):243, 2017.

[71] Thomas Hach and Johannes Steurer. A Novel RGB-Z Camera for High-Quality Motion Picture Applications Categories and Subject Descriptors. *ACM International Conference Proceeding Series*, (Ldv), 2013.

[72] Seong Jin Kim, Byongmin Kang, James D.K. Kim, Keechang Lee, Chang Yeong Kim, and Kinam Kim. A 1920x1080 3.65μm-pixel 2D/3D image sensor with split and binning pixel structure in 0.11pm standard CMOS. *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, 55(3):396–397, 2012.

[73] Lilong SHI and Ilia OVSIANNIKOV. Demosaicing for RGBZ sensor (Nb US 2015/0022869 A1), 2015.

[74] Z Wang, a C Bovik, H R Sheikh, and E P Simmoncelli. Image quality assessment: form error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[75] Wenmiao Lu and Yap-Peng Tan. Color filter array demosaicking: new method and performance measures. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 12(10):1194–1210, 2003.

[76] Mark D Fairchild. *Color Appearance Models*. 1997.

[77] Bruce Justin Lindbloom. RGB/XYZ Matrices. http://www.brucelindbloom.com/.

[78] M Mahy, L Van Eycken, and A Oosterlinck. Evaluation of Uniform Color Spaces Developed after the Adoption of CIELAB and CIELUV. *Color Research and Application*, 19(2):105–121, 1994.

[79] Peter Dillon and Bryce Bayer. Signal processing for Discrete-Sample-Type-Color-Video Signal (Nb 4,148,059), 1979.

[80] Soo Chang Pei and Io Kuong Tam. Effective color interpolation in CCD color filter arrays using signal correlation. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(6):503–513, 2003.

[81] Kaiming He, Jian Sun, and Xiaoou Tang. Guided Image Filtering. *European Geriatric Medicine*, 8:S69, 2010.

[82] Robert Hibbard. Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients. (19), 1995.

[83] Keigo Hirakawa and Thomas W. Parks. Adaptive homogeneity-directed demosaicing algorithm. *IEEE Transactions on Image Processing*, 14(3):360–369, 2005.

[84] Ibrahim Pekkucuksen and Yucel Altunbasak. Gradient based threshold free color filter array interpolation. *IEEE International Conference on Image Processing*, pages 137–140, 2010.

[85] James E. Adams. Design of practical color filter array interpolation algorithms for digital cameras. *IEEE International Conference on Image Processing*, 1(x):488–492, 1998.

[86] Alain Horé and Djemel Ziou. An edge-sensing generic demosaicing algorithm with application to image resampling. *IEEE Transactions on Image Processing*, 20(11):3136–3150, 2011.

[87] Rastislav Lukac and Konstantinos N. Plataniotis. Universal demosaicking for imaging pipelines with an RGB color filter array. *Pattern Recognition*, 38(11):2208–2212, 2005.

[88] Keigo Hirakawa, Xiao Li Meng, and Patrick J. Wolfe. A framework for wavelet-based analysis and processing of color filter array images with applications to denoising and demosaicing. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1:597–600, 2007.

[89] Daniele Menon, Stefano Andriani, and Giancarlo Calvagno. Demosaicing with directional filtering and a posteriori decision. *IEEE Transactions on Image Processing*, 16(1):132–141, 2007.

[90] Bahadir K. Gunturk, Yucel Altunbasak, and Russell M. Mersereau. Color plane interpolation using alternating projections. *IEEE Transactions on Image Processing*, 11(9):997–1013, 2002.

[91] Xin Li. Demosaicing by Successive Approximation. *IEEE Transactions on Image Processing*, 14(3):370–379, 2005.

[92] Antoni Buades, Bartomeu Coll, Jean Michel Morel, and Catalina Sbert. Self-similarity driven color demosaicking. *IEEE Transactions on Image Processing*, 18(6):1192–1202, 2009.

[93] Daniel Stanley Tan, Wei Yang Chen, and Kai Lung Hua. DeepDemosaicking: Adaptive Image Demosaicking via Multiple Deep Fully Convolutional Networks. *IEEE Transactions on Image Processing*, 27(5):2408–2419, 2018.

[94] Shaojie Zhuo, Xiaopeng Zhang, Xiaoping Miao, and Terence Sim. Enhancing low light images using near infrared flash images. *International Conference on Image Processing(ICIP)*, pages 2537–2540, 2010.

[95] Yusuke Monno, Daisuke Kiku, Sunao Kikuchi, Masayuki Tanaka, and Masatoshi Okutomi. Multispectral demosaicking with novel guide image generation and residual interpolation. *2014 IEEE International Conference on Image Processing, ICIP 2014*, pages 645–649, 2014.

[96] Shinzo Koyama, Yuichi Inaba, Masahiro Kasano, and Takahiko Murata. A Day and Night Vision MOS Imager With Robust. 55(3):754–759, 2008.

[97] Rajeev Ramanath and Wesley E. Snyder. Adaptive demosaicking. *journal of electronic imaging*, 12:633–642, 2003.

[98] Boris Rodrigues, Marie Guillon, Nicolas Billon-pierron, Jean-baptiste Mancini, Benoit Giffard, Yvon Cazaux, Pierre Malinge, Patrice Waltz, Auguste Ngoua, Alisée Taluy, Sarah Kuster, Sylvain Joblot, François Roy, Guo-neng Lu, Nanotechnologies De Lyon, and Bernard Lyon. Indirect ToF Pixel integrating fast buried-channel transfer gates and gradual epitaxy , and enabling CDS. *International Image Sensor Workshop*, pages 4–7, 2017.

[99] Boris Rodrigues. *Développement d ' un pixel innovant de type « temps de vol » pour des capteurs d ' images 3D -CMOS*. PhD thesis, 2017.

[100] Andreas Breitbarth, Timothy Schardt, Cosima Kind, Julia Brinkmann, Paul-Gerald Dittrich, and Gunther Notni. Measurement accuracy and dependence on external influences of the iPhone X TrueDepth sensor. (October):7, 2019.

[101] Deepika Adlakha, Devender Adlakha, and Rohit Tanwar. Analytical Comparison between Sobel and Prewitt Edge Detection Techniques. 7(1):1482–1485, 2016.

[102] Juha Alakarhu. Image Sensors and Image Quality in Mobile Phones. *International Image Sensor Workshop, Ogunquit*, pages 7–18, 2007.