UNIVERSITÀ DI PADOVA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria delle Telecomunicazioni

Demosaicking di immagini a colori:

tecniche, analisi delle prestazioni, campionamento.

RELATORE:
Prof. Giancarlo Calvagno

Tesi di laurea di
LUCA GALEAZZO
Matr. N. 600981

Anno Accademico 2011-2012

In memoria di

Jacopo

Indice

1 Introduzione **5**

2 Tecniche di demosaicking **13**

2.1 Metodi euristici 14
2.2 Metodi edge-directed 22
2.3 Metodi nel dominio della frequenza 27
2.4 Metodi basati sulla trasformata Wavelet 32
2.5 Metodi di ricostruzione 34
2.6 Confronti 37

3 Analisi della qualità **41**

3.1 CIELAB 41
3.2 Zipper Effect 48
3.3 SSIM 50
3.4 Valutazione della qualità in assenza di riferimento 55
3.5 Analisi della qualità monocromatica 63

4 Demosaicking e campionamento **71**

4.1 Primo metodo 71
4.2 Secondo metodo 75
4.3 Terzo metodo 77
4.4 Quarto metodo 82
4.5 Confronto tra i metodi 85

Ringraziamenti **93**

Sommario

L'utilizzo di fotocamere e videocamere digitali oggigiorno è oramai diventato una consuetudine, in entrambi i dispositivi l'immagine viene raccolta tramite matrici di fotorivelatori, a tali sensori nella maggior parte delle volte sono anteposti dei filtri, denominati *Color Filter Array* (CFA), che lasciano passare sono una determinata lunghezza d'onda per ogni singolo sensore.

I CFA più diffusi sono quelli di Bayer, chiamati così in onore del loro creatore, il vantaggio derivante dal loro utilizzo è che non sono necessarie tre diverse matrici di sensori (sensori di Foveon) ma solamente una, come controparte l'immagine appena catturata è, dal punto di vista cromatico, solo una parte di quella totale, proprio come se fosse un mosaico.

Da qui nasce la necessità di ricostruzione o demosaicizzazione (*demosaicking*) dell'immagine originale per ottenere le tre componenti di colore per ogni pixel. In questa trattazione verranno affrontate le principali tecniche di demosaicking attualmente conosciute e tra loro verranno comparate.

A seguire verranno descritti i più conosciuti metodi di stima delle prestazioni per algoritmi di demosaicking sia a partire da un immagine di riferimento sia senza di questa (*No-reference*).

Infine verranno trattate alcune tecniche che consentono di passare in modo rapido ma efficiente da immagini ottenute da CFA di Bayer ad immagini ricostruite e campionate aventi un numero di pixel inferiore (*Downscaling*).

Capitolo 1

Introduzione

L'era della fotografia su pellicola ha subito un inesorabile declino, anche i più grossi produttori di pellicola al mondo sembra abbiano terminato quasi totalmente la produzione: nel 2008 Polaroid, nel 2009 Kodak, tutto questo dovuto al largo successo del digitale.

Il digitale ha subito una rapida ascesa nel corso degli ultimi anni, il suo utilizzo è divenuto vastissimo, fotocamere, telecamere, cellulari, apparecchi medici, e molto altro ancora; i fattori che hanno portato ad una così grande popolarità sono in primo luogo il costo contenuto, la riduzione delle dimensioni fisiche dei dispositivi, la velocità, il fattore ambientale e molti altri.

Nelle fotocamere digitali, il processo di acquisizione delle immagini avviene grazie a sensori in grado di trasformare uno stimolo luminoso in uno elettrico, i più comuni oggigiorno sono i CCD e i CMOS, i primi hanno come vantaggio un'immagine ad alta qualità con basso livello di rumore ed una semplicità costruttiva maggiore, i secondi invece consumano meno energia elettrica, hanno un costo più contenuto e sono più veloci.

Questi sensori sono organizzati in matrici, in modo tale che ciascun sensore occupi una porzione precisa nello spazio, ovvero un pixel. Il problema dell'acquisizione è che un sensore è sensibile ad un ampio range di lunghezze d'onda, quindi per poter isolare ciascuna delle componenti cromatiche, ad ogni sensore viene anteposto un filtro che lasci passare la sola lunghezza d'onda desiderata.

Per poter visualizzare un'immagine a colori, per ciascun pixel devono essere acquisite le tre componenti: verde, rosso e blu; da qui nasce un problema fisico di come porre tre sensori sensibili alle tre diverse lunghezze d'onda nel medesimo posto.

Il primo passo effettuatosi era scomporre la luce nelle tre componenti volute attraverso dei prismi e proiettarla sui sensori, questa tecnica è stata presto scartata poiché in pratica venivano utilizzati il triplo dei sensori rispetto al numero di pixel rendendo più elevati i costi di produzione, l'architettura e l'occupazione di memoria.

Il metodo che si è invece affermato è stato quello che prevede anteposizione di un CFA (*color filter array*) alla matrice di sensori, il CFA è composto da una ripetizione periodica di differenti filtri colorati secondo uno schema predefinito. In questo modo si

utilizza solamente un sensore per ogni pixel dell'immagine ma ovviamente viene raccolta una sola componente cromatica per ogni pixel, le restanti due devono essere ricostruite con il processo noto come *demosaicking* o *demosaicing*.

In figura 1.0.1 si può osservare una tipica architettura per un sistema di acquisizioni di immagini con CFA, lo strato indicato con *(a)* è un sistema di lenti che focalizza e filtra la componente infrarossa per ciascun sensore, con *(b)* viene indicato il CFA, con *(c)* la matrice di sensori ingrandita poi riportata sul chip in *(d)*.

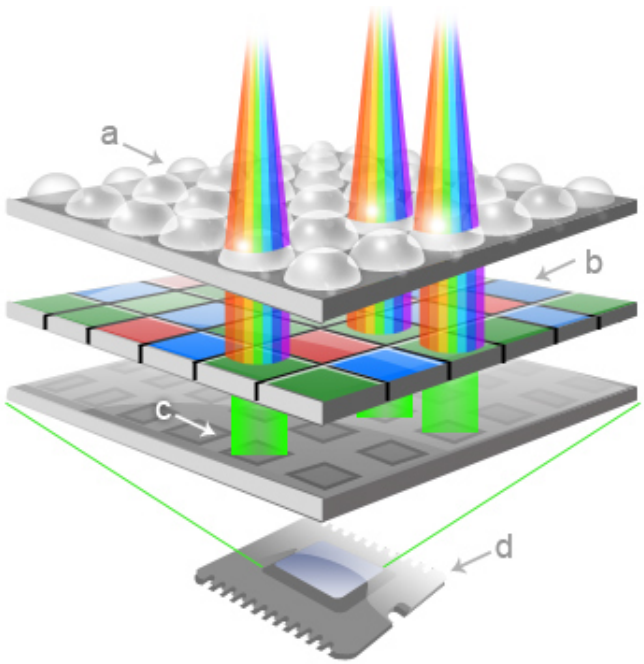


Figura 1.0.1: Sistema di acquisizione con CFA

Solo negli ultimi anni sono stati sviluppati i sensori Foveon (fig 1.0.2), questi sono in grado di raccogliere con un solo sensore le tre componenti di colore contemporaneamente grazie ad un sistema di sovrapposizione di tre strati di silicio, tuttavia i sensori Foveon rimangono ancora poco utilizzati a causa del loro elevato *cross-talk*, del costo di produzione ed della loro scarsa sensibilità (dovendo un fotone attraversare più strati).

Il processo di ricostruzione di un'immagine digitale si può riassumere con lo schema illustrato in figura 1.0.3: dopo che il fascio di luce è passato attraverso un sistema di lenti, attraversa la matrice di filtri colorati arrivando così al sensore dove viene focalizzato; i dati qui raccolti vengono comunemente definiti con *raw data*, ovvero l'immagine non ancora ricostruita.

Successivamente avvengono una serie di processi che non necessariamente rispecchiano l'ordine descritto in figura 1.0.3, nelle fotocamere professionali ad esempio l'operazione di memorizzazione avviene anche sui dati *raw*, permettendo al fotografo di gestire al meglio tutte le operazioni di ricostruzione e *post-processing* con un computer.

Blocchi fondamentali del processo di ricostruzione sono:

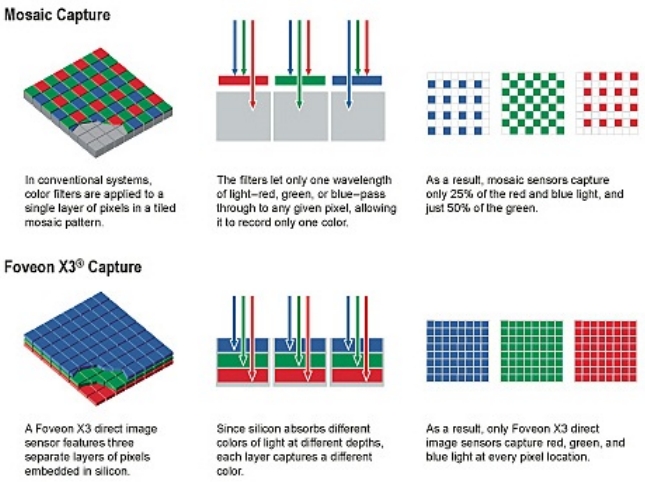


Figura 1.0.2: Confronto tra sistema con CFA e un sensore Foveon X3

- Demosaicking: ricostruzione dell'immagine a colori in seguito all'acquisizione con un CFA.

- Bilanciamento del bianco: a seconda della tipologia della sorgente luminosa, il colore bianco viene adattato in modo che l'occhio umano lo recepisca come tale.

- Correzione cromatica: vengono utilizzate delle curve di colore o correzioni gamma per correggere i valori di colore falsati dal dispositivo di acquisizione.

- Riduzione del rumore: ogni dispositivo al momento dell'acquisizione introduce un rumore sui dati, solitamente formato da una parte dipendente dal tipo di segnale e modellata con una variabile Poisson, e da una parte indipendente dal segnale modellata con una Gaussiana. Il *denoising* può essere effettuato sia prima che dopo il demosaicking, tuttavia scambiando l'ordine delle operazioni cambia il risultato finale e ricostruire dati in cui è presente rumore è sicuramente più difficile e modifica le caratteristiche del rumore.

- Compressione: nei dispositivi *consumer* l'immagine prima di essere memorizzata, viene spesso compressa con metodi *lossy* come JPEG, e successivamente formattata con *EXIF* che permette la memorizzazione di dati ulteriori che racchiudono informazioni su come, dove e quando è stata acquisita l'immagine.

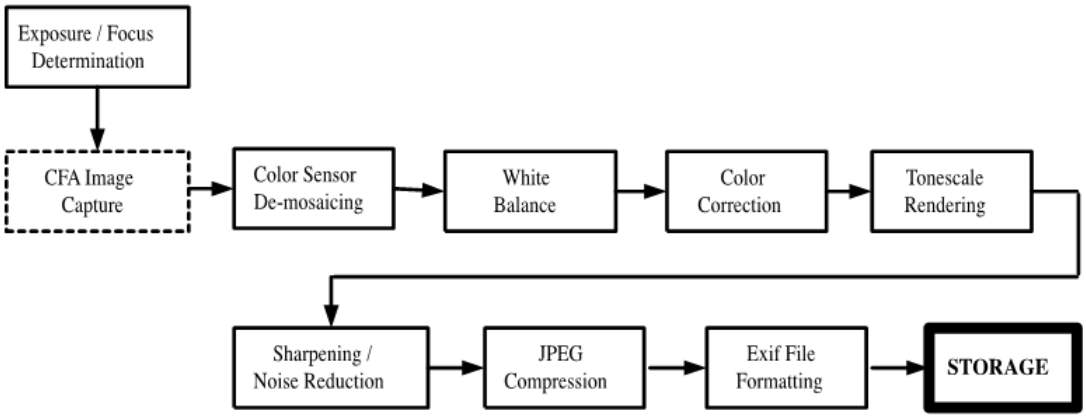


Figura 1.0.3: Processo di ricostruzione [30]

CFA

Nel corso degli anni, sono state realizzate molte tipologie diverse di *color filter array* (CFA) a volte chiamati anche *color filter mosaic* (CFM), il loro compito è di filtrare la luce incidente la matrice di sensori lasciando passare solamente una certa lunghezza d'onda per sensore.

In figura 1.0.4 (a) si può osservare il più famoso e diffuso CFA inventato da Bayer nel 1976 nel quale compaiono filtri colorati secondo le seguenti percentuali: 50% verdi, 25% rossi, 25% blu. La seguente scomposizione è stata pensata per riprodurre il comportamento dell'occhio umano, la percezione della luminanza avviene grazie all'azione combinata di diversi tipi di coni i quali sono più sensibili alle lunghezze d'onda del verde; nella figura 1.0.4 (c) abbiamo una variante di questo schema presentato in [28].

Il CFA in figura 1.0.4 (b) è derivante da quello di Bayer con uno dei due verdi sostituito da un verde smeraldo, questa tipologia viene utlizzata in alcuni dispositivi Sony.

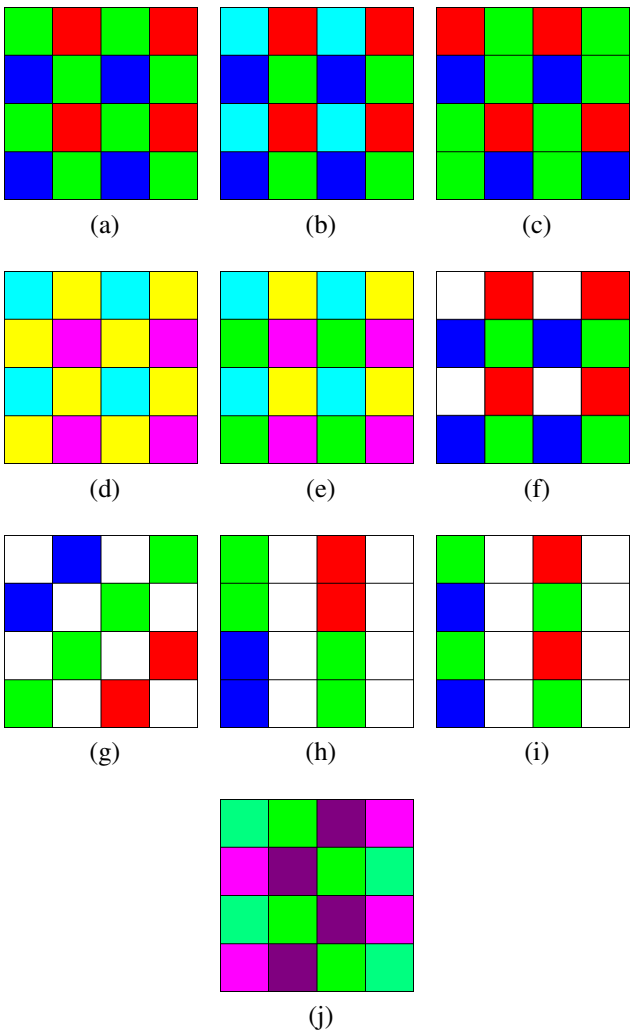


Figura 1.0.4: Tipologie di *color filter array*

La matrice CYGM (ciano, giallo, verde, magenta) in fugura 1.0.4 (e), così come CYYM (ciano, giallo, magenta) in figura 1.0.4 (d) sono dei CFA che utilizzano maggiormente colori secondari e permettono una maggiore rilevelazione della luce incidente

In figura 1.0.4 (f) si ha un CFA RGBW (rosso, verde, blu, bianco), in questa matrice di filtri viene incluso il bianco, ovvero un elemento trasparente che non filtra alcuna lunghezza d'onda incidente; questo tipo di pixel viene chiamato anche *pancromatico*, Kodak nel 2007 ha introdotto molti CFA di questo tipo, la loro peculiarità è che ignorando le celle pancromatiche i pixel rimanenti possono essere ricostruiti con gli stessi algoritmi utilizzati per CFA di Bayer. Altri tre esempi di schemi con il 50% di pixel pancromatici porposti da Kodak sono riportati nelle figure 1.0.4 (g), (h), (i).

Nell'articolo [29] viene proposta una nuova configurazione capace di ridurre la componente di aliasing grazie all'utilizzo di colori ottenuti come combinazioni lineari dei colori RGB, lo schema dei filtri è riportato in figura 1.0.4 (j).

Occhio umano

In questa sezione viene affrontata una panoramica basilare dell'ultimo ricevitore dell'intero sistema: l'occhio.

La visione del colore è un processo molto complesso che avviene nel nostro organismo attraverso gli occhi e il cervello. Si pensi all'occhio come ad una camera oscura di forma sferica, la luce ha accesso dalla pupilla che ha stessa funzione di un diaframma.

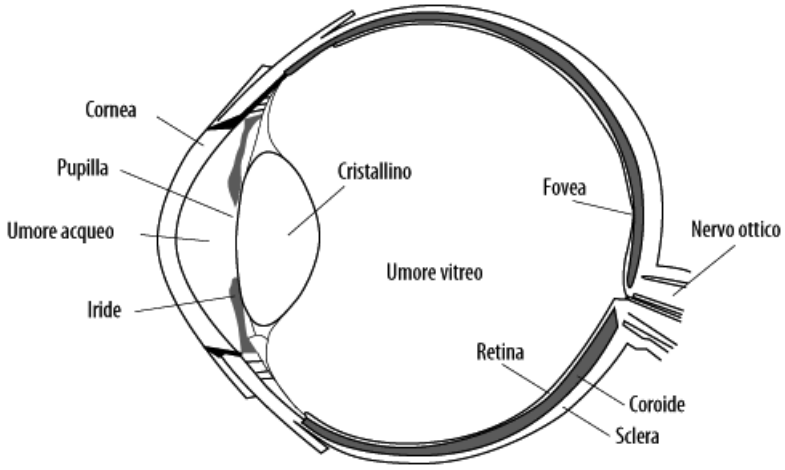


Figura 1.0.5: La struttura dell'occhio

Le pareti del bulbo oculare sono costituite di tre differenti strati [27, 26]:

- la sclera: è lo strato più esterno e il più robusto, essa dona all'occhio la sua forma sferica, è elastica e ricca di fibre, la sua funzione è quella di involucro. La sua parte anteriore, la cornea, è trasparente con curvatura di raggio solitamente attorno agli 8mm; nella sua parte posteriore ha un foro dove scorrono le fibre del nervo ottico.

- la coroide: è la parete intermedia delle pareti, ricca di vasi sanguigni e di colore scuro. Funge da parete assorbente di luce in maniera da impedire diffusioni e riflessioni interne. La sua parte anteriore è detta iride, che ha funzione di diaframma, ed il foro centrale che questa forma permettendo l'ingresso della luce è detto pupilla.

- la retina: è la parete più interna dell'occhio, è formata da una membrana sottilissima di terminazioni nervose le quali convogliano infine nel nervo ottico.

Sull'asse ottico dell'occhio, posto sul fondo della parete esiste una zona giallastra di 3 mm^2 circa che viene chiamata "macchia gialla", posta al centro di questa v'è la la fovea centrale, proprio qui avviene il puntamento e la messa a fuoco dell'immagine.

Passando ora alla parte interna dell'occhio, si nota subito una divisione in due parti, denominate umor acqueo e umor vitreo, tra loro separate dal cristallino, il quale ha una forma di lente biconvessa, questo ha il compito di focalizzare l'immagine sulla fovea centrale grazie ad opportune deformazioni.

L'iride ha lo stesso compito di un diaframma in una macchina fotografica, dilatandosi aumenta la quantità di luce che entra nell'occhio, questo avviene in particolare nella visione di immagini scure, con la conseguente dilatazione della pupilla; viceversa sotto lo stimolo di immagini molto luminose, l'iride si chiude il più possibile, e la pupilla si restringe limitando la quantità di luce in ingresso.

L'umor acqueo, situato nella parte anteriore dell'occhio, ha la funzione esercitare una pressione tale da mantenere costante la curvatura della cornea; l'umor vitreo, presente nella parte posteriore dell'occhio, mantiene costante la distanza tra cristallino e retina.

Quest'ultima, la retina, è la parte fotosensibile dell'occhio ed occupa una superficie molto vasta, all'incirca i $\frac{2}{3}$ della superficie totale interna, ed il suo spessore medio è di 300 μm circa, se osservata al microscopio, si possono vedere 10 strati, quello di maggiore interesse è costituito da coni e bastoncelli, i quali assieme compongono le terminazioni nervose fotosensibili del nervo ottico.

Bastoncelli

Il nome dei bastoncelli deriva dalla loro forma, la loro funzione è quella di trasformare la somma degli stimoli in un impulso elettrico ed inviarlo al cervello.

I bastoncelli sono collegati in gruppi alla struttura nervosa, visibile in figura 1.0.6, sono in grado di percepire la variazione di intensità e di adattarsi a seconda della tipologia di luce; il fatto che siano collegati in gruppo permette l'acquisizione di luce di basse intensità.

Le ricerche sui bastoncelli hanno dedotto che contengano un pigmento fotosensibile chiamato rodopsina, quando il bastoncello viene colpito da un fotone interagisce con la rodopsina la quale decolorandosi provoca uno stimolo nervoso. Se la quantità di luce è eccessiva la rodopsina si decolora determinando il fenomeno tipico di cecità temporanea, questo sintomo dura sino a che non viene ripristinata la giusta concentrazione di rodopsina tramite la circolazione sanguigna.

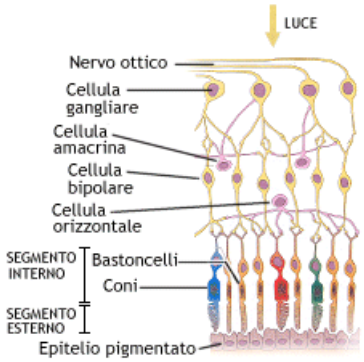


Figura 1.0.6: Struttura nervosa nella retina

L'informazione data da i bastoncelli è solamente quella di luminosità, senza cromaticità, a questa funzione ci pensano i coni.

Coni

I coni (figura 1.0.7) forniscono l'informazione di colore in merito alle differenze di blu-giallo e di verde-rosso, il loro comportamento non è ancora del tutto chiaro, tuttavia si può affermare che nei coni siano presenti pigmenti sensibili alla lunghezza d'onda, sui quali si forma la teoria del tristimolo appunto.

Esistono quindi tre tipi di coni: sensibili alle lunghezze d'onda corte (435nm: blu), alle medie (546nm: verde) e alle lunghe (671nm: rosso).

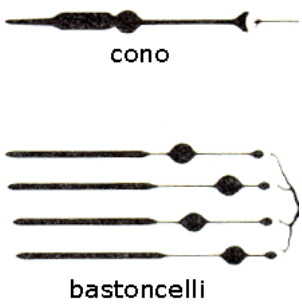


Figura 1.0.7: Coni e Bastoncelli

La sensibilità ai vari tipi di lunghezza d'onda e la combinazione dei tre tipi di coni fornisce al nostro cervello la percezione del colore.

I coni si concentrano nella zona centrale della retina, la fovea, invece i bastoncelli, i quali sono più sensibili al movimento e sono essenziali nella visione del buio, si concentrano nella zona periferica della retina.

Un'altra differenza tra i coni e bastoncelli, è nel loro modo di funzionamento, i primi agiscono individualmente ed ogni stimolo da loro rilevato si tramuta in un impulso che viene avviato al cervello, i secondi invece agiscono a gruppi di migliaia, e l'informazione da questi ricavata converge su di un singolo inter-neurone, da tutti questi impulsi ne viene ricavata una sorta di informazione media che viene poi inviata al cervello.

Capitolo 2

Tecniche di demosaicking

Il *demosaicking* è il processo che permette di ottenere un'immagine *full* RGB a partire da un'immagine acquisita con un CFA, in questo capitolo verranno descritte le principali tecniche di *demosaicking* suddivise in quattro aree di appartenenza: metodi euristici, metodi *edge-directed*, metodi nel dominio della frequenza, metodi basati sulla trasformata *wavelet* ed infine metodi di ricostruzione.

Interpolazione bilineare

Le prime tecniche di interpolazione per la ricostruzione di un'immagine completa a colori sono molto semplici, intuitive e veloci dal punto di vista computazionale. Una delle prime utilizzate è la semplice trasposizione o spostamento di un pixel vicino, si pensi al solo canale verde nel pattern Bayer, esistono metà pixel già identificati e metà da ricostruire, ciascun pixel verde viene ricopiato dal suo vicino a sinistra, un analogo procedimento viene poi applicato agli altri canali; ovviamente questa è una tecnica di interpolazione molto rude che porta moltissimi artefatti cromatici e ad una bassa resa visiva.

Una tecnica più interessante ed ancora molto diffusa seppur molto basilare è l'interpolazione bilineare, ciascun pixel viene stimato come una media aritmetica dei pixel vicini; si consideri il canale verde nel pattern Bayer come in figura 2.0.1, il pixel in posizione centrale viene calcolato in questo modo:

$$G(2,3) = \frac{1}{4}G(1,3) + \frac{1}{4}G(2,2) + \frac{1}{4}G(2,4) + \frac{1}{4}G(3,3)$$

Per gli altri canali ovviamente si possiedono meno pixel già assegnati, si ricorre quindi ad un procedimento analogo in due step, prima si calcolano i punti della cornice del blocchetto in figura 2.0.2, successivamente il punto centrale esattamente come nel canale verde:

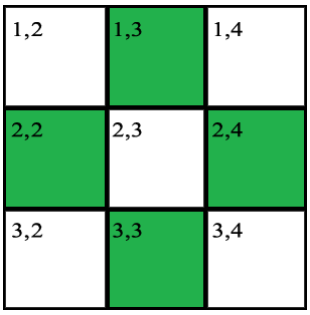


Figura 2.0.1: Green channel

$$
\begin{aligned}
R(2,2) &= \tfrac{1}{2}\left[R(1,2) + R(3,2)\right] \\
R(2,4) &= \tfrac{1}{2}\left[R(1,4) + R(3,4)\right] \\
R(1,3) &= \tfrac{1}{2}\left[R(1,2) + R(1,4)\right] \\
R(3,3) &= \tfrac{1}{2}\left[R(3,2) + R(3,4)\right] \\
R(2,3) &= \tfrac{1}{4}\left[R(1,3) + R(2,2) + R(2,4) + R(3,3)\right]
\end{aligned}
$$

Il procedimento è simile per il canale blu con gli opportuni indici, per quanto riguarda il bordo dell'immagine bisogna applicare un approccio leggermente diverso (come capita in ogni algoritmo di demosaicking) sempre attraverso delle opportune medie o traslazioni di pixel.

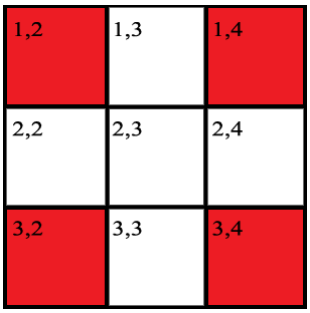


Figura 2.0.2: Red channel

L'interpolazione bilineare è un metodo di demosaicking ancora molto utilizzato tuttavia è risaputo che il suo funzionamento è accettabile solo nelle zone di colore uniforme ed introduce molti errori lungo i contorni e nelle zone ad alta frequenza. L'errore che affligge maggiormente questo metodo è lo Zipper Effect (descritto in sez. 3.2) il quale si presenta nei punti dove il colore subisce un cambio brusco, questo tipo di errore viene risaltato nelle componenti ad alta frequenza ad esempio nella palizzata in figura 2.0.3.

2.1 Metodi euristici

Gli approcci di ricostruzioni delle immagini euristici si fondano su assunzioni e considerazioni sui colori e sulle immagini piuttosto che su strutture matematiche.

Figura 2.0.3: Esempio interpolazione bilineare

Precedentemente si era detto che l'interpolazione bilineare fallisce laddove si presenta un cambio repentino di colore nell'immagine, questa problematica è stata affrontata da Cok [1].

Il metodo di Cok cerca di mantenere costante il colore tra pixel vicini, l'immagine ricostruita del canale verde viene ottenuta tramite interpolazione bilineare, invece per quanto concerne i canali rosso e blu, viene effettuata una media dei valori di tonalità.

L'algoritmo si basa sul concetto di tonalità come un rapporto tra i valori del colore in questione sul valore del verde ricostruito, ad esempio si consideri la figura 2.1.1, volendo ricostruire il valore B(2,2), dopo che si era già ricostruito totalmente il canale verde, bisogna calcolare:

$$B(2,2) = \frac{1}{2}G(2,2)\left[\frac{B(1,2)}{G(1,2)} + \frac{B(3,2)}{G(3,2)}\right]$$

$$B(3,3) = \frac{1}{2}G(3,3)\left[\frac{B(3,2)}{G(3,2)} + \frac{B(3,4)}{G(3,4)}\right]$$

$$B(2,3) = \frac{1}{4}G(2,3)\left[\frac{B(3,2)}{G(3,2)} + \frac{B(3,4)}{G(3,4)} + \frac{B(1,2)}{G(1,2)} + \frac{B(1,4)}{G(1,4)}\right]$$

L'approccio al problema della ricostruzione viene invece affrontato da Freeman [2] con l'introduzione dell'operatore mediano: dopo aver applicato l'interpolazione bilineare ai tre canali, per ogni singolo pixel non presente nativamente nel Bayer pattern, viene applicata una media delle differenze dei valori circostanti, ad esempio si consideri la figura 2.0.2:

$$R(2,3) = G(2,3) + \frac{1}{8} \sum_{\substack{i,j = 1,2,3 \\ (i,j) \neq (2,3)}} (R(i,j) - G(i,j))$$

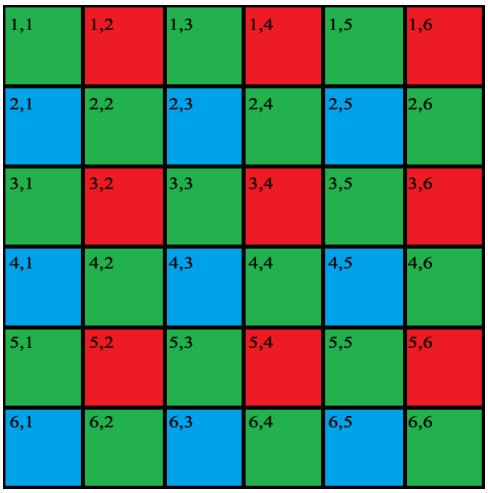


Figura 2.1.1: Bayer pattern

lo stesso si applica al canale blu con ovvie sostituzioni.

Tuttavia entrambi i metodi, sia quello di Cok che di Freeman, anche se portano a migliorie rispetto all'interpolazione bilineare e restando ad un livello computazionale basso, presentano forti limitazioni, soprattutto lungo i contorni delle figure e nelle componenti ad alta frequenza dell'immagine. Si può comunque affermare che tra i due metodi Freeman ha un comportamento migliore nelle zone di transizione di colore.

Figura 2.1.2: Esempio metodo di Cok

Interpolazione lineare ad alta qualità

In ambito lavorativo, talvolta è richiesto un tipo di demosaicking che abbia un costo computazionale basso: magari per poter realizzare anteprime al "volo" degli scatti o fotogrammi appena realizzati, si cerca quindi di realizzare un algoritmo che sia si efficiente come resa visiva ma soprattutto che abbia una elaborazione con tempi bassi.

La tecnica che viene descritta di seguito, tratta dall'articolo [11] è sicuramente un algoritmo facente parte della sfera euristica, questo metodo propone un semplice filtraggio lineare capace di realizzare un miglioramento di circa 5.5dB rispetto all'interpolazione bilineare, senza introdurre artefatti e tempi computazionali elevati tipici in demosaicking con filtraggi non lineari.

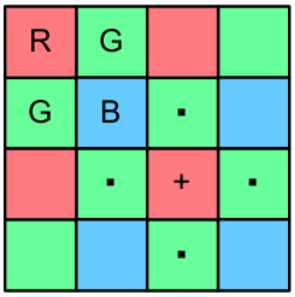


Figura 2.1.3: La tipica immagine Bayerizzata tratta da un CCD singolo

Le ipotesi di partenza prevedono che dovendo partire da un immagine sottocampionata con un CFA di Bayer classico, il canale verde porti la maggior parte dell'informazione di luminanza per un osservatore, avendo il doppio dei campioni rispetto al canale rosso e blu. Inizialmente l'immagine di Bayer viene interpolata con l'algoritmo bilineare, essendo quello più semplice, che tuttavia esegue le operazioni sui singoli canali indipendentemente; il fatto di non considerare la correlazione esistente tra i canali ovviamente genera numerosi artefatti lungo i contorni e nelle zone ad alta frequenza.

Prendiamo in considerazione la figura 2.1.3, nell'interpolazione bilineare il valore G(i,j) da ricostruire che cade nella posizione di un pixel rosso (posizione segnata con "+" in figura) o blu, è calcolato come la media dei pixel vicini (posizione segnata con "·" in figura):

$$\hat{G}(i,j) = \frac{1}{4} \sum_{(m,n)=\{(0,-1),(0,1),(-1,0),(1,0)\}} G(i+m, j+n)$$

Per i canali rosso e blu vengono applicati gli stessi calcoli, la complessità computazionale dell'algoritmo è O(1) e viene garantita l'assenza di *overflow* poiché il range dei valori in uscita è lo stesso dei valori in ingresso. L'idea principale di questo algoritmo è appunto integrare la correlazione esistente tra i canali al concetto di interpolazione bilineare.

Molti metodi di cui è stato discusso nelle altre sezioni, ricorrono all'utilizzo di filtraggi non lineari nelle regioni accanto ai pixel adiacenti a quello in esame; viene quindi da pensare a cosa succeda dal punto di vista qualitativo se si estenda la regione di pixel, ad esempio 5x5, nel calcolo di un interpolatore del tipo lineare; questo tipo di approccio viene introdotto in [22], stimando in *primis* R e G con interpolazione bilineare e successivamente stimando le differenze R-G e B-G, i calcoli vengono effettuati mediando su 4

punti e assumendo l'invarianza locale, se si riportano queste elaborazioni al caso lineare si trova che è equivalente a un filtro su una finestra quadrata di 5 pixel centrata sul pixel da ricostruire.

Il metodo proposto in [11], invece di utilizzare un approccio a tonalità costante, utilizza il seguente criterio che nei contorni si trova una componente di luminanza più forte della crominanza. Questo comporta ad esempio che quando si interpola un valore di verde in una posizione di un pixel rosso, viene memorizzata l'informazione del valore rosso in quella posizione; tale valore di rosso viene comparato con il valore stimato dall'interpolazione bilineare per i suoi pixel rossi vicini, se il valore è diverso dalla stima, è probabile che sia presente un brusco cambio di luminanza in quel pixel, allora è necessario correggere il valore verde interpolato aggiungendo una porzione di questo cambio di luminanza.

Ad esempio, in figura 2.1.3 per interpolare una valore verde nella posizione di un rosso (posizione segnata con "+" in figura), si usa la seguente formula:

$$\hat{G}(i,j) = \hat{G_{Bil}}(i,j) + \alpha \triangle_R(i,j)$$

dove $\hat{G_{Bil}}(i,j)$ viene calcolato con l'interpolazione bilineare e $\triangle_R(i,j)$ è il gradiente di R in quella posizione, calcolato in questo modo:

$$\triangle_R(i,j) \triangleq R(i,j) - \frac{1}{4} \sum_{(m,n)=\{(0,-2),(0,2),(-2,0),(2,0)\}} R(i+m, j+n)$$

in questo modo si riesce a correggere l'interpolazione bilineare aggiungendo la misura del gradiente $\triangle_R(i,j)$, il fattore di guadagno chiamato α gestisce il livello di correzione. Questo metodo si può definire appunto un approccio bilineare a correzione di gradiente, con fattori di guadagno capaci di controllare la correzione.

Per interpolare un pixel verde che giace sulla posizione di un pixel blu, si utilizza una formula simile:

$$\hat{G}(i,j) = \hat{G}_{Bil}(i,j) + \alpha \triangle_B(i,j)$$

dove in questo caso $\triangle_B(i,j)$ è così definito:

$$\triangle_B(i,j) \triangleq B(i,j) - \frac{1}{4} \sum_{(m,n)=\{(0,-2),(0,2),(-2,0),(2,0)\}} B(i+m, j+n)$$

Per quanto riguarda i pixel rossi, si differenziano tre maschere correttive diverse a seconda della posizione di tale pixel, in particolare per un rosso che cade nell'immagine bayerizzata nella riga dei rossi e nella colonna dei blu, si ha:

$$\hat{R}(i,j) = \hat{R}_{Bil}(i,j) + \beta \Delta_G(i,j)$$

dove $\hat{R}_{Bil}(i,j)$ viene ricopiato dall'interpolazione bilineare e $\Delta_G(i,j)$ è il termine di correzione dato da:

$$\Delta_G(i,j) = G(i,j) - \frac{1}{5} \sum_{(m,n)=\{(i,j\pm1),(i\pm1,j),(i,j\pm2)\}} G(m,n) + \frac{1}{10} \sum_{(m,n)=\{(i\pm2,j)\}} G(m,n)$$

In questo caso $\Delta_G(i,j)$ è un gradiente calcolato su di una zona a 9 punti come si vede in figura 2.1.4 e β è un fattore di guadagno.

In modo analogo ma con le opportune differenze, vengono calcolati i pixel rossi mancanti nelle posizioni righe blu e colonne rosse, per quanto riguarda i pixel rossi che nell'immagine di Bayer sono situati nelle righe blu e colonne blu, si applica:

$$\hat{R}(i,j) = \hat{R}_{Bil}(i,j) + \gamma\Delta_B(i,j)$$

dove γ è ancora un fattore di guadagno, e $\Delta_B(i,j)$ è come prima un gradiente su una regione a 5 punti.

Si procede allo stesso modo nella ricostruzione del canale blu, sempre osservando le maschere per il filtraggio in figura 2.1.4.

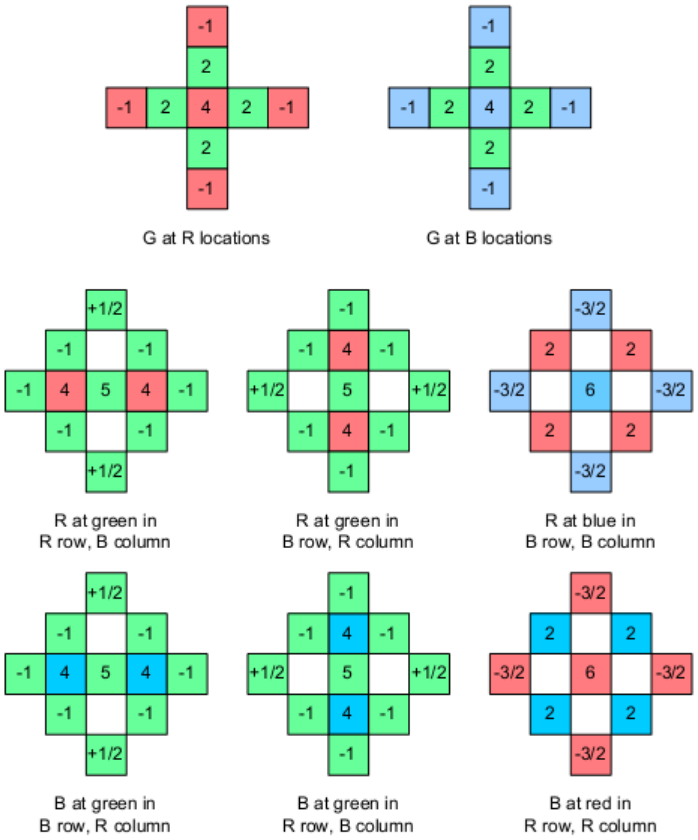


Figura 2.1.4: Coefficienti proposti per il filtro lineare

I fattori di guadagno sono stati determinati usando un approccio di tipo Wiener, calcolando i coefficienti che minimizzassero l'errore quadratico medio dell'interpolazione da una statistica del secondo ordine basata sul database Kodak [23]. L'unico modo per progettare un filtro lineare con un minor errore quadratico medio è aumentare la regione di pixel in considerazione, ovviamente ciò comporta un aumento della complessità di calcolo.

Nella mia implementazione di questo algoritmo ho utilizzato i fattori di guadagno suggeriti dagli autori:

$$\alpha = \frac{1}{2} \quad \beta = \frac{5}{8} \quad \gamma = \frac{3}{4}$$

I risultati in tabella 2.1 ottenuti con questo metodo rispecchiano tutte le considerazioni precedenti, di seguito si riporta una tabella dei valori di PSNR valutati sul database delle immagini Kodak:

Tabella 2.1: PSNR a confronto [dB]

Immagine\PSNR	Bilineare	[24]	[11]
1	26,19	38,21	31,71
2	33,11	38,78	37,66
3	34,37	42,10	39,48
4	33,61	40,45	38,54
5	26,69	38,04	33,55
6	27,69	39,73	32,85
7	33,46	42,22	39,39
8	23,61	36,07	28,83
9	32,46	42,51	38,13
10	32,34	42,54	38,26
11	29,26	39,77	34,77
12	32,91	43,01	37,54
13	23,92	34,92	29,58
14	29,27	36,31	34,58
15	31,54	39,34	36,13
16	31,38	43,27	36,36
17	31,95	41,80	37,49
18	27,88	37,33	33,50
19	28,00	40,25	33,46
20	31,36	40,90	36,23
21	28,57	39,32	34,11
22	30,52	38,40	35,59
23	35,26	42,06	41,35
24	26,68	35,45	32,04
Media	30,08	39,70	35,46

Ho evidenziato un confronto tra il metodo bilineare e uno dei migliori algoritmi di demosaicking in mio possesso [24], mediamente l'incremento di PSNR dal metodo bilineare è di +5,38 dB, invece differisce dal metodo adattativo [24] di -4,24 dB. Per un

Figura 2.1.5: Immagine ricostruita

confronto completo tra tutti i metodi da me analizzati si legga il capito *No Reference quality evaluation* (sez. 3.4).

Si possono ancora notare artefatti lungo le zone ad alta frequenza, tuttavia il risultato è buono e anche i tempi di elaborazione[1], come riportato in tabella 2.2:

[1] I risultati sono stati eseguiti su una media di 20 prove sulla stessa immagine su un PC con Intel Centrino Duo T5450 @ 1.67Ghz.

Tabella 2.2: Tempi di eleaborazione

	Tempo [s]
Bilineare	0.3022
[24]	1.0954
[11]	0.3344

2.2 Metodi edge-directed

Per risolvere le problematiche relative ai forti cambiamenti di colore derivanti dalla presenza dei contorni delle figure all'interno della nostra immagine sono stati pensati dei metodi di ricostruzione che identificano la direzione di un contorno ed eseguono di conseguenza delle opportune operazioni nella demosaicizzazione.

La problematica dei contorni nella ricostruzione nasce dal fatto che nei metodi precedentemente analizzati si effettuano medie tra pixel che appartengono a zone di colore molto differenti tra loro, senza rispecchiare le figure presenti nelle immagini originarie; se invece si riesce a determinare l'andamento di un contorno si possono interpolare i colori dei pixel mancanti verso una direzione preferenziale, evitando talvolta artefatti quali lo *zipper effect*.

I primi algoritmi di demosaicking *egde-directed* vennero sviluppati utilizzando i gradienti: si calcolano prima il gradiente orizzontale e successivamente quello verticale; viene interpolato solamente il canale verde nella direzione in cui il gradiente risulta minore, ossia dove la variazione di colore è minore, quindi si conserva localmente l'informazione; il canale rosso viene poi stimato dal canale verde e così pure per il canale blu. Il gradiente del canale verde (il canale più ricco di informazione) può venire calcolato sia attraverso l'esame dei valori di blu e rosso noti, sia attraverso una differenza dei valori di verde [3].

Nel metodo proposto da Hamilton ed Adams [4] [5], si esegue una analisi iniziale su quale interpolazione offra le migliori prestazioni con un basso costo computazionale. Prendendo un pattern Bayer classico si può osservare che il verde è complessivamente campionato di un fattore due lungo le righe e le colonne, considerando la sua trasformata zeta:

$$G_s(z) = \frac{1}{2}G(z) + \frac{1}{2}G(-z)$$

Ne consegue che il filtraggio ideale che non introduce aliasing sia un filtro con risposta in frequenza pari ad un *rect*, quindi applicando un filtro:

$$h_0 = [a_1 \;\; 1 \;\; a_1]$$

si ottiene una risposta in frequenza che non è un filtro passa basso ideale, questo introduce un fattore di aliasing:

$$\hat{G}(z) = \frac{1}{2}G_s(z)H_0(z) + \frac{1}{2}G_s(-z)H_0(z)$$

Viene quindi applicato un filtraggio $h_1$sulla componente rossa e sommata in questo modo:

$$\hat{G}(z) = \frac{1}{2}G_s(z)H_0(z) + \frac{1}{2}G_s(-z)H_0(z) + \frac{1}{2}R_s(z)H_1(z) + \frac{1}{2}R_s(-z)H_1(z)$$

Il filtro h_1 viene progettato in modo che elida le componenti di aliasing del filtraggio con h_0 e che rispetti lo stesso comportamento alle basse frequenze, Hamilton e Adams hanno proposto un filtro di quinto ordine:

$$h = [(0 \;\; a_1 \;\; 1 \;\; a_1 \;\; 0) + (a_2 \;\; 0 \;\; a_0 \;\; 0 \;\; a_2)]$$

la cui risposta in frequenza è :

$$H(w) = 1 + a_0 + 2a_1\cos(w) + 2a_2\cos(2w)$$

I coefficienti proposti sono:

$$\begin{cases} a_0 = \frac{1}{3} \\ a_1 = \frac{1}{2} \\ a_2 = -\frac{1}{6} \end{cases} \qquad \text{oppure} \qquad \begin{cases} a_0 = \frac{1}{2} \\ a_1 = \frac{1}{2} \\ a_2 = -\frac{1}{4} \end{cases}$$

Con tali coefficienti si approssimano abbastanza bene l'interpolatore ideale, commettendo comunque un errore di ricostruzione, l'algoritmo progettato prevede quindi una pre-analisi su quale direzione tra verticale ed orizzontale causi meno aliasing.

Solitamente le componenti di aliasing si manifestano con maggior intensità nelle zone ad alta frequenza dell'immagine, scegliere una direzione dove l'aliasing è minore significa quindi mantenere la direzione ove il colore varia più lentamente, ovvero di cercare di interpolare dove il colore è più omogeneo.

La valutazione del termine di aliasing viene effettuata grazie al gradiente Laplaciano per la luminanza e per la crominanza, prendiamo in esempio il calcolo del pixel G(3,4) che cade nella posizione di un pixel rosso, con riferimento alla figura 2.2.1:

$$\delta_H = |G(3,3) - G(3,5)| + |2R(3,4) - R(3,2) - R(3,6)|$$

$$\delta_V = |G(2,4) - G(4,4)| + |2R(3,4) - R(1,4) - R(5,4)|$$

Ne segue la scelta a seconda di quale gradiente risulti minore, ad esempio se è minore il gradiente orizzontale δ_H la direzione di interpolazione è orizzontale :

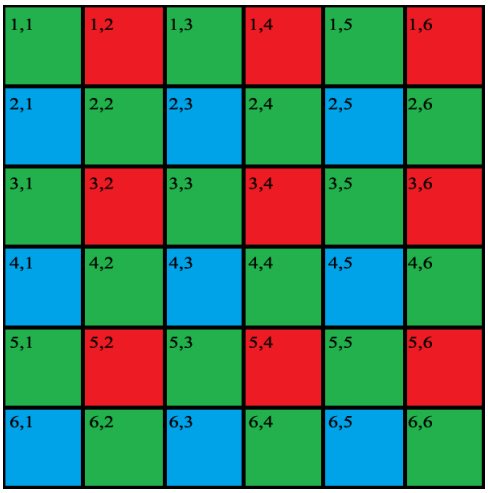


Figura 2.2.1: Bayer pattern

$$G(3,4) = \begin{cases} \frac{G(3,3)+G(3,5)}{2} + \frac{2R(3,4)-R(3,2)-R(3,6)}{4} & \delta_H < \delta_V \\ \frac{G(2,4)+G(4,4)}{2} + \frac{2R(3,4)-R(1,4)-R(5,4)}{4} & \delta_H > \delta_V \\ \frac{G(3,3)+G(3,5)+G(2,4)+G(4,4)}{4} + \frac{4R(3,4)-R(3,2)-R(3,6)-R(1,4)-R(5,4)}{8} & \delta_H = \delta_V \end{cases}$$

Nel caso di uguaglianza tra i gradienti viene effettuata un'interpolazione bilineare affiancata dall'informazione data dai pixel vicini rossi del pattern Bayer; trovata tutta la matrice del canale verde, i restanti due canali, rosso e blu, vengono ricavati come differenze verde-blu o verde-rosso. Questo algoritmo ovviamente prende in esame solamente le direzioni orizzontali e verticali tralasciando la direzione diagonale e anti-diagonale, oltre a questa limitazione non fornisce un buon funzionamento nelle zone ad alta frequenza per le motivazioni spiegate prima, tuttavia porta a migliorie rispetto agli algoritmi trattati in precedenza riuscendo a ridurre notevolmente l'effetto zipper.

Figura 2.2.2: Esempio di interpolazione Hamilton-Adams

Un ulteriore algoritmo basato sull'interpolazione direzionale viene descritto in [13], dopo aver ricostruito l'intero canale verde sia con direzioni orizzontali che verticali viene effettuata una decisione a posteriori sulla migliore direzione di interpolazione con il metodo del gradiente, successivamente vengono interpolati i canali rosso e blu nella stessa direzione adottata nel canale verde. Infine viene utilizzato un *refining step* per ridurre gli artefatti basato sull'intercorrelazione tra i canali.

Metodi *Homogeneity-Directed*

L'algoritmo proposto da Hirakawa e Parks in [6] è basato appunto sull'omogeneità, ovvero un pixel ricostruito viene interpolato a seconda dell'informazione data da i pixel vicini.

Inizialmente l'immagine viene ricostruita seguendo le direzioni orizzontale e verticale, viene supposto che la variazione verde-rosso sia lenta, e come in tutti i metodi di demosaicking la prima matrice ad essere ricostruita è la verde.

Questo algoritmo produce quindi due distinte immagini *full color* una orizzontale ed una verticale, tali immagini vengono quindi convertite nel formato LAB (vedi sez. 3.1.2).

Successivamente viene ricavata una mappa di omogeneità che ha il compito di misurare quanto siano coerenti i valori interpolati con i pixel adiacenti.

Sfruttando le informazioni date dalla mappa di intensità viene ricavata l'immagine complessiva con elementi appartenenti sia all'interpolazione orizzontale che a quella verticale.

Infine vengono effettuati step iterativi di riduzione degli artefatti con un filtro mediano delle differenze di colore.

Figura 2.2.3: Esempio di interpolazione Hirakawa-Parks

Un metodo simile a questa tecnica di demosaicking viene descritto da Zhang e Wu in [9], a differenza del precedente le due immagini ricostruite seguendo linee verticali e orizzontali vengono fuse assieme.

La ricostruzione avviene facendo una somma pesata di entrambe le immagini ricostruite, il peso che viene assegnato è determinato in base al criterio *linear minimum square error.*

Figura 2.2.4: Esempio di interpolazione Zhang-Wu

Ricostruzione con *Primary Consistent Soft Decision* (PCSD)

Il termine *Primary consistent* rappresenta la somiglianza tra i gradienti dei tre canali verde, rosso e blu che formano un immagine. Spesso negli algoritmi precedenti un pixel veniva interpolato utilizzando una direzione preferenziale diversa a seconda del canale, tuttavia esiste una forte correlazione all'interno di una immagine tra i canali. Nella ricostruzione dell'immagine full RGB, interpolare con diverse direzioni lo stesso pixel può causare forti aberrazioni ed artefatti soprattutto nelle frange ad alta frequenza.

Zhang e Wu hanno proposto questo metodo in [8], dopo aver interpolato tutte e tre le componenti di colore di uno stesso pixel nella medesima direzione (verticale od orizzontale), viene scelta la miglior strada di interpolazione con tecnica *soft decision*, dopo aver ricostruito per intero l'immagine nelle due direzioni, con un evidente spreco di memoria.

Un altro fattore importante da considerare nella ricostruzione dell'immagine è il contesto del singolo pixel, ovvero l'idea che i pixel adiacenti possano fornire molte informazioni utili nel demosaicking, mentre negli algoritmi euristici questo fattore non veniva considerato poiché il pixel veniva assunto indipendente dai vicini, in questo caso la decisione sulla direzione di interpolazione viene effettuata su di un'area di pixel.

2.3 Metodi nel dominio della frequenza

Esistono tutta una serie di tecniche di ricostruzione delle immagini che sfruttano la correlazione esistente tra i canali e in particolare in questa sezione vengono affrontati i metodi nel dominio della frequenza.

Seguendo l'approccio descritto in [21], partiamo dall'immagine originale $X(n)$ dove sono presenti per intero i tre canali RGB, quando questa viene catturata con un CFA, otteniamo una versione campionata dell'immagine data da $f_{CFA}\,[n1, n2]$ con $\{n_1, n_2\}$ appartenenti alla matrice rettangolare utilizzata Λ, che rappresenta l'output del sensore.

La matrice Λ è ripartita in tre sottoinsiemi disgiunti, Λ_R,Λ_G e Λ_B, le componenti che stiamo cercando di stimare completamente vengono chiamate f_R, f_G, f_B. Il sottocampionamento può essere rappresentato come una moltiplicazione delle funzioni sottocampionate $m_i[n_1, n_2]$ che hanno valore 1 se appartiene alla matrice Λ, e zero altrimenti.

$$
\begin{aligned}
m_G[n_1, n_2] &= \tfrac{1}{2}\left(1 + (-1)^{n_1+n_2}\right) \\
m_R[n_1, n_2] &= \tfrac{1}{4}\left(1 + (-1)^{n_1}\right)\left(1 + (-1)^{n_2}\right) \\
m_B[n_1, n_2] &= \tfrac{1}{4}\left(1 + (-1)^{n_1}\right)\left(1 - (-1)^{n_2}\right)
\end{aligned}
$$

Il segnale ottenuto dal CFA può venire espresso con:

$$
f_{CFA}[n_1, n_2] = \sum_{i \in \{R,G,B\}} f_i[n_1, n_2] m_i[n_1, n_2]
$$

Sviluppando i termini e passando alla luminanza e crominanza si ottiene:

$$
f_{CFA}[n_1, n_2] = f_L[n_1, n_2] + f_{C1}[n_1, n_2] \exp(j2\pi(n_1+n_2)/2) + f_{C2}[n_1, n_2] \exp(j2\pi(n_1-n_2)/2)
$$

La formula appena scritta può essere interpretata come una componente in banda base di luminanza f_L , una componente f_{C1} e una seconda componente f_{C2} entrambe contenenti informazioni di crominanza modulate in banda passante.

Ora utilizzando la trasformata di Fourier si ottiene:

$$
F_{CFA}(u, v) = F_L(u, v) + F_{C1}(u - 0.5, v - 0.5) + F_{C2}(u - 0.5, v) - F_{C2}(u, v - 0.5)
$$

dove con $\{u, v\}$ sono state espresse in cicli per pixel (c/px).

In notazione matriciale, le relazioni tra luminosità/cromaticità e tra i canali RGB sono date da:

$$\begin{bmatrix} f_L \\ f_{C1} \\ f_{C2} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ -\frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} f_R \\ f_G \\ f_B \end{bmatrix}$$

$$\begin{bmatrix} f_R \\ f_G \\ f_B \end{bmatrix} = \begin{bmatrix} 1 & -1 & -2 \\ 1 & 1 & 0 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} f_L \\ f_{C1} \\ f_{C2} \end{bmatrix}$$

Ogni segnale in cui i valori dei tre canali sono tra loro identici $f_R = f_G = f_B$ le componenti di crominanza sono zero (siamo in presenza di un valore in scala di grigi). In generale le componenti di crominanza avranno minore energia e larghezza di banda rispetto alla componente di luminanza.

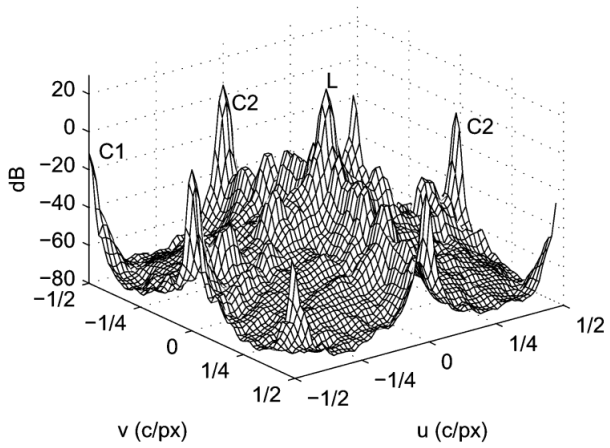


Figura 2.3.1: Densità spettrale di potenza dell'immagine Kodak del faro

Nella figura 2.3.1 , si osserva che esiste interferenza o *cross-talk* molto accentuata tra le componenti di luminosità f_L e di crominanza f_{C2} ; da questo spettro è possibile recuperare la componente di luminosità con un filtro passa basso e le componenti di crominanza con dei filtraggi appropriati in banda passante. Successivamente utilizzando le formule inverse introdotte in precedenza si possono trasformare i valori nel dominio RGB. Questo approccio non elimina la sovrapposizione delle componenti f_L e f_{C2} , tuttavia esistendo due copie separate della componente di crominanza $C2$, ognuna delle quali ha una diversa sovrapposizione con la luminanza è possibile ridurre in modo significativo il *cross talk*.

Nell'articolo [40] gli autori propongono un metodo capace di ridurre l'aliasing tra le componenti. Si consideri inizialmente il canale verde: in un *color filter array* di tipo Bayer questo canale presenta il doppio dei campioni rispetto agli altri, nel dominio della frequenza significa che tale canale presenta meno aliasing rispetto al rosso ed al blu ed inoltre che le componenti rosse e blu possono perdere le informazioni delle alte frequenze che sono presenti invece nella componente verde.

Si considera la trasformata zeta del segnale in ingresso:

$$R_S(z_1, z_2) = \frac{1}{4}R(z_1, z_2) - \frac{1}{4}R(-z_1, z_2) + \frac{1}{4}R(z_1, -z_2) - \frac{1}{4}R(-z_1, -z_2)$$

$$B_s(z_1, z_2) = \frac{1}{4}B(z_1, z_2) + \frac{1}{4}B(-z_1, z_2) - \frac{1}{4}B(z_1, -z_2) - \frac{1}{4}B(-z_1, -z_2)$$

$$G_s(z_1, z_2) = \frac{1}{2}G_s(z_1, z_2) + \frac{1}{2}G_s(-z_1, -z_2)$$

dove i termini del tipo $X(z_1, z_2)$ rappresentano i termini utili in bassa frequenza, ed i termini del tipo $X(-z_1, z_2), X(z_1, -z_2), X(-z_1, -z_2)$ rappresentano i termini di *aliasing* , che possono distorcere l'immagine in ingresso dove questa contiene componenti in alta frequenza.

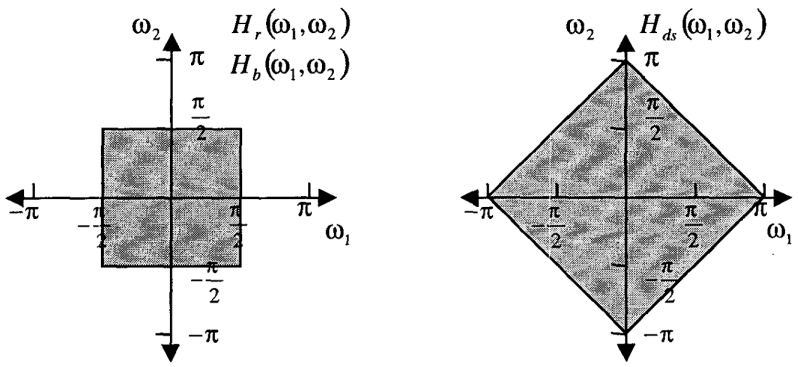


Figura 2.3.2: Filtraggi ideali per l'interpolazione del CFA

In figura 2.3.2 si nota che un CFA introduce una interpolazione dove per le componenti rosse e blu viene utilizzato un filtro di tipo passa-basso rettangolare, mentre per la componente verde viene utilizzato un filtro passa-basso a forma di diamante.

Considerando la forte correlazione esistente tra i canali e che il verde possiede più informazioni rispetto agli altri due, un metodo per ricostruire i canali rosso e blu è utilizzando l'informazione in bassa frequenza dei canali rosso e blu e l'informazione in alta frequenza del canale verde, con questa idea sono stati creati i filtri per le alte frequenze verticale ed orizzontale H_v, H_h in figura 2.3.3 , l'uscita di questi filtri viene quindi sommata all'uscita dei filtraggi passa-basso aumentando in questo modo la nitidezza dei canali rosso e blu. Per rimuovere le componenti di *aliasing* dal canale rosso e blu, è necessario sottrarre ed aggiungere termini del canale verde in questo modo:

$$\hat{R}(z_1, z_2) = R(z_1, z_2)H_r(z_1, z_2) + G_h(z_1, z_2) + G_h(-z_1, z_2) + G_v(z_1, z_2) - G_v(z_1, -z_2)$$

$$\hat{B}(z_1, z_2) = B(z_1, z_2)H_b(z_1, z_2) + G_h(z_1, z_2) - G_h(-z_1, z_2) + G_v(z_1, z_2) + G_v(z_1, -z_2)$$

dove:

$$G_h(z_1, z_2) = G_s(z_1, z_2)H_h(z_1, z_2)$$

$$G_v(z_1, z_2) = G_s(z_1, z_2)H_v(z_1, z_2)$$

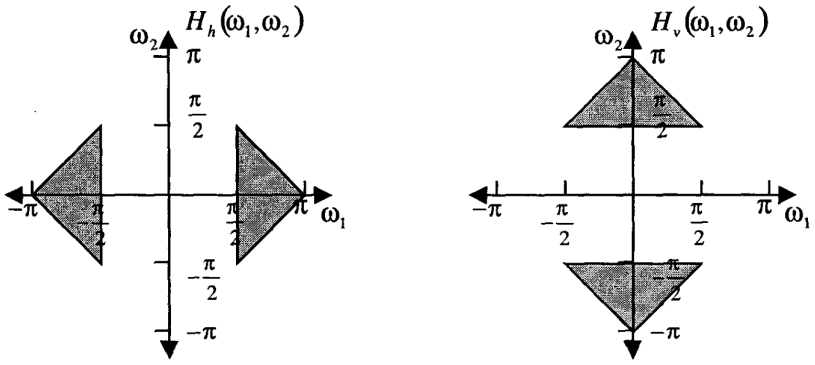


Figura 2.3.3: Filtraggi ideali passa-alto per il canale verde

Nell'articolo [42] viene riportata un'analisi ancora nel dominio della frequenza, i principi alla base di questo metodo sono due: le alte frequenze sono simili tra le tre componenti di colore, e le alte frequenze lungo gli assi orizzontali e verticali sono essenziali per una ricostruzione di qualità; nelle posizioni del rosso e del blu le diverse componenti in frequenza vengono modulate come in fig 2.3.4, mentre nei pixel verdi, che sono disposti a quinconce, i termini di differenza di colore modulati alle frequenze $(0, \pm\pi)$ e $(\pm\pi, 0)$ svaniscono.

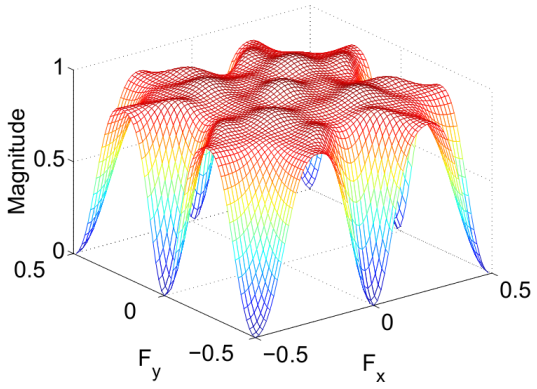


Figura 2.3.4: Trasformata di Fourier del filtro passa-basso proposto in [41]

Nelle posizioni appena descritte ci sono quindi meno sovrapposizioni spettrali tra crominanza e luminanza e quindi la stima della luminanza per i pixel verdi viene calcolata

con il filtro passa-basso proposto in fig 2.3.5 , mentre per i pixel rossi e blu viene utilizzato un approccio del tipo adattativo. Successivamente i colori vengono ricostruiti con una interpolazione bilineare di $R - \hat{L}, G - \hat{L}, B - \hat{L}$ utilizzando i valori noti dal CFA. Poiché esiste una forte correlazione tra le componenti di colore nelle regioni ad alta frequenza, anche le componenti ad alta frequenza della luminosità del verde sono fortemente correlate a quelle dei canali rosso e blu.

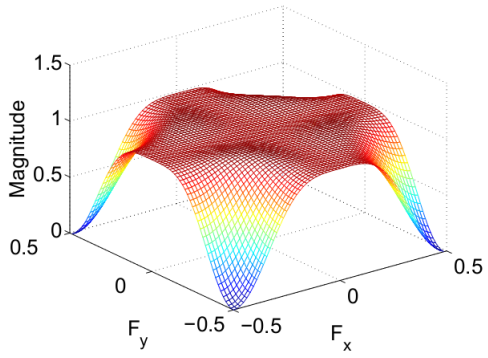


Figura 2.3.5: Spettro del filtro passa-basso per la ricostruzione della luminanza del verde

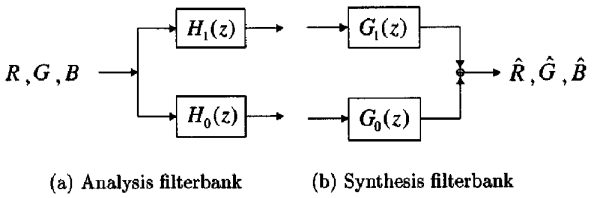


Figura 2.3.6: Banco di filtri utilizzato in [10]

L'algoritmo prevede che inizialmente venga interpolato il canale verde, con un metodo semplice di demosaicking quale il bilineare o edge-directed, poi utilizzando i campioni del blu nel pattern di Bayer si ottiene una versione sottocampionata del canale blu, quindi i valori del canale verde interpolati corrispondenti alle posizioni di blu nel pattern di Bayer vengono utilizzati per formare un'immagine sottocampionata del canale verde. Ora vengono decomposti i due canali sotto-campionati (verde e blu) nelle loro quattro sotto-bande, ovvero applicando il filtro passa-basso $h_0 = \frac{1}{4}[1, 2, 1]$ e passa-alto $h_1 = \frac{1}{4}[1, -2, 1]$ alle righe e alle colonne per ciascun canale si ottengono le quattro sotto-bande LL, LH, HL, HH. Le sotto-bande ad alte frequenze (HL, LH, HH) del canale verde vengono sostituite con le sottobande corrispondenti del canale blu, quindi viene ricostruito il canale verde e inseriti i pixel nelle loro posizioni corrispondenti nella stima iniziale del canale verde. Questa procedura viene ripetuta con le opportune sostituzioni anche per la ricostruzione dei pixel verde che sono situati nei pixel rossi nel pattern.

2.4 Metodi basati sulla trasformata Wavelet

Nell'articolo [10] gli autori utilizzano un diverso approccio per rimuovere la componente di aliasing sulle alte frequenze basato sulla intercorrelazione tra i canali.

Viene ribadito il fatto che la struttura di un CFA di Bayer porta inevitabilmente ad una maggior accuratezza del campionamento del verde rispetto al rosso ed al blu il che si traduce in un maggior aliasing sulle componenti ad alta frequenza di questi due canali, per risolvere il problema viene sfruttata l'alta intercorrelazione esistente tra i canali, cioè vengono estrapolate le informazioni nelle alte frequenze del verde per ricostruire le alte frequenze degli altri due canali.

Vengono introdotti due insiemi con vincoli, uno proveniente dai dati osservati dal CFA:

$$C_0 = \{S(n_1, n_2) : S(n_1, n_2) = O(n_1, n_2) \quad \forall (n_1, n_2) \in \Lambda_s, S = R, G, B\}$$

e l'altro basato sulla intercorrelazione tra i canali C_d, ottenuto dalla scomposizione in sottobande dei canali con il sistema di filtri in figura 2.3.6 (in pratica una trasformazione Wavelet) con H_0 filtro passa-basso e H_1filtro passa-alto. La ricostruzione perfetta viene ottenuta quando: $H_0(z)G_0(z) + H_1(z)G_1(z) = 1$

Considerando le risposte impulsive dei filtri si possono scrivere le quattro sottobande di un segnale bidimensionale $S(n_1, n_2)$:

$$\begin{aligned}
(W_1 S)(n_1, n_2) &= h_0(n_1) * [h_0(n_2) * S(n_1, n_2)] \\
(W_2 S)(n_1, n_2) &= h_1(n_1) * [h_0(n_2) * S(n_1, n_2)] \\
(W_3 S)(n_1, n_2) &= h_0(n_1) * [h_1(n_2) * S(n_1, n_2)] \\
(W_4 S)(n_1, n_2) &= h_1(n_1) * [h_1(n_2) * S(n_1, n_2)]
\end{aligned}$$

Il secondo insieme di vincoli forza i dettagli delle immagini rossa e blu ad essere simili ai dettagli dell'immagine verde:

$$C_d = \left\{ \begin{array}{c} S(n_1, n_2) : |(W_k S)(n_1, n_2) - (W_k G)(n_1, n_2)| \leq T(n_1, n_2) \\ \forall (n_1, n_2); k = 2, 3, 4; S = R, B \end{array} \right\}$$

dove con T viene indicata la soglia positiva che quantifica la somiglianza massima delle sottobande.

Nell'articolo [15] viene presentato un metodo iterativo basato sulla *color difference rule*. Tale regola afferma che considerando la correlazione tra i canali, il segnale differenza di colore è approssimativamente piatto (costante) lungo i contorni di un oggetto, il che si traduce in:

$$D_{a,b}(x) = I_a(x) - I_b(x) = constant$$

L'autore dell'articolo ha voluto sviluppare un algoritmo con basso livello computa-

zionale che non introduce operazioni non lineari o filtraggi, si parte come di consueto dall'analisi del canale verde, poiché dominante in un pattern di tipo Bayer, il quale viene utilizzato come riferimento per le crominanze seguenti ((i,j) rappresentano le coordinate spaziali):

$$D_R(i,j) = R(i,j) - G(i,j) \quad , \quad D_B(i,j) = B(i,j) - G(i,j)$$

che vengono utilizzate nella ricostruzione dei pixel mancanti rossi e blu; il pixel verde che copre la posizione rossa nel pattern viene aggiornato in questo modo:

$$\hat{D}_R^n(2i-1,2j) = \frac{1}{4}\left[D_R^n(2i-2,2j) + D_R^n(2i,2j) + D_R^n(2i-1,2j-1) + D_R^n(2i-1,2j+1)\right]$$

$$G^{n+1}(2i-1,2j) = R^n(2i-1,2j) - \hat{D}_R^n(2i-1,2j)$$

Per la posizione blu si procede in modo analogo; i pixel rossi vengono aggiornati in due passi: si interpolano i pixel rossi con $i+j$=pari :

$$\hat{D}_R^n(2i,2j) = \frac{1}{2}\left[D_R^n(2i-2,2j) - D_R^n(2i+1,2j)\right]$$

$$\hat{D}_R^n(2i+1,2j+1) = \frac{1}{2}\left[D_R^n(2i+1,2j) + D_R^n(2i+1,2j+2)\right]$$

$$R^{n+1}(2i,2j) = G^n(2i,2j) + \hat{D_R^n}(2i,2j)$$

$$R^{n+1}(2i+1,2j+1) = G^n(2i+1,2j+1) + \hat{D_R^n}(2i+1,2j+1)$$

e successivamente si interpolano i pixel rossi con i=pari, j=dispari (con la stessa tecnica vengono calcolati i pixel blu):

$$\hat{D}_R^n(2i,2j-1) = \frac{1}{4}\left[D_R^n(2i-1,2j-1) + D_R^n(2i+1,2j-1) + D_R^n(2i,2j+-2) + D_R^n(2i,2j)\right]$$

$$R^{n+1}(2i,2j-1) = G^n(2i,2j-1) + \hat{D_R^n}(2i,2j-1)$$

Gli autori dell'articolo [44] utilizzano una ricostruzione delle immagini fondata su banchi di filtri, trasparente alla tipologia di CFA adoperato. Osservando la correlazione esistente tra i canali, si nota la ridondanza in alta frequenza delle tre componenti, questo suggerisce che la ricostruzione delle superfici e dei bordi sono possibili se le componenti spaziali e spettrali dell'immagine vengono modellate in maniera congiunta.

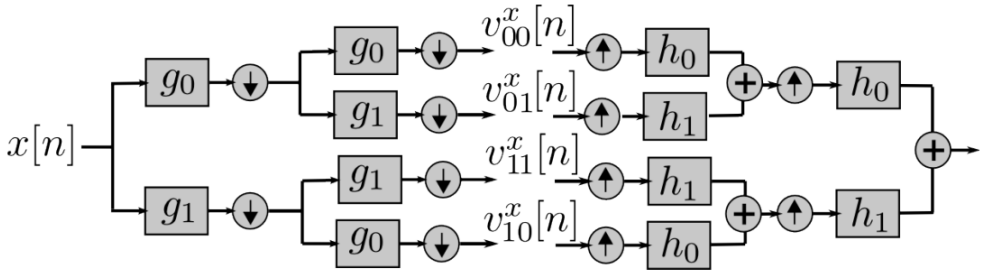


Figura 2.4.1: Banco di filtri utilizzato in [44]

Viene utilizzata una struttura a banchi di filtri come in figura 2.4.1 , se:

$$\sum_n g_i[n]z^{-n} = 1 + (-1)^i z\,, \quad \sum_n h_i[n]z^{-n} = \tfrac{z^{-1}+(-1)^i}{2}$$

allora $\{g_i, h_i\}_{i\in\mathbb{Z}^2}$ costituisce una trasformata di Haar. Se chiamiamo $v_i^x[n]$, $v_i^y[n]$ le sequenze di coefficienti del primo livello del banco di filtri di Haar a due dimensioni, corrispondenti a x e y rispettivamente, si ha la relazione:

$$v_i^{xy}[n] = v_i^x[n] \circledast v_i^y[n] := 4^{-I} \sum_{(i',j')\in\mathbb{Z}_2^I\times\mathbb{Z}_2^I} v_{i+i',j+j'}^x[n] v_{i',j'}^y[n]$$

che descrive la convoluzione bidimensionale a sottobande (dove $\circledast$ indica la convoluzione ciclica). Calcolando la trasformata per il segnale $y(n)$ all'uscita del banco di filtri si ottiene:

$$v_{i,j}^y[n] = \gamma v_{i,j}^{x_g}[n] + 4^{-I} v_{i,j}^{C_r}[n] v_{0,0}^{\alpha}[n] + 4^{-I} v_{i,j}^{C_b}[n] v_{0,0}^{\beta}[n]$$

ogni coefficiente $v_{i,j}^y[n]$ è una combinazione lineare di $v_{i,j}^{x_g}[n]$, $v_{0,0}^{\alpha}[n]$ e $v_{0,0}^{\beta}[n]$. Il problema della ricostruzione dell'immagine è equivalente alla ricerca di una stima di $v_{i,j}^{x_g}[n]$, $v_{0,0}^{\alpha}[n]$ e $v_{0,0}^{\beta}[n]$ a partire da $v_{i,j}^y[n]$. Sotto l'ipotesi di scarsità di coefficienti delle alte frequenze, la ricostruzione della crominanza è reinterpretata con uno schema *robust regressive* (funzione *robustfit* in Matlab), se si considerano informazioni *a priori* sull'immagine, ad esempio utilizzando [4], è possibile migliorare le prestazioni della *robust regression*.

2.5 Metodi di ricostruzione

Un altro modo di pensare alla ricostruzione di un'immagine campionata con un CFA è considerare il modello seguente [45]:

$$\mathcal{I}_s(n) = \mathcal{H}\mathcal{I}(n) + \eta(n)$$

dove $\mathcal{I}_s(n)$ è l'immagine raccolta dal sensore, n rappresenta la posizione del pixel, $\mathcal{I}(n)$ è l'immagine originale, $\eta(n)$ è il rumore incorrelato a $\mathcal{I}(n)$ che viene prodotto dal sensore

ed $\mathcal{H}$ è un operatore lineare che sostituisce il processo di acquisizione dell'immagine con un CFA. Il problema del demosaicking si traduce quindi nella ricerca di una versione stimata $\hat{\mathcal{I}}(n)$ dell'immagine a partire dalla versione sottocampionata $\mathcal{I}_s(n)$.

Una possibile soluzione del problema si ottiene applicando il criterio MAP (*maximum a posteriori probability*), dove $\mathcal{I}_s(n), \mathcal{I}(n), \eta(n)$ vengono considerati processi aleatori, la formulazione della stima MAP è quindi:

$$\hat{\mathcal{I}} = argmax_{\mathcal{I}}\, p(\mathcal{I}|\mathcal{I}_s)$$

utilizzando poi la formula di Bayes sulla probabilità condizionata $p(\mathcal{I}|\mathcal{I}_s)$ si ottiene:

$$\hat{\mathcal{I}} = argmax_{\mathcal{I}}\, p(\mathcal{I}_s|\mathcal{I})p(\mathcal{I})$$

quindi per ricostruire l'immagine $\hat{\mathcal{I}}$ bisogna avere conoscenza della PSD (*probability density function*) $p(\mathcal{I})$ dell'immagine originale $\mathcal{I}$; i metodi che sono fondati su queste ipotesi vengono spesso chiamati approcci Bayesani.

In [31] per trovare una stima Bayesiana, è necessario specificare la probabilità a priori $p(i)$, utilizzando modelli lineari ogni immagine viene considerata come una somma pesata di N_{basis} immagini base rappresentate a loro volta con colonne di $N_{\text{rows}}, N_{\text{cols}}, N_{\text{wls}}$di N_{basis} matrici base B ; in questo modo ogni immagine i può venire espressa come una somma pesata di queste colonne:

$$i = Bw$$

dove w è una multivariata Normale con media u_w e covarianza K_w; le immagini rappresentate con questa equazione fanno parte del modello lineare definito dalla matrice B. Si ottiene quindi che:

$$p(i) = N(Bu_w, BK_wB^T)$$

la distribuzione di probabilità appena scritta rispecchia due caratteristiche delle immagini naturali, la prima che la potenza media dello spettro decade rapidamente, la seconda che segnali in bande di colore differenti sono fortemente correlati tra loro.

Altri autori [34] utilizzano la teoria dei MRF (*Markov Random Fields*), le probabilità vengono modellate con:

$$p(w) = \frac{1}{Z} e^{\frac{-U(w)}{T}}$$

con Z costante e T parametro di temperatura, in questo modo la ricerca del massimo della probabilità si trasforma nella ricerca del minimo dell'energia $U(w)$, che nella teoria dei MRF viene definita come

$$U = \sum_{x_1,x_2} \sum_{k} \alpha_k U_k(x_1, x_2)$$

dove α_k pesano ciascuna energia.

La soluzione del problema inverso posto all'inizio di questa sezione può essere affrontata con il metodo MMSE (*minimum mean squared error*), ovvero cercando di minimizzare l'errore quadratico medio $E[||I - \hat{I}||^2]$, che porta alla seguente soluzione:

$$\hat{\mathcal{I}} = r_{\mathcal{I}\mathcal{I}_S}(r_{\mathcal{I}\mathcal{I}})^{-1}\mathcal{I}_s$$

dove $r_{\mathcal{I}\mathcal{I}_S}$è la funzione di correlazione tra $\mathcal{I}$ e $\mathcal{I}_s$ mentre $r_{\mathcal{I}\mathcal{I}}$è la funzione di autocorrelazione di $\mathcal{I}$ (per distribuzioni di tipo gaussiano le formule MAP e MMSE sono identiche). Nell'articolo [35] viene seguita questa strada, in particolare viene costituito un modello a priori WSS (*wide sense stationary*) con scala unica per ricostruire l'immagine attraverso LLMSE (*linear least mean squared error*), la densità spettrale di potenza dell'immagine originale viene descritta da:

$$\Gamma_Y(w_r, w_\theta) = w_r^{-2} f(w_\theta)\Gamma_0$$

con (w_r, w_θ) frequenze radiali e angolari, $f(w_\theta)$ distribuzione angolare (spesso considerata costante) e Γ_0 matrice autocorrelazione $D \times D$ derivante da un insieme adeguato di riflettanze della superficie spettrale, questa immagine viene quindi utilizzata per trovare uno stimatore LLMSE.

Nell'articolo [46] viene utilizzata la teoria *compressed sensing* per la ricostruzione dell'immagine, partendo dal presupposto che le immagini naturali ammettano una *sparse decomposition* con un dizionario ridondante, viene creato un apposito dizionario e quindi sfruttato l'algoritmo K-SVD (*single value decomposition*) per analizzare la correlazione esistente tra i canali ed infine ricostruire l'immagine.

Il problema iniziale $\mathcal{I}_s(n) = \mathcal{H}\mathcal{I}(n) + \eta(n)$ viene detto *ill-posed* ovvero mal posto, attraverso metodi di regolarizzazione è possibile convertirlo in un problema *well-posed* cioè ben posto, la regolarizzazione è stata ampiamente utilizzata nelle immagini digitali per *denoising*, *deblurring*, compensazione del movimento, ingrandimenti e *super-resolution*. La soluzione regolarizzata:

$$\hat{\mathcal{I}} = \text{argmin}_{\mathcal{I}} \left\{ \psi(\mathcal{I}, \mathcal{I}_s) + \sum_k \lambda_k J_k(\mathcal{I}) \right\}$$

con $\psi(\mathcal{I}, \mathcal{I}_s)$ termine di *data-fidelity*, che fornisce una misura della distanza tra l'immagine stimata e i dati osservati, $J_k(\mathcal{I})$ costanti di regolarizzazione derivanti dalla conoscenza a priori dell'immagine, λ_k *trade-off* tra i diversi termini.

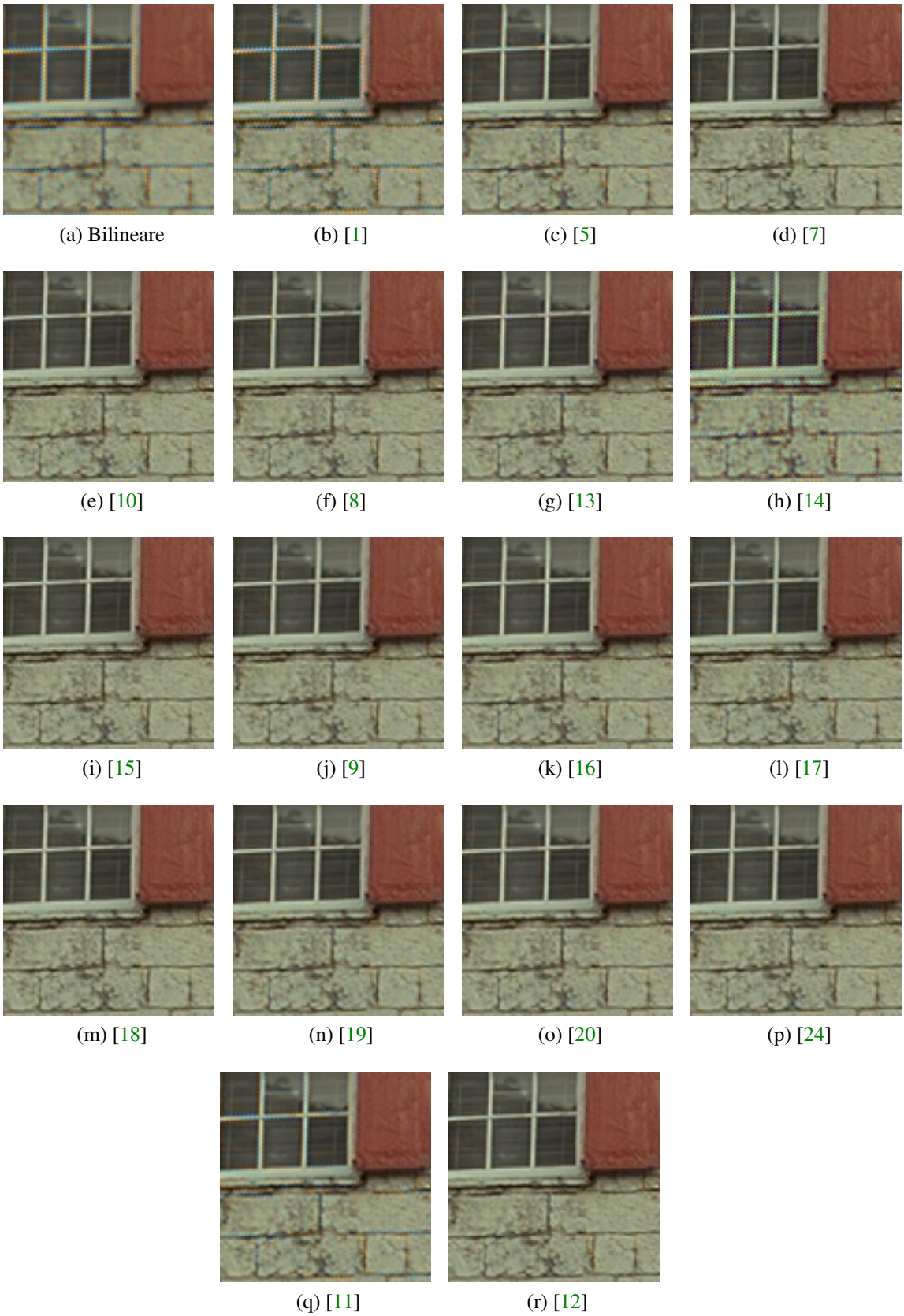
Negli articoli [20, 24] si fa uso inizialmente di una regolarizzazione quadratica con $J_k(\mathcal{I}) = ||M_n\mathcal{I}||_2^2$ e successivamente ad una regolarizzazione adattativa con $J_k(\mathcal{I}) = ||M_n\mathcal{I}||_{W_i}^2$ (dove W_i è una matrice diagonale che tiene conto delle caratteristiche locali) provvedendo inoltre ad una stima adattativa della luminanza, che porta a risultati migliori

lungo i contorni e le discontinuità dell'immagine.

2.6 Confronti

Sono state eseguite delle simulazioni con i metodi in mio possesso, alcuni da me implementati, altri invece forniti liberamente dagli stessi autori degli articoli[2].

[2]Un ringraziamento speciale a Daniele Menon per la disponibilità



(a) Bilineare (b) [1] (c) [5] (d) [7]

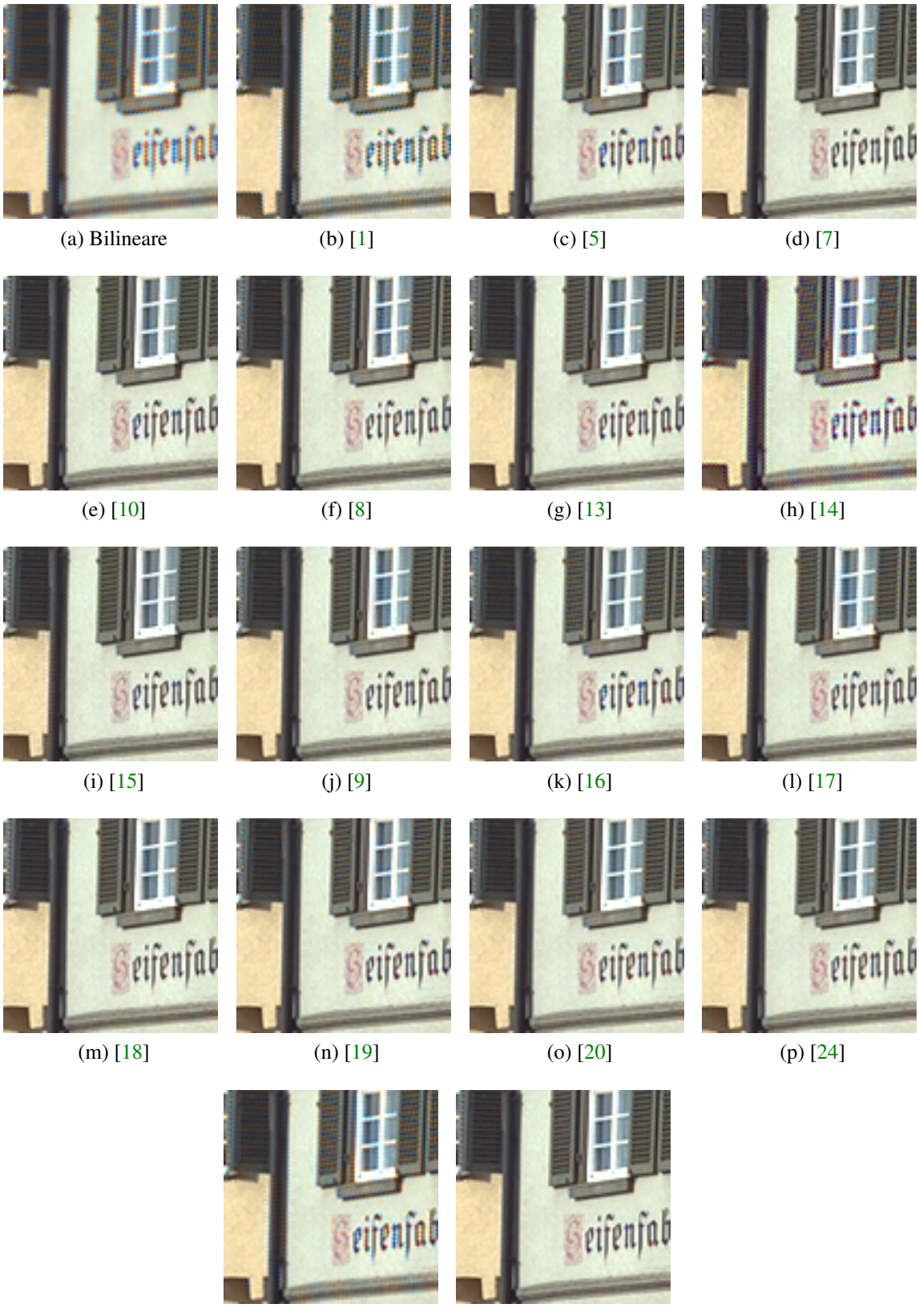
(e) [10] (f) [8] (g) [13] (h) [14]

(i) [15] (j) [9] (k) [16] (l) [17]

(m) [18] (n) [19] (o) [20] (p) [24]

(q) [11] (r) [12]

Figura 2.6.1: Demosaicking dell'immagine Kodak nr.1



(a) Bilineare (b) [1] (c) [5] (d) [7]

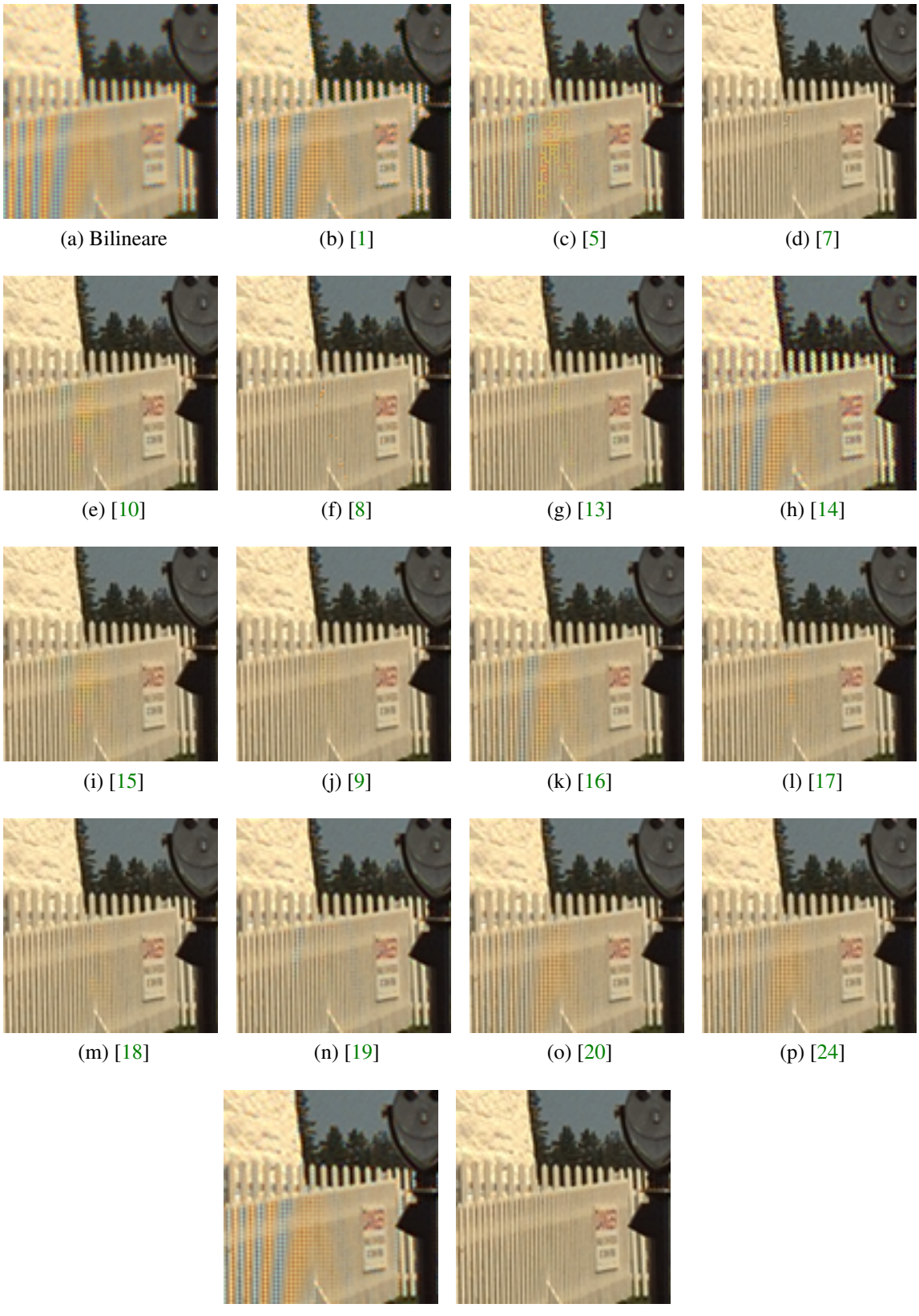
(e) [10] (f) [8] (g) [13] (h) [14]

(i) [15] (j) [9] (k) [16] (l) [17]

(m) [18] (n) [19] (o) [20] (p) [24]

(q) [11] (r) [12]

Figura 2.6.2: Demosaicking dell'immagine Kodak nr.8



(a) Bilineare (b) [1] (c) [5] (d) [7]

(e) [10] (f) [8] (g) [13] (h) [14]

(i) [15] (j) [9] (k) [16] (l) [17]

(m) [18] (n) [19] (o) [20] (p) [24]

(q) [11] (r) [12]

Figura 2.6.3: Demosaicking dell'immagine Kodak nr.1

Capitolo 3

Analisi della qualità

3.1 CIELAB

Introduzione agli spazi uniformi

E' naturale pensare ad una relazione delle distanze dei colori in tre dimensioni per capirne la loro differenza. Prima di far ciò è necessario comprendere quali siano i metodi per quantificare le differenza percepite dei colori. Per un'ampia varietà di differenti colori, gli osservatori interpretano il colore con soggettività e variabilità a seconda delle situazioni. Esiste però un valore pratico nel quantificare piccole differenze di colore, a questo proposito la nozione di JND (*just noticeably difference*) è stata introdotta come unità di misura da molti scienziati in questo campo [30].

Sono state effettuate molte ricerche della distribuzione della JND nella cromaticità CIE xy e negli spazi CIE XYZ ed è stato confermato che JND varia molto a seconda dello spazio di colore utilizzato.

Lo spazio CIE XYZ non è percepito uniformemente e le differenze percepite tra colori non corrispondono ad uguali distanze nello spazio tristimolo; poiché l'uniformità di percezione è una caratteristica indispensabile [32], per definire le tolleranze nel riprodurre i colori sono state condotte molte ricerche per lo sviluppo di uno spazio di colori uniforme, le difficoltà sono essenzialmente due: la determinazione di una scala di luminosità uniforme e un digramma di cromaticità uniforme per determinare un *equi-lightness color stimuli*.

Tali scale sono state definite attraverso sperimentazioni empiriche [33], si è scoperto che la legge che soddisfa tali relazioni è del tipo radice cubica; la CIE ha raccomandato due spazi di colore uniformi per applicazioni pratiche: lo spazio *CIE 1976 L*u*v* (CIELUV) e lo spazio *CIE 1976 L*a*b* (CIELAB).

Gli spazi sono definiti in termini di trasformazioni provenienti dal CIE XYZ tristimolo, entrambi gli spazi utilizzano una scala di luminosità in comune L^* che dipende solo dal valore di luminanza Y. La scala di luminosità è combinata con diversi diagrammi di cromaticità uniformi sino ad ottenere uno spazio uniforme di colori a tre dimensioni.

Le trasformazioni includono una normalizzazione del tristimolo del bianco, il quale provvede una approssimazione sommaria dell'adattamento dell'occhio umano.

Le distanze euclidee, in entrambi i casi, forniscono una formula per valutare le differenze cromatiche in unità percettivamente rilevanti. Sia il CIELUV che CIELAB includono anche termini di correlazione per le tre componenti percettive di luminosità, cromaticità e tinta.

3.1.1 Lo spazio CIE 1976 L*u*v

I valori *L*u*v** corrispondenti al tristimolo da i valori *XYZ* del CIE XYZ sono dati da:

$$L^* = 116f\left(\frac{Y}{Y_n}\right) - 16$$

$$u^* = 13L * (u' - u'_n)$$

$$v^* = 13L * (v' - v'_n)$$

dove:

$$f(x) = \begin{cases} x^{\frac{1}{3}} & x > 0.008856 \\ 7.787x + \frac{16}{116} & x \leq 0.008856 \end{cases}$$

$$u' = \frac{4X}{X + 15Y + 3Z}$$

$$v' = \frac{9Y}{X + 15Y + 3Z}$$

$$u'_n = \frac{4X_n}{X_n + 15Y_n + 3Z_n}$$

$$v'_n = \frac{9Y_n}{X_n + 15Y_n + 3Z_n}$$

e X_n, Y_n, Z_nsono i tristimoli dello stimolo del bianco, il quale è tipicamente lo stimolo più lucente nel campo visibile.

Le distanze euclidee tra due stimoli di colore nel CIELUV sono indicate come $\triangle E^{*uv}$ (deltaE-uv) ed è una misura della differenza complessiva di colore. Mediamente, un valore di $\triangle E^{*uv}$attorno a 2.9 corrisponde a un JND. Il valore L^*è un termine di correlazione della luminosità.

Nel piano u^*v^*, la distanza radiale $\sqrt{(u^*)^2 + (v^*)^2}$e la posizione angolare $\arctan(\frac{u^*}{v^*})$ correlano rispettivamente la cromaticità e la tinta.

3.1.2 Lo spazio CIE 1976 L*a*b

La coordinata L* dello spazio CIELAB è identica alla coordinata L* nello spazio CIE-LUV, e le trasformazioni per a* e b* sono date da:

$$a^* = 500\left(f\left(\frac{X}{X_n}\right)\right) - \left(f\left(\frac{Y}{Y_n}\right)\right)$$

$$b^* = 200 = 500\left(f\left(\frac{Y}{Y_n}\right)\right) - \left(f\left(\frac{Z}{Z_n}\right)\right)$$

Poiché CIELAB è usato principalmente nelle immagini, le trasformazioni verso e dallo spazio CIELAB verso altri spazi di colori sono molto utilizzate. È quindi utile specificare le relazioni inverse che convertono uno spazio CIELAB in uno CIE XYZ:

$$f_Y = \frac{L^* + 16}{116}$$

$$f_X = \frac{a^*}{500} + f_Y$$

$$f_Z = f_Y - \frac{b^*}{200}$$

$$X = X_n f^{-1}(f_x)$$

$$Y = Y_n f^{-1}(f_Y)$$

$$Z = Z_n f^{-1}(f_Z)$$

con trasformazione inversa di $f(\cdot)$ data da :

$$f^{-1}(t) = \begin{cases} t^3 & t > 0.206893 \\ \frac{1}{7.787}\left(t + \frac{16}{116}t\right) & 0 \leq t \leq 0.206893 \end{cases}$$

Nel CIELAB, le distanze radiali sono espresse nel piano a*b* come segue:

$$C^*_{ab} = \sqrt{(a^*)^2 + (b^*)^2}$$

ed esprime un termine di correlazione o misura della cromaticità percepita. La posizione angolare nel piano a*b* è data da:

$$h^*_{ab} = \arctan\left(\frac{a^*}{b^*}\right)$$

e rappresenta il termine di correlazione della tinta percepita. La distanza Euclidea tra due stimoli di colori è chiamata $\triangle E^*_{ab}$(delta E-ab), per un colore campionato con valori L^*_2, a^*_2, b^*_2, la differenza da un colore standard con valori in CIELABL^*_s, a^*_s, b^*_s è data da:

$$\triangle E^*_{ab} = \sqrt{(\triangle L^*)^2 + (\triangle a^*)^2 + (\triangle b^*)^2}$$

con:

$$\triangle L^* = L^*_2 - L^*_s$$

$$\triangle a^* = a^*_2 - a^*_s$$

$$\triangle b^* = b^*_2 - b^*_s$$

Un valore di $\triangle E^*_{ab}$ attorno a 2.3 corrisponde ad un JND. Questa correlazione è ad ogni modo una approssimazione, ed esistono significanti variazioni in un JND su di uno spazio di colore.

Lo spazio dei colori CIELAB incorpora anche una codifica ad opposizione di colore (figura 3.1.1). Nei tre assi principali sono situate le componenti L*, a*, b*: la componente a* corrisponde alle tinte rosse e verdi aventi direzioni opposte, le distanze lungo la parte positiva di a* corrispondono a misure di rossi, viceversa le distanze lungo il semiasse negativo corrispondono a misure di verdi; lo stesso ragionamento si applica all'asse b* al quale sono associate le tinte di giallo e di blu. I punti che si trovano sull'asse L* (cioè aventi a*=b*=0) appartengono alla scala dei grigi e sono definiti come neutrali o acromatici. Ad esempio, un valore CIELAB di 50,0,70 rappresenta un colore giallo con luminosità media; un valore CIELAB di 90,-10,-70 descrive invece un colore pastello verde-bluastro.

3.1.3 S-CIELAB

Un modello molto popolare e semplice è l'estensione spaziale del CIELAB e viene chiamato S-CIELAB. Tale modello per prima cosa trasforma un'immagine a colori in tre piani opposti di colore: O_1, O_2, O_3, corrispondenti rispettivamente alle componenti nero-bianco (luminanza), rosso-verde, giallo-blu.

Queste tre coordinate di colore sono definite come una trasformazione lineare dei valori tristimolo CIE XYZ in questo modo:

$$O_1 = 0.279X + 0.72Y - 0.107Z$$

$$O_2 = -0.449X + 0.29Y - 0.077Z$$

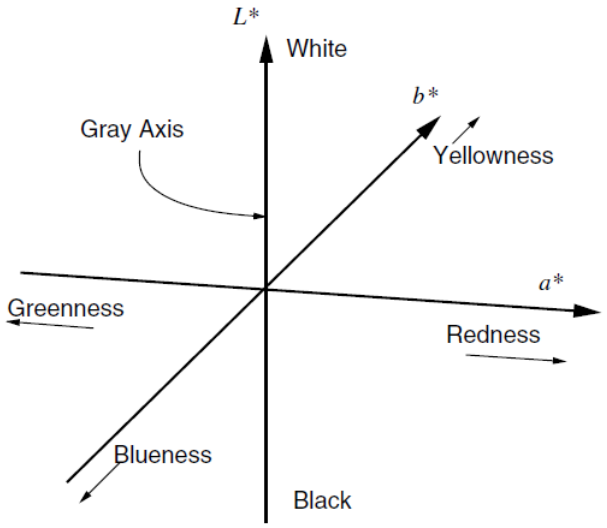


Figura 3.1.1: Interpretazione degli assi CIELAB

Tabella 3.1: Parametri per il modello S-CIELAB HVS

	w_i^j	σ_i^j
Luminanza $O_1(i=3)$	0.921	0.0283
	0.105	0.133
	-0.108	4.336
Rosso-Verde $O_2(i=2)$	0.531	0.0392
	0.330	0.494
Blu-Giallo $O_3(i=2)$	0.488	0.0536
	0.371	0.386

$$O_3 = 0.086X + 0.592Y - 0.501Z$$

Ognuna delle tre componenti è quindi filtrata con la funzione a due variabili gaussiana:

$$f^i(x,y) = k^i \sum_i w_j^i E_i^j(x,y)$$

$$E_i^j(x,y) = k_i^j \exp\left(\frac{-(x^2+y^2)}{\sigma_i^{2j}}\right)$$

dove $f^i(x,y)$ per i=1,2,3 sono i kernel per O_1, O_2, O_3.

Le variabili x e y indicano le dimensioni spaziali, σ_i^j indica la deviazione standard. Il termine k è scelto in modo tale che $E_i^j(x,y)$ sommino ad uno. I valori di w_i^j,σ_i^j sono scelti come mostrato in tabella 3.1.

A seguito di questo filtraggio:

$$O'_j(x,y) = (O_j(x,y) * f^j(x,y)) \qquad j = 1,2,3$$

le immagini filtrate O'_1, O'_2, O'_3 sono quindi ritrasformate in CIE XYZ utilizzando le for-

mule inverse. Le coordinate CIE XYZ sono quindi trasformate in CIELAB utilizzando le formule prima descritte.

L'immagine di differenza di colore ottenuta da questo processo rappresenta una mappa spaziale della differenza visiva o della distorsione tra le immagini. La mappa di distorsione viene utilizzata direttamente per determinare le regioni dove è percepita la differenza, o dove esistono errori nella ricostruzione dei pixel.

Il modello S-CIELAB è molto utilizzato per la sua semplicità e somiglianza allo standard CIELAB già largamente adottato.

S-CIELAB è una estensione della formula CIE*L*a*b*DeltaE.

Tale estensione è compresa nel passaggio di *pre-processing* presente nel *Pattern-color sensitivity measurements* di Allen Poirson e Brian Wandell.

Come nella metrica CIELAB Delta E, la metrica S-CIELAB è del tipo *perceptual color fidelity*, cioè è in grado di misurare l'accuratezza della riproduzione del colore quando vista da un osservatore umano.

La separazione dei colori è determinata da molteplici fattori, ad esempio il cono di sensibilità dell'occhio umano, l'illuminazione dell'ambiente circostante e molti altri fattori. Le differenze di colore e la loro percezione sono funzioni dello *spatial pattern*. In generale, se la frequenza spaziale dell'immagine cresce (variazioni sempre più piccole nello spazio), è sempre più arduo notarne la differenza di colore, soprattutto lungo le direzioni di colore blu-giallo. Questo risultato deriva da molti studi, di autori quali Poirson e Wandell (1993, 1996), Kelly (1983).

Quindi, per poter applicare le metriche CIE L*a*b* Delta E alle immagini a colore, bisogna innanzitutto considerare lo *spatial pattern* dell'immagine. L'obiettivo della metrica S-CIELAB è quello di aggiungere il passaggio di *pre-processing* allo standard CIELab Delta E che tenga conto della sensibilità cromatica spaziale dell'occhio umano.

Il calcolo del S-CIELAB è illustrato in figura 3.1.2.

Le componenti principali sono la trasformazione di colore e il filtraggio spaziale prima di effettuare il calcolo dello standard CIELAB Delta E (è un'architettura di tipo *Pattern-Color Separable*).

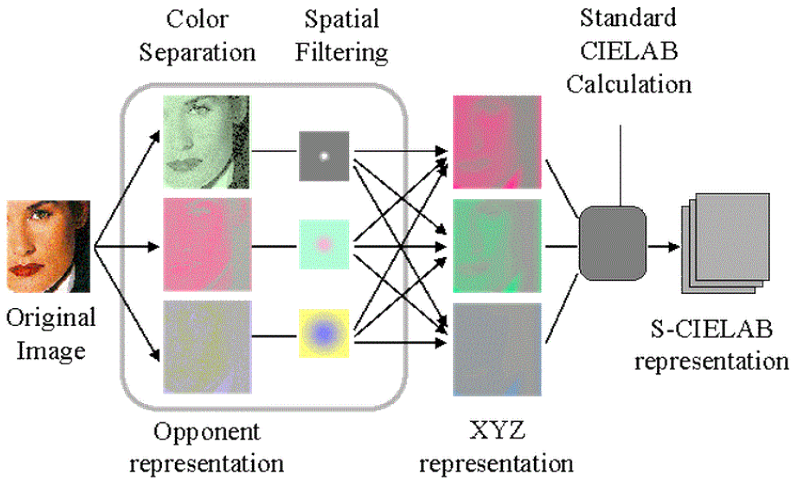


Figura 3.1.2: Modello S-CIELAB[36]

Risultati

Method	Bil.	[1]	[5]	[7]	[10]	[8]	[13]	[14]	[15]
1	2,886	2,703	1,807	1,250	1,307	1,199	1,296	3,091	1,282
2	1,212	1,372	0,822	0,821	0,802	0,720	0,686	1,141	1,003
3	0,936	0,879	0,612	0,561	0,613	0,519	0,534	1,055	0,728
4	1,204	1,397	0,907	0,912	0,911	0,818	0,812	1,263	1,029
5	2,551	2,740	1,803	1,295	1,244	1,052	1,126	2,912	1,678
6	2,167	1,985	1,289	0,876	1,011	0,788	0,880	2,035	0,973
7	1,080	1,002	0,744	0,691	0,685	0,627	0,628	1,628	0,902
8	3,789	3,504	2,088	1,424	1,562	1,342	1,459	3,562	1,637
9	1,128	1,094	0,768	0,622	0,707	0,588	0,590	1,382	0,768
10	1,058	1,033	0,736	0,622	0,669	0,579	0,592	1,284	0,677
11	1,747	1,700	1,147	0,846	0,862	0,767	0,799	1,879	0,919
12	1,042	0,951	0,653	0,528	0,562	0,497	0,535	1,058	0,603
13	3,630	3,242	2,525	1,896	1,802	1,709	1,880	3,579	1,717
14	1,950	1,823	1,294	1,161	1,188	1,079	1,050	2,072	1,607
15	1,144	1,253	0,821	0,808	0,798	0,737	0,753	1,223	0,876
16	1,422	1,337	0,907	0,576	0,699	0,534	0,583	1,449	0,662
17	1,129	1,036	0,799	0,599	0,625	0,573	0,594	1,316	0,615
18	2,113	1,888	1,518	1,315	1,153	1,138	1,156	2,238	1,361
19	1,931	1,734	1,203	0,866	0,876	0,808	0,825	2,054	0,880
20	1,091	1,009	0,841	0,649	0,709	0,632	0,698	1,228	0,706
21	1,933	1,818	1,327	1,008	1,013	0,937	1,023	1,993	1,004
22	1,639	1,445	1,151	1,175	1,082	1,065	1,043	1,686	1,241
23	0,753	0,754	0,577	0,599	0,597	0,550	0,541	0,880	0,760
24	2,037	1,826	1,393	1,289	1,230	1,129	1,241	2,124	1,363

Tabella 3.2: SCIELAB delle immagini Kodak, algoritmi 0-8

...	[9]	[16]	[17]	[18]	[19]	[20]	[24]	[11]	[12]	Best
...	1,132	1,212	1,177	1,054	1,281	1,245	1,222	1,946	1,054	[12]
...	0,680	0,666	0,731	0,754	0,735	0,771	0,801	0,825	0,649	[12]
...	0,493	0,490	0,536	0,518	0,539	0,542	0,558	0,613	0,478	[12]
...	0,775	0,767	0,786	0,808	0,812	0,751	0,773	0,836	0,757	[20]
...	1,071	1,006	1,047	1,085	1,203	1,069	1,116	1,385	1,021	[16]
...	0,800	0,961	0,793	0,795	0,883	0,885	0,882	1,513	0,757	[12]
...	0,584	0,531	0,549	0,626	0,643	0,570	0,595	0,692	0,560	[16]
...	1,332	1,392	1,468	1,301	1,437	1,408	1,396	2,575	1,220	[12]
...	0,563	0,582	0,621	0,590	0,629	0,584	0,592	0,804	0,548	[12]
...	0,559	0,556	0,598	0,590	0,628	0,552	0,558	0,732	0,540	[12]
...	0,736	0,791	0,753	0,734	0,805	0,797	0,798	1,182	0,704	[12]
...	0,473	0,513	0,474	0,501	0,519	0,533	0,540	0,751	0,466	[12]
...	1,553	1,670	1,569	1,496	1,791	1,672	1,617	2,307	1,472	[12]
...	0,967	0,987	1,044	1,042	1,088	1,018	1,050	1,238	0,922	[12]
...	0,696	0,671	0,685	0,765	0,736	0,704	0,721	0,792	0,688	[16]
...	0,534	0,680	0,523	0,520	0,606	0,590	0,586	0,996	0,507	[12]
...	0,530	0,555	0,548	0,566	0,631	0,527	0,521	0,771	0,532	[24]
...	1,052	1,058	1,058	1,107	1,146	1,031	1,038	1,296	1,028	[12]
...	0,747	0,787	0,807	0,757	0,846	0,790	0,785	1,330	0,714	[12]
...	0,590	0,558	0,618	0,653	0,677	0,598	0,607	0,743	0,628	[16]
...	0,899	0,922	0,924	0,896	0,988	0,939	0,933	1,293	0,844	[12]
...	0,950	0,920	0,924	1,020	0,961	0,940	0,961	1,113	0,940	[16]
...	0,493	0,496	0,519	0,549	0,527	0,527	0,550	0,515	0,499	[9]
...	1,056	1,042	1,035	1,164	1,104	1,064	1,080	1,318	1,054	[17]

Tabella 3.3: SCIELAB delle immagini Kodak, algoritmi 9-17

3.2 Zipper Effect

Lo *Zipper Effect* o ZE è un artificio derivante dalla ricostruzione delle immagini che si presenta come un effetto *on-off*.

Per un pixel isolato, lo *Zipper Effect*, si manifesta come un aumento del valore della differenza di colore tra il pixel in esame e il più simile pixel che appartiene alla sua cornice.

Solitamente lo ZE ha luogo attorno alle regioni dei contorni netti, ovvero dove entra in crisi il processo di *demosaicking* a causa dei limiti imposti dal CFA e dalla struttura geometrica dei foto-ricevitori.

La stima di questo effetto, avviene col metodo descritto come *with reference*, ovvero si necessita dell'immagine originale e non ricostruita, detto questo, si capisce che il metodo fornisce una valutazione oggettiva e univoca.

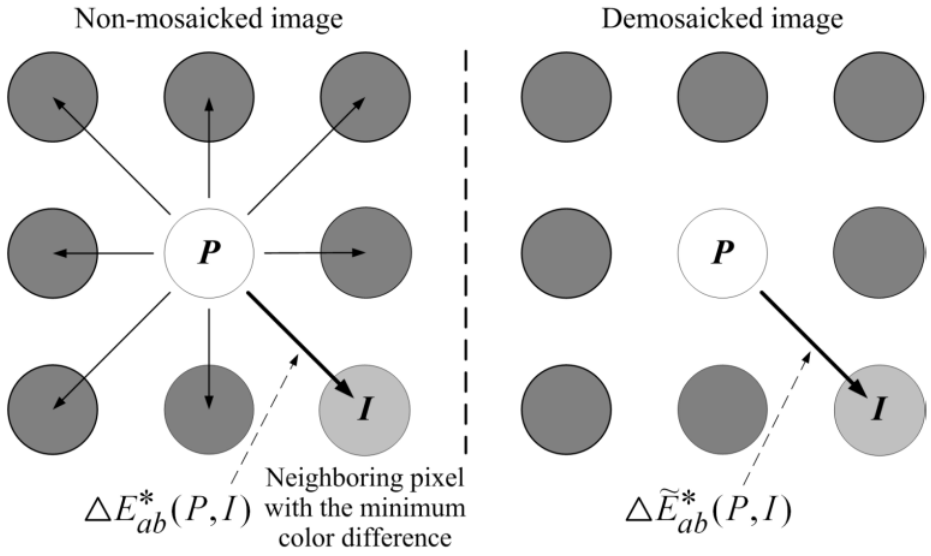


Figura 3.2.1: Blocco di pixel in esame

Dopo tali osservazioni, l'algoritmo che porta al calcolo dello ZE è il seguente [37]:

- Si consideri nell'immagine originale non ricostruita, il pixel P, e si calcoli la differenza (assoluta) da i suoi otto pixel vicini (che chiamiamo N), come mostrato in figura 3.2.1

- Chiamiamo I il pixel che ha la minor differenza di colore nello spazio *Lab*, e lo definiamo come segue:
$$I = \min_{i \in N} \triangle E^*_{ab}(P, i)$$

- Quindi viene calcolata la differenze di colore CIELAB $\triangle E^*_{ab}$ sulla stessa coppia di pixel, P e I, nell'immagine ricostruita dopo il demosaicking, e lo definiamo come:
$$\triangle \tilde{E}^*_{ab}(P, i)$$

- La differenza di colore tra i pixel P e I data da:
$$\psi = \triangle \tilde{E}^*_{ab}(P, i) - \triangle E^*_{ab}(P, i)$$

- Quando la differenza $|\psi| > \delta$ dove $\delta = 2.3$, il pixel P nell'immagine ricostruita ha subito una notevole variazione di colore rispetto agli altri pixel vicini.

- La percentuale di pixel con $\psi > \delta$ su i pixel totali, costituisce il valore dello ZE

A seconda del segno di ψ il pixel P potrebbe aver aumentato o ridotto la differenza di colore, in particolare $\psi < -\delta$ indica che il pixel P ha ridotto il contrasto dal pixel I, mentre $\psi > \delta$ determina l'esistenza dello *Zipper Effect*.

Risultati

Dalle prove effettuate sulle prime 24 immagini Kodak [23] (su di una sezione di 100x100 pixel) ed applicando i metodi di demosaicking in mio possesso si ottengono i seguenti risultati:

Image\Algh	Bil.	[1]	[5]	[7]	[10]	[8]	[13]	[14]
1	0,384	0,256	0,209	0,150	0,038	0,070	0,052	0,304
2	0,235	0,195	0,145	0,150	0,102	0,108	0,095	0,194
3	0,212	0,116	0,078	0,019	0,009	0,008	0,009	0,129
4	0,045	0,033	0,016	0,015	0,011	0,009	0,010	0,033
5	0,446	0,301	0,238	0,181	0,093	0,093	0,091	0,406
6	0,067	0,023	0,015	0,015	0,010	0,009	0,006	0,024
7	0,042	0,024	0,020	0,018	0,011	0,015	0,013	0,028
8	0,206	0,148	0,140	0,103	0,100	0,075	0,077	0,145
9	0,064	0,038	0,041	0,016	0,042	0,010	0,006	0,045
10	0,077	0,037	0,029	0,026	0,019	0,021	0,018	0,049
11	0,084	0,027	0,024	0,007	0,004	0,005	0,001	0,028
12	0,171	0,174	0,179	0,190	0,190	0,179	0,178	0,159
13	0,349	0,229	0,185	0,158	0,048	0,074	0,068	0,307
14	0,311	0,205	0,177	0,180	0,083	0,107	0,095	0,246
15	0,419	0,415	0,326	0,404	0,359	0,333	0,304	0,370
16	0,158	0,057	0,048	0,032	0,027	0,021	0,020	0,049
17	0,214	0,147	0,109	0,086	0,069	0,044	0,030	0,141
18	0,181	0,127	0,111	0,150	0,118	0,105	0,080	0,120
19	0,570	0,408	0,318	0,044	0,057	0,025	0,021	0,299
20	0,327	0,219	0,210	0,202	0,111	0,122	0,105	0,268
21	0,405	0,260	0,230	0,173	0,075	0,107	0,092	0,324
22	0,041	0,022	0,020	0,040	0,029	0,028	0,017	0,031
23	0,071	0,030	0,025	0,015	0,019	0,014	0,013	0,044
24	0,302	0,204	0,188	0,176	0,130	0,120	0,106	0,245

Tabella 3.4: ZE delle immagini kodak, algoritmi 0-8

3.3 SSIM

L'indice SSIM (*Structural SIMilarity*) fornisce un metodo per la misurazione di somiglianze tra due immagini, può essere pensato come una misura di qualità di una delle immagini da comparare con un immagine di qualità perfetta. L'indice SSIM è derivato da una versione migliorata dell'*universal image quality index* proposto in [39].

Il metodo di confronto più semplice tra due immagini è MSE, però due immagini aventi lo stesso MSE posso avere diversi tipi di errori, alcuni più visibili altri meno.

Molti approcci di misurazione proposti in letteratura sono derivanti da misurazioni empiriche psicofisiche sugli umani o psicologiche su animali.

...	[15]	[9]	[16]	[17]	[18]	[19]	[20]	[24]	[11]	[12]
...	0,027	0,026	0,045	0,030	0,029	0,060	0,036	0,032	0,033	0,033
...	0,130	0,085	0,093	0,106	0,100	0,089	0,094	0,095	0,096	0,096
...	0,009	0,007	0,008	0,007	0,009	0,008	0,007	0,007	0,007	0,007
...	0,022	0,008	0,011	0,007	0,012	0,011	0,006	0,007	0,010	0,010
...	0,109	0,085	0,086	0,081	0,088	0,099	0,089	0,086	0,079	0,079
...	0,013	0,005	0,006	0,008	0,010	0,004	0,009	0,008	0,008	0,008
...	0,017	0,011	0,012	0,007	0,012	0,009	0,012	0,011	0,014	0,014
...	0,121	0,082	0,082	0,088	0,093	0,075	0,072	0,073	0,088	0,088
...	0,045	0,020	0,008	0,031	0,012	0,008	0,025	0,032	0,013	0,013
...	0,026	0,015	0,016	0,014	0,021	0,019	0,017	0,017	0,019	0,019
...	0,003	0,003	0,003	0,002	0,003	0,002	0,002	0,001	0,001	0,001
...	0,184	0,182	0,179	0,175	0,193	0,178	0,181	0,185	0,179	0,178
...	0,046	0,044	0,059	0,045	0,046	0,067	0,053	0,050	0,042	0,042
...	0,094	0,074	0,080	0,073	0,080	0,083	0,078	0,079	0,078	0,078
...	0,429	0,308	0,288	0,318	0,372	0,278	0,301	0,311	0,336	0,336
...	0,029	0,022	0,024	0,022	0,031	0,017	0,028	0,029	0,024	0,024
...	0,059	0,044	0,046	0,059	0,053	0,028	0,053	0,056	0,055	0,055
...	0,159	0,088	0,079	0,100	0,115	0,064	0,080	0,086	0,104	0,104
...	0,050	0,019	0,070	0,024	0,025	0,021	0,059	0,055	0,015	0,015
...	0,116	0,098	0,110	0,099	0,106	0,092	0,105	0,105	0,106	0,106
...	0,071	0,067	0,089	0,064	0,077	0,087	0,078	0,078	0,075	0,075
...	0,044	0,023	0,013	0,018	0,036	0,014	0,019	0,022	0,026	0,026
...	0,031	0,014	0,013	0,017	0,021	0,012	0,015	0,017	0,020	0,020
...	0,164	0,104	0,111	0,106	0,133	0,097	0,104	0,110	0,113	0,113

Tabella 3.5: ZE delle immagini kodak,algoritmi 9-17

In figura 3.3.1 viene illustrato un metodo generico di valutazione della qualità sensibile agli errori. Prendiamone in esame le singole componenti:

- Pre-processing: elimina le distorsioni note dalle immagini da comparare.
- CSF (*contrast sensitivity function*) filtering: descrive la sensibilità dell' HVS (*human visual system*) in diverse frequenze spaziali e temporali presenti nello stimolo visivo.
- Channel Decomposition: le immagini vengono scomposte in canali o sottobande con trasformate quali DCT o Wavelet.
- Error Normalization: Viene calcolato l'errore tra l'immagine di riferimento decomposta e i segnali distorti in ogni canale secondo un certo modello di maschera. Il processo di normalizzazione serve per convertire l'errore in unità JND.
- Error Pooling: l'ultimo passo deve combinare i segnali di errore normalizzato sull'estensione spaziale dell'immagine, e attorno ai diversi canali, in un singolo valore

(metrica di Minkowski):

$$E\left(\{e_{l,k}\}\right) = \left(\sum_{l}\sum_{k}|e_{l,k}|^{\beta}\right)^{\frac{1}{\beta}}$$

dove $e_{l,k}$ è l'errore normalizzato del k-esimo coefficiente nel canale l, e β è una costante (tra 1 e 4).

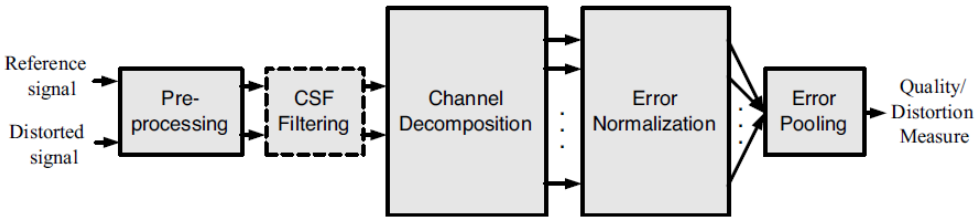


Figura 3.3.1: Schema generico di un sistema di valutazione basato sugli errori.

Le limitazioni di questo modello:

- Definizione del problema della qualità: non è chiaro come un errore visibile causi una perdita di qualità.
- Il problema *Suprathreshold*: molti esperimenti hanno permesso di identificare le soglie alle quali lo stimolo è appena visibile, tuttavia altri studi hanno dimostrato che i livelli di soglia che caratterizzano le distorsioni percepite sono significativamente maggiori.
- Il problema della complessità delle immagini naturali: gli esperimenti psicofisici sono stati effettuati su semplici pattern i quali non riescono a rappresentare la complessità e l'imprevedibilità di un'immagine naturale.
- Il problema della decorrelazione: con la metrica di Minkowski per semplicità si assume che gli errori siano statisticamente indipendenti, questo non è vero, come si è già osservato nella trasformata wavelet, esiste una forte dipendenza intra- e inter-canale dei coefficienti wavelet nelle immagini naturali.
- Il problema dell'interazione cognitiva: è risaputo che la comprensione cognitiva e il processo visivo (come il movimento degli occhi) influenza la qualità percepita dell'immagine. Precedenti informazioni riguardo al contenuto dell'immagine, il livello di attenzione e la durata della visione, possono alterare il giudizio sulla qualità. La maggior parte di questi fattori non viene considerata dai metodi di valutazione.

Il metodo proposto basato sull'indice SSIM è basato sull'dea che il sistema visivo umano è particolarmente adattato ad estrarre le informazioni strutturali dal campo visivo, quindi

una misura sul cambio delle informazioni strutturali di una immagine può dare una buona approssimazione della distorsione percepita dell'immagine.

La luminosità della superficie di un oggetto è il prodotto dell'illuminazione e della riflettenza, ma la struttura di un oggetto è indipendente dall'illuminazione, per questo motivo bisogna separare l'influenza dell'illuminazione, viene quindi usata la lunimosità locale e il contrasto.

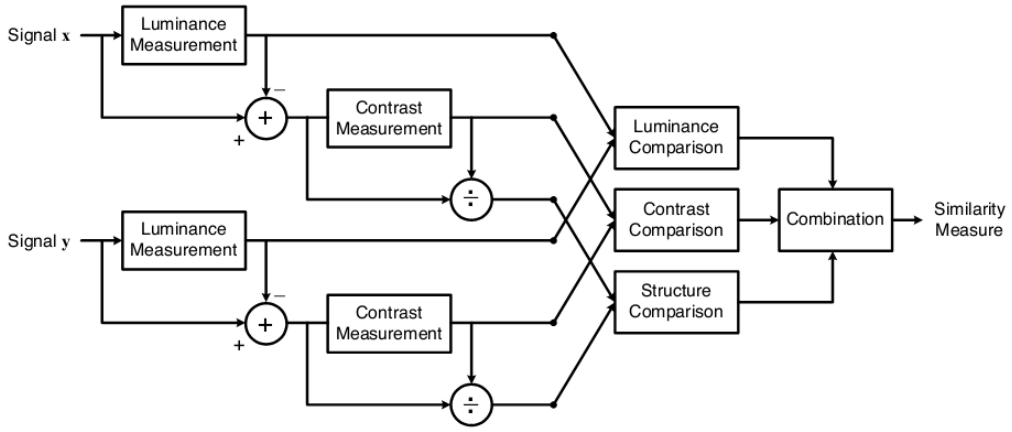


Figura 3.3.2: Diagramma della struttura del sistema SSIM.

Il diagramma del sistema è riportato in figura 3.3.2, x e y sono le due immagini in esame, se consideriamo che uno dei due segnali ha qualità perfetta, allora questo metodo fornisce una misura di qualità per l'altra immagine.

Il sistema separa il processo in tre comparazioni: luminosità, contrasto e struttura. La luminosità è stimata come l'intesità media:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

la funzione di paragone di luminosità $l(x, y)$ è quindi una funzione di μ_x, μ_y. Con il secondo passo si rimuove l'intesità media dall'immagine. Si usa quindi la deviazione standard come una stima del contrasto dell'immagine:

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}$$

Il *contrast comparison* $c(x, y)$è quindi il confronto tra σ_x, σ_y. Nel terzo passo il segnale viene normalizzato (diviso) per la sua deviazione standard, quindi la *structure comparison* $s(x, y)$ è effettuata su questi segnali normalizzati $\frac{x-\mu_x}{\sigma_x}$ e $\frac{x-\mu_y}{\sigma_y}$.

Infine i tre componenti vengono combinati per produrre una misura complessiva di somiglianza:

$$S(x, y) = f\left(l(x, y), C(x, y), s(x, y)\right)$$

Tabella 3.6: SSIM index delle immagini Kodak, algoritmi 0-8

Method	Bil.	[1]	[5]	[7]	[10]	[8]	[13]	[14]	[15]
1	0,9123	0,9601	0,9785	0,9911	0,9924	0,9906	0,9913	0,9620	0,9936
2	0,9521	0,9627	0,9827	0,9848	0,9841	0,9858	0,9879	0,9736	0,9786
3	0,9710	0,9849	0,9907	0,9944	0,9943	0,9948	0,9950	0,9837	0,9929
4	0,9851	0,9857	0,9915	0,9900	0,9911	0,9920	0,9923	0,9883	0,9877
5	0,9551	0,9790	0,9907	0,9932	0,9949	0,9949	0,9952	0,9799	0,9915
6	0,9258	0,9614	0,9815	0,9928	0,9924	0,9931	0,9935	0,9637	0,9935
7	0,9848	0,9916	0,9954	0,9952	0,9960	0,9960	0,9963	0,9899	0,9943
8	0,9251	0,9661	0,9831	0,9922	0,9934	0,9924	0,9928	0,9680	0,9938
9	0,9865	0,9906	0,9942	0,9959	0,9956	0,9961	0,9960	0,9893	0,9956
10	0,9870	0,9913	0,9951	0,9963	0,9962	0,9966	0,9966	0,9900	0,9965
11	0,9440	0,9704	0,9857	0,9922	0,9928	0,9927	0,9937	0,9711	0,9925
12	0,9587	0,9779	0,9884	0,9935	0,9939	0,9940	0,9944	0,9793	0,9940
13	0,8993	0,9470	0,9675	0,9837	0,9886	0,9846	0,9854	0,9416	0,9912
14	0,9524	0,9758	0,9870	0,9903	0,9911	0,9910	0,9924	0,9742	0,9844
15	0,9676	0,9765	0,9855	0,9863	0,9886	0,9883	0,9893	0,9771	0,9864
16	0,9403	0,9690	0,9846	0,9943	0,9936	0,9946	0,9948	0,9701	0,9942
17	0,9897	0,9934	0,9958	0,9971	0,9972	0,9972	0,9971	0,9921	0,9974
18	0,9774	0,9860	0,9904	0,9917	0,9930	0,9930	0,9928	0,9839	0,9908
19	0,9674	0,9794	0,9892	0,9941	0,9940	0,9943	0,9938	0,9816	0,9947
20	0,9689	0,9809	0,9864	0,9897	0,9907	0,9900	0,9903	0,9792	0,9908
21	0,9589	0,9779	0,9866	0,9907	0,9924	0,9914	0,9916	0,9755	0,9924
22	0,9524	0,9743	0,9836	0,9846	0,9883	0,9868	0,9881	0,9724	0,9866
23	0,9865	0,9904	0,9941	0,9927	0,9939	0,9936	0,9941	0,9899	0,9913
24	0,9498	0,9755	0,9860	0,9905	0,9924	0,9918	0,9922	0,9741	0,9915

Risultati

Dalle prove effettuate sulle prime 24 immagini Kodak [23], utilizzando $K = [0.01\, 0.03]$ e $window = fspecial('gaussian', 11, 1.5)$ ed applicando i metodi di demosaicking in mio possesso si ottengono i seguenti risultati riportati nelle tabelle 3.6 e 3.7.

Tabella 3.7: SSIM index delle immagini Kodak,algoritmi 9-17

...	[9]	[16]	[17]	[18]	[19]	[20]	[24]	[11]	[12]	Best
...	0,9941	0,9924	0,9930	0,9947	0,9914	0,9926	0,9933	0,9755	0,9937	[18]
...	0,9888	0,9880	0,9859	0,9871	0,9879	0,9859	0,9849	0,9816	0,9884	[9]
...	0,9957	0,9952	0,9949	0,9957	0,9953	0,9952	0,9950	0,9910	0,9958	[12]
...	0,9930	0,9932	0,9933	0,9924	0,9934	0,9933	0,9929	0,9920	0,9856	[19]
...	0,9958	0,9959	0,9957	0,9958	0,9956	0,9959	0,9958	0,9906	0,9959	[16]
...	0,9947	0,9915	0,9946	0,9950	0,9933	0,9938	0,9941	0,9762	0,9948	[18]
...	0,9967	0,9970	0,9965	0,9965	0,9968	0,9967	0,9965	0,9954	0,9968	[16]
...	0,9945	0,9930	0,9933	0,9950	0,9929	0,9938	0,9942	0,9763	0,9946	[18]
...	0,9967	0,9965	0,9965	0,9970	0,9966	0,9963	0,9963	0,9930	0,9903	[18]
...	0,9969	0,9971	0,9970	0,9972	0,9971	0,9968	0,9968	0,9938	0,9921	[18]
...	0,9944	0,9931	0,9942	0,9946	0,9937	0,9937	0,9937	0,9819	0,9948	[12]
...	0,9952	0,9938	0,9952	0,9954	0,9947	0,9947	0,9948	0,9854	0,9955	[12]
...	0,9899	0,9880	0,9900	0,9913	0,9864	0,9887	0,9900	0,9706	0,9907	[18]
...	0,9934	0,9930	0,9918	0,9928	0,9925	0,9923	0,9919	0,9863	0,9936	[12]
...	0,9906	0,9906	0,9902	0,9897	0,9906	0,9908	0,9904	0,9872	0,9910	[12]
...	0,9957	0,9924	0,9957	0,9959	0,9946	0,9948	0,9950	0,9791	0,9958	[18]
...	0,9977	0,9975	0,9975	0,9977	0,9974	0,9975	0,9976	0,9953	0,9977	[9]
...	0,9943	0,9942	0,9942	0,9941	0,9942	0,9942	0,9943	0,9905	0,9942	[9]
...	0,9957	0,9940	0,9951	0,9957	0,9947	0,9943	0,9946	0,9842	0,9956	[9]
...	0,9919	0,9917	0,9911	0,9917	0,9910	0,9916	0,9917	0,9870	0,9914	[9]
...	0,9934	0,9929	0,9929	0,9934	0,9924	0,9929	0,9930	0,9862	0,9935	[12]
...	0,9900	0,9904	0,9900	0,9893	0,9900	0,9904	0,9902	0,9845	0,9902	[16]
...	0,9950	0,9952	0,9945	0,9943	0,9952	0,9947	0,9942	0,9944	0,9950	[16]
...	0,9936	0,9933	0,9933	0,9934	0,9930	0,9934	0,9935	0,9860	0,9937	[12]

3.4 Valutazione della qualità in assenza di riferimento

Di seguito viene trattato un metodo di valutazione della qualità di immagini ricostruite con tecniche di demosaicking trattato nell articolo [25].

Nella quotidianità dell'elaborazioni di immagini, la maggior parte delle volte non ci è data la possibilità di avere l'immagine originale a piena risoluzione in tutte le componenti RGB ma solamente un'immagine acquisita attraverso un CFA, il che rende impossibile effettuare una valutazione della bontà dell algoritmo di demosaicking utilizzato.

Occorre quindi fare una distinzione tra i metodi di valutazione: "*with reference*" e "*no-reference*".

Il primo metodo "*with reference*", viene attuato quando si ha a disposizione l'immagine originale, senza che questa sia stata catturata da un CFA e quindi senza che questa soffra di artefatti o di fenomeni quali ZE; questo metodo è usato nella maggior parte degli articoli sul demosaicking e nelle simulazioni in laboratorio. Il metodo "*with reference*" consente oltretutto di avere una analisi oggettiva tra l'immagine elaborata e come l'immagine dovrebbe essere.

Il secondo metodo "*no-reference*", consente di effettuare valutazioni quando non si dispone dell'immagine originale, quello descritto nell'articolo [25] e da me implementato in Matlab, descrive un processo basato sulla tecnica di *Double Interpolation.*

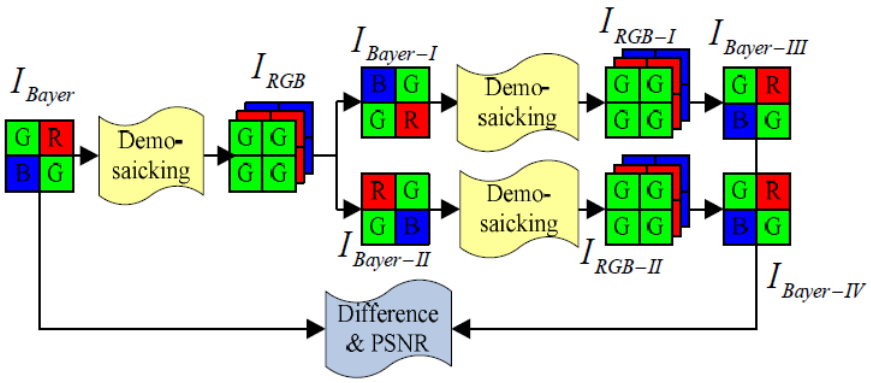


Figura 3.4.1: Digramma per il calcolo del DI-PSNR

Il meccanismo di doppia interpolazione funziona in questo modo:

1. L'immagine viene acquisita con un pattern Bayer classico (I_{bayer})
2. Viene elaborata con l'algoritmo di demosaicking scelto
3. La risultante è un immagine a tre canali RGB (I_{RGB})
4. Ad I_{RGB} viene applicato un pattern bayer modificato (BG,GR) generando $I_{Bayer-I}$
5. Viene riapplicato l'algoritmo di demosaicking scelto
6. La risultante è l'immagine RGB (I_{RGB-I})
7. Ad I_{RGB-I} viene applicato il pattern bayer classico generando $I_{Bayer-III}$
8. Si ripetono i punti 4,5,6,7 per un secondo patter Bayer modificato (RG,GB) generando $I_{Bayer-II}$, I_{RGB-II}, $I_{Bayer-IV}$
9. Viene calcolata la differenza assoluta tra $I_{Bayer-IV}$ o $I_{Bayer-III}$ e I_{bayer} , questa viene definita come *DI-difference map*
10. Viene calcolato il PSNR tra $I_{Bayer-IV}$ o $I_{Bayer-III}$ e I_{bayer} , questo viene definito come *DI-PSNR*

La *DI-difference map* da' una informazione locale della qualità dell'algoritmo di demosaicking, i punti più critici infatti appaino subito evidenti.

In figura 3.4.2 si osservano chiaramente i punti deboli di un algoritmo di demosaicking, questi sono stati amplificati di un fattore 5 per risaltare i pixel peggio ricostruiti, il tratto di palizzata i piedi del faro è uno dei punti più difficili che colpisce tantissimi algoritmi.

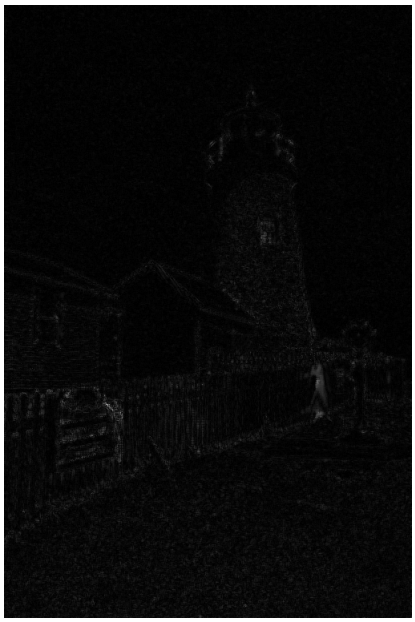


Figura 3.4.2: Esempio di *DI-difference map* (errori amplificati)

Il DI-PSNR viene calcolato come segue:

$$MSE = \frac{1}{W \times H} \sum \sum \left[I'\left(x, y\right) - I\left(x, y\right) \right]$$

$$PSNR = 20 \times \log \left(\frac{255}{\sqrt{MSE}} \right)$$

dove $I(x, y)$ rappresenta l'immagine originale e $I'(x, y)$ l'immagine ricostruita, W e H sono le dimensioni del frame considerato.

Se si è in posseso anche dell'immagine originale, è possibile calcolare il vero PSNR, in particolare il CPSNR:

$$CPSNR = 10 log_{10} \frac{255^2}{\frac{1}{3} \sum_{X=R,G,B} MSE(X)}$$

ovvero il *color peak signal-to-noise ratio*, dove il *Mean Square Error* è mediato per le tre componenti di colore RGB.

Implementazione

La maggior parte degli algoritmi di demosaicking viene implementato per accettare in ingresso dati derivanti da un CFA di bayer classico, quindi nella seconda e nella terza interpolazione descritte nello schema a blocchi in figura 3.4.1, si dovrebbe re-implemetare tutto l'algoritmo di demosaicking per riadattarlo ai due differenti schemi di CFA.

Per ovviare questa problematica, ho scelto di considerare la prima immagine di Bayer $I_{Bayer-I}$ come una I_{Bayer} privata della prima riga e $I_{Bayer-II}$ come una I_{Bayer} privata della prima colonna:

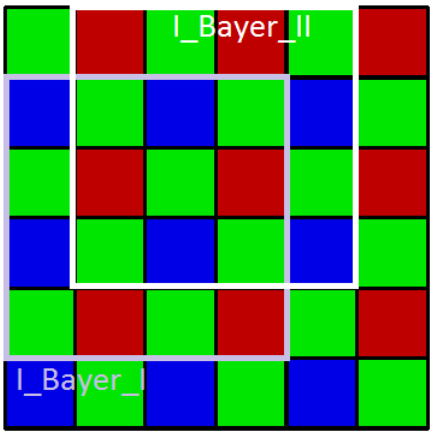


Figura 3.4.3: CFA adattato

Lo schema a blocchi modificato è quindi riportato in figura 3.4.4.

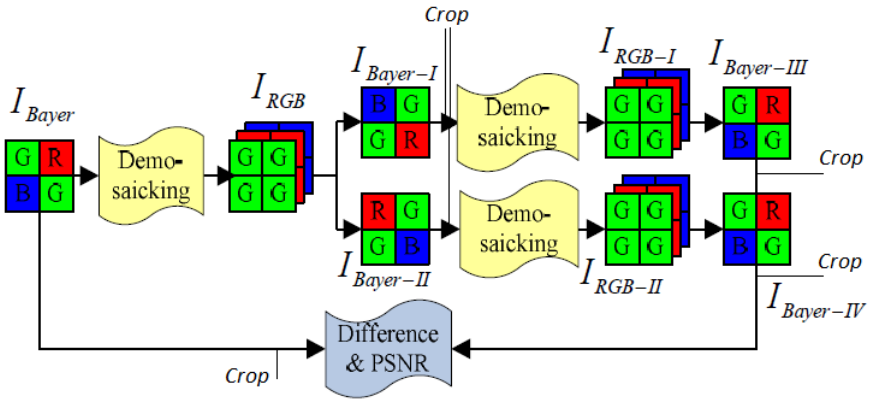


Figura 3.4.4: Schema a blocchi utlizzato

Le due immagini risulatanti,$I_{Bayer-III}$ e $I_{Bayer-IV}$, devono quindi essere ulteriormente tagliate per considerare l'area dei pixel comune ad entrambe: nel successivo passo risulta evidente che il frame di pixel utilizzabili ai fini del confronto tra i CPSNR e DIPSNR, è quello ottenuto privando I_{Bayer} delle opportune righe e colonne.

La scelta di sacrificare il bordi delle immagini per il confronto è giustificata dal fatto che la maggior parte degli algoritmi di demosaicking esistenti utilizza per l'interpolazione dei bordi delle tecniche diverse da quelle utilizzate nel centro che non permettono la stima corretta dei pixel sul bordo, tali pixel sono di scarsa attendibilità e quindi scartabili in un'analisi qualitativa.

Figura 3.4.5: Esempio tratto da Matlab

Risultati

E' stata eseguita la simulazione, sulle immagini *Lossless True Color* della Kodak [23], di tutti gli algoritmi in possesso, i risultati sono riportati nel grafico 3.4.6.

Si osserva che esiste una forte correlazione tra i valori di CPSNR calcolati tra l'immagine Kodak e la sua ricostruzione, ed i valori DI-PSNR derivanti dall'applicazione di questo procedimento.

La correlazione viene evidenziata dalla concetrazione di valori attorno alla bisettrice degli assi *x,y* , tanto più il valore calcolato si avvicina alla linea tratteggiata, tanto più il valore DI-PSNR è attendibile.

Nonostante la forte correlazione tra i dati nella simulazione, si evince che il processo di doppia interpolazione calcola con buona approsimazione il valore di PSNR, ma è incapace di stabilire con precisione l'algoritmo ottimale per una data immagine.

Per questo motivo, a questa tecnica, che ci fornisce un valore numerico della bontà di ricostruzione, è giusto affiancare una verifica visiva sia sulla *DI-difference map* che sull'immagine ricostruita.

Image\Algh	Bil.	[1]	[5]	[7]	[10]	[8]	[13]	[14]
1	26,19	28,72	31,81	35,11	37,70	36,36	36,87	28,61
2	33,11	33,41	37,54	38,90	38,75	39,50	40,17	34,88
3	34,37	36,18	39,24	41,35	40,94	41,95	42,21	35,71
4	33,61	33,30	37,67	38,62	39,18	39,47	39,87	34,89
5	26,69	27,23	32,79	35,46	37,10	37,10	37,42	29,09
6	27,69	30,12	33,25	37,60	38,33	38,75	39,24	30,19
7	33,46	35,79	39,01	40,33	41,12	41,17	41,67	34,59
8	23,61	26,47	29,85	33,78	35,56	34,79	35,39	27,00
9	32,46	35,03	38,39	41,11	41,51	42,05	42,66	34,47
10	32,34	34,82	38,05	40,49	41,46	41,56	42,06	34,36
11	29,26	31,51	34,56	37,56	39,05	38,57	39,27	31,32
12	32,91	35,31	38,78	41,66	42,37	42,46	42,98	35,56
13	23,92	26,28	28,43	31,42	34,50	32,91	33,30	25,57
14	29,27	31,18	34,28	35,34	35,49	35,73	36,78	30,98
15	31,54	32,74	36,13	37,75	38,41	38,57	38,99	33,44
16	31,38	33,76	36,80	41,46	41,22	42,68	43,01	33,56
17	31,95	34,20	36,99	39,40	41,24	40,55	40,92	33,56
18	27,88	30,06	32,40	34,48	36,81	35,88	36,26	29,67
19	28,00	30,74	34,58	38,20	39,92	39,31	39,96	31,14
20	31,36	33,52	36,20	38,93	39,18	39,74	40,13	33,28
21	28,57	31,00	33,72	36,62	38,65	37,57	37,96	30,66
22	30,52	32,76	35,23	36,50	37,63	37,04	37,73	32,37
23	35,26	37,22	40,47	41,69	41,44	42,19	42,63	36,96
24	26,68	28,99	30,87	33,19	34,33	33,95	34,26	28,65

Tabella 3.8: PSNR [dB] delle immagini kodak, algoritmi 0-8

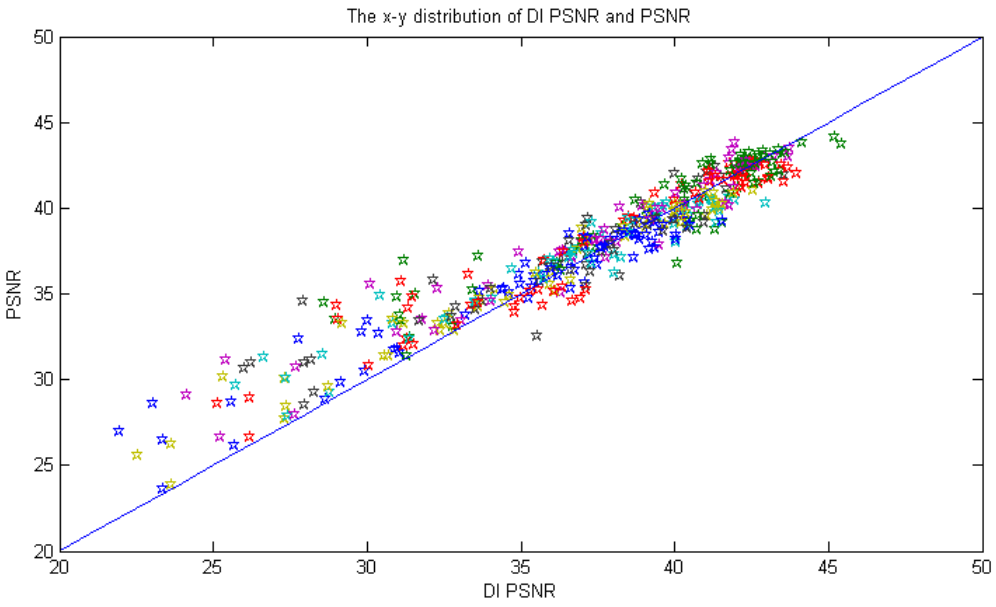


Figura 3.4.6: Risultati della simulazione

...	[15]	[9]	[16]	[17]	[18]	[19]	[20]	[24]	[11]	[12]
...	38,40	38,33	37,55	38,04	39,13	36,78	37,75	38,21	31,71	38,50
...	36,83	40,46	40,34	39,19	39,62	39,94	39,12	38,78	37,66	40,63
...	39,63	42,58	42,74	41,72	42,54	42,23	42,35	42,10	39,48	42,87
...	38,06	40,38	40,46	40,34	40,08	40,14	40,65	40,45	38,54	40,55
...	35,11	38,03	38,09	37,90	38,03	37,42	38,08	38,04	33,55	38,16
...	39,12	40,04	38,09	39,95	40,41	38,82	39,45	39,73	32,85	40,20
...	39,36	42,13	42,78	41,99	41,90	42,00	42,51	42,22	39,39	42,42
...	35,69	36,03	35,34	35,30	36,74	35,15	35,76	36,07	28,83	36,57
...	40,88	43,12	42,89	42,23	43,14	42,61	42,65	42,51	38,13	43,31
...	41,44	42,37	42,58	42,06	42,37	42,11	42,55	42,54	38,26	42,81
...	38,96	40,08	39,43	39,83	40,24	39,18	39,61	39,77	34,77	40,30
...	42,04	43,41	42,79	42,99	43,48	42,91	43,03	43,01	37,54	43,80
...	35,80	34,78	33,86	35,16	35,56	33,25	34,29	34,92	29,58	35,26
...	32,52	37,11	37,34	36,07	36,43	36,91	36,71	36,31	34,58	37,47
...	37,67	39,35	39,51	39,28	39,02	39,33	39,49	39,34	36,13	39,69
...	42,12	43,45	41,27	43,72	44,18	42,57	43,04	43,27	36,36	43,86
...	41,45	41,84	41,50	41,57	41,76	40,89	41,58	41,80	37,49	41,86
...	36,24	37,01	36,87	37,15	37,29	36,44	37,08	37,33	33,50	37,39
...	40,13	40,85	40,15	40,34	41,22	40,03	40,01	40,25	33,46	41,27
...	39,58	39,79	41,00	40,24	41,03	40,25	40,85	40,90	36,23	39,89
...	39,13	39,08	38,75	38,87	39,62	38,09	39,00	39,32	34,11	39,41
...	37,13	38,53	38,64	38,21	38,16	38,26	38,45	38,40	35,59	38,54
...	39,58	43,23	43,20	41,84	42,63	42,89	42,56	42,06	41,35	43,15
...	34,57	35,14	34,72	35,28	34,79	34,55	35,27	35,45	32,04	35,19

Tabella 3.9: PSNR [dB] delle immagini kodak,algoritmi 9-17

Image\Algh	Bil.	[1]	[5]	[7]	[10]	[8]	[13]	[14]
1	25,69	25,58	30,95	34,60	37,58	36,74	36,62	23,04
2	32,79	31,10	36,69	38,30	41,30	39,41	40,30	30,96
3	33,38	33,31	38,33	40,23	42,00	41,87	41,83	31,09
4	32,60	30,81	36,84	38,51	41,37	40,01	40,45	30,42
5	25,22	26,56	30,94	33,95	37,74	36,67	36,27	24,15
6	27,33	27,33	32,30	36,82	38,39	39,06	38,98	25,31
7	31,69	32,14	37,14	38,88	41,70	40,47	40,76	27,92
8	23,35	23,33	29,15	33,21	35,93	35,24	35,35	21,96
9	31,37	31,54	37,33	40,33	42,74	42,09	42,24	28,59
10	31,29	31,45	37,05	40,00	42,75	42,02	42,22	29,02
11	28,77	28,56	33,51	36,63	40,00	38,56	38,63	26,65
12	32,18	32,28	37,70	41,17	42,07	42,66	42,88	30,10
13	23,63	23,63	27,38	30,56	34,48	32,83	32,50	22,54
14	28,28	28,18	32,89	34,01	37,13	35,60	36,25	26,22
15	31,07	30,35	35,79	37,72	40,04	39,51	39,61	29,99
16	31,30	31,07	36,16	40,73	42,05	43,18	42,84	28,96
17	31,19	31,31	35,77	38,68	42,19	40,79	40,66	29,05
18	27,43	27,38	31,36	33,52	37,30	35,76	35,59	25,73
19	27,65	27,70	34,00	37,40	40,74	39,23	39,36	25,41
20	30,68	30,82	35,46	38,35	40,24	40,34	40,19	29,20
21	27,98	27,96	32,75	36,00	39,64	38,02	37,95	25,97
22	29,94	29,80	34,39	35,98	39,15	37,39	37,63	27,77
23	33,45	33,59	38,73	40,22	42,61	41,84	41,94	31,20
24	26,17	26,20	30,04	32,95	35,71	34,80	34,81	25,14

Tabella 3.10: DI-PSNR [dB] delle immagini kodak, algoritmi 0-8

...	[15]	[9]	[16]	[17]	[18]	[19]	[20]	[24]	[11]	[12]
...	38,71	37,06	36,44	39,28	39,37	35,16	37,09	37,90	30,86	36,57
...	40,08	40,33	40,31	41,56	40,48	39,06	40,50	40,74	37,31	40,95
...	40,20	41,78	42,41	43,05	42,53	41,12	42,28	42,32	38,50	42,45
...	40,03	40,80	41,21	42,94	41,30	39,62	41,77	41,96	37,61	41,17
...	36,32	37,00	37,20	39,48	38,11	34,92	37,39	37,92	31,80	37,57
...	39,67	39,15	37,18	41,45	41,32	37,01	39,15	40,06	32,37	39,99
...	39,93	41,02	42,15	43,55	41,48	39,98	42,30	42,34	37,18	41,33
...	37,08	35,45	34,41	36,63	37,48	33,64	35,45	36,23	28,62	36,22
...	41,82	42,13	42,26	43,41	43,51	41,17	42,33	42,63	36,96	42,88
...	42,31	41,98	42,76	43,95	43,68	41,09	42,97	43,46	37,05	42,58
...	40,39	39,33	38,48	41,43	40,77	37,31	39,42	40,17	34,05	39,71
...	41,35	41,86	42,31	43,68	43,72	41,77	42,93	43,21	36,37	41,97
...	36,64	33,67	32,67	36,11	35,95	31,19	33,60	34,63	28,75	34,15
...	35,51	36,44	37,18	38,23	36,63	35,72	37,11	37,24	33,73	37,61
...	39,33	39,29	40,01	41,52	40,48	38,85	40,17	40,41	34,91	39,19
...	43,22	43,37	40,37	45,43	45,18	40,91	42,72	43,58	36,31	44,15
...	42,57	41,31	41,11	43,56	43,07	39,36	41,58	42,37	36,38	41,93
...	38,02	35,92	36,08	38,27	38,03	34,72	36,58	37,22	32,53	36,73
...	41,79	40,02	38,93	41,65	41,90	38,20	39,56	40,31	33,05	40,95
...	41,06	39,98	41,01	41,54	42,55	39,15	41,27	41,90	35,96	41,23
...	40,95	38,72	38,10	40,43	40,94	36,56	39,13	40,01	33,52	39,42
...	38,69	38,19	38,33	40,02	38,85	37,18	38,74	39,10	34,98	38,68
...	40,75	42,43	42,72	43,15	42,22	41,16	42,65	42,51	39,67	43,11
...	36,64	35,31	34,97	37,11	36,93	33,59	35,59	36,36	31,51	36,05

Tabella 3.11: DI-PSNR [dB] delle immagini kodak,algoritmi 9-17

3.5 Analisi della qualità monocromatica

Un altro metodo pensato per la valutazione di un algoritmo di demosaicking di tipo *No-Reference* è basato sulle immagini monocromatiche.

Si pensi di dover stimare la qualità di un metodo di ricostruzione avendo a disposizione un dispositivo quale una fotocamera o una videocamera equipaggiato con un *color filter array* classico di tipo Bayer, se in tale dispositivo fosse accessibile e rimovibile il CFA, si avrebbero a disposizioni immagini *full* di tipo monocromatico.

Da questo tipo di acquisizione, applicando un normale algoritmo di demosaicizzazione per immagini a colori, dovrebbe ugualmente risultare un'immagine monocromatica, eventuali presenze di colore all'interno dell'immagine ricostruita sono quindi da identificarsi come errori di ricostruzione.

L'idea alla base di questo metodo di analisi è quella appena descritta, nella fase implementativa, non avendo a disposizione immagini acquisite senza il CFA, si è scelto di partire dalle immagini del database [23] essendo queste immagini *full RGB* non compresse e accessibili a tutti, il primo passo è stato quindi la conversione da RGB a *Gray-scale*,

per far ciò si può calcolare la media delle tre componenti dell'immagine per ogni pixel e successivamente creare un immagine monocromatica in formato RGB ricopiando su ciascun canale lo stesso valore di pixel.

Successivamente sono stati applicati gli stessi algoritmi di demosaicking descritti nei capitoli precedenti che vengono comunemente sfruttati nella ricostruzione di immagini Bayerizzate.

Infine viene valutato l'errore di ricostruzione commesso da ogni singola tecnica, l'errore è stato calcolato in due modi differenti, il primo modo prevede l'applicazione di distanza euclidea, una sorta di MSE tra un immagine RGB e una in scala di grigi:

$$Err = \frac{1}{m \times n} \sum_i \sum_j (\hat{R}(i,j) - I_{gray}(i,j))^2 + (\hat{G}(i,j) - I_{gray}(i,j))^2 + (\hat{B}(i,j) - I_{gray}(i,j))^2$$

dove $\hat{R}(i,j), \hat{G}(i,j), \hat{B}(i,j)$ costituiscono le tre componenti dell'immagine ricostruita, $I_{gray}(i,j)$ l'immagine in scala di grigi e $m \times n$ in numero di pixel di cui sono composte le immagini.

E si può quindi definire di conseguenza il PSNR:

$$PSNR = 20 \times \log\left(\frac{255}{\sqrt{Err}}\right)$$

Un altro modo per stimare la qualità degli algoritmi è quello di convertire le due immagini a confronto in formato LAB e quindi calcolare il $\triangle E^*_{ab}$ come segue:

$$\triangle E^*_{ab} = \sqrt{(\triangle L^*)^2 + (\triangle a^*)^2 + (\triangle b^*)^2}$$

dove si definiscono i vari termini in questo modo:

$$\triangle L^* = \hat{L}^* - L^*_{gray}$$

$$\triangle a^* = \hat{a}^* - a^*_{gray}$$

$$\triangle b^* = \hat{b}^* - b^*_{gray}$$

In figura 3.5.1 è riportato un esempio eseguito sull'immagine del faro, nella figura in alto si osserva la conversione eseguita in scala di grigi (ovvero l'equivalante di avere tolto il CFA di Bayer), nella figura in basso a sinistra si osserva come l'algoritmo di demosaicking bilineare abbia elaborato i dati (le zone colorate sono quelle dove sono stati commessi gli errori di ricostruzione) nell'ultima immagine ho enfatizzato tali errori come la differenza tra l'immagine in scala di grigi e quella ricostruita (come ci si poteva

aspettare gli errori nascono lungo i contorni e nelle zone ad alta frequenza concentrandosi soprattutto nella palizzata).

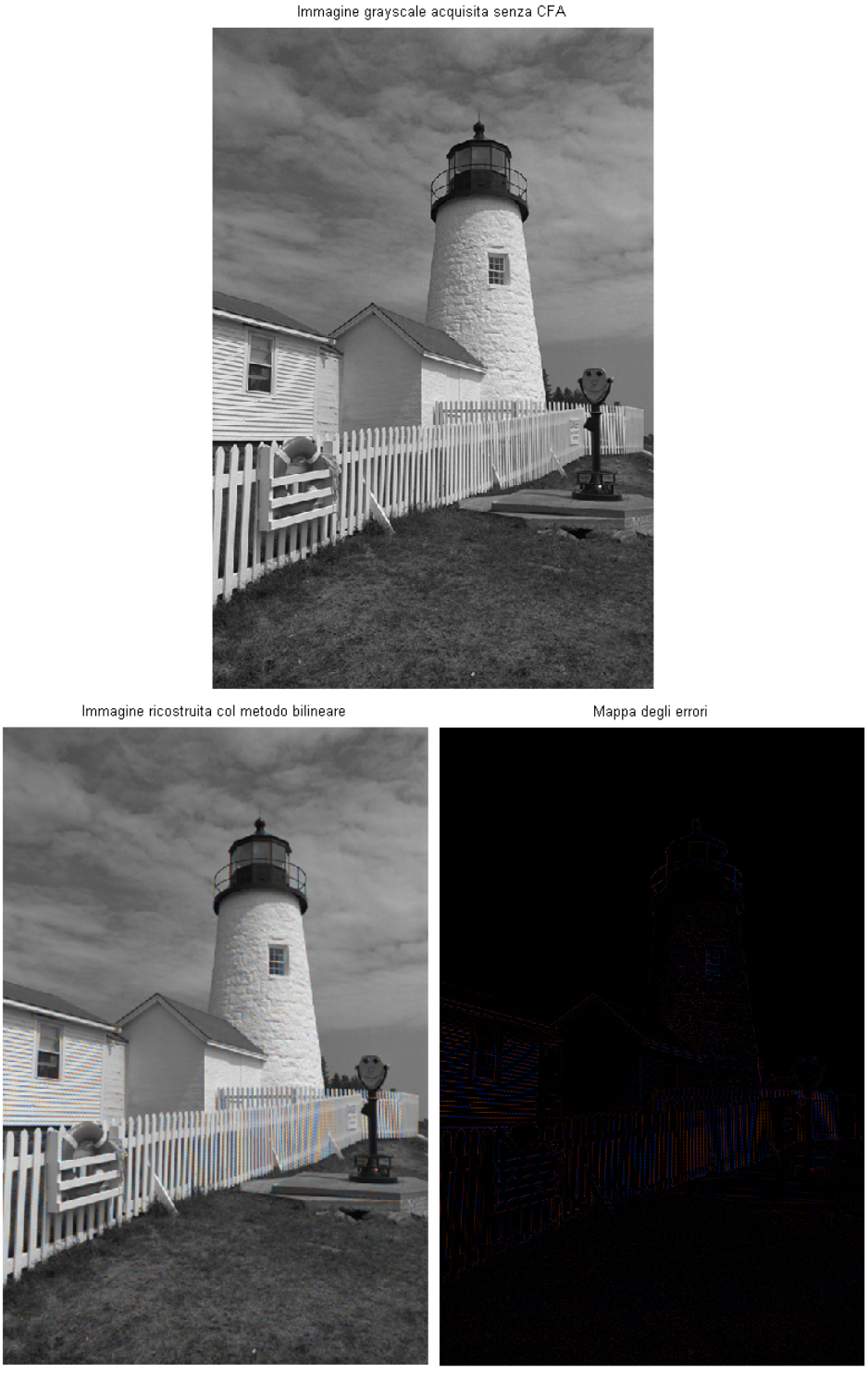


Figura 3.5.1: Esempio del metodo di analisi

E' stata effettuata una simulazione sulle prime 24 immagini del database kodak [23].

Image\Algh	Bil.	[1]	[5]	[7]	[10]	[8]	[13]	[14]
1	21,21	23,63	26,35	31,08	33,14	31,96	32,91	23,92
2	27,39	28,57	31,02	37,99	35,88	39,08	39,66	29,67
3	28,22	29,14	31,62	40,78	36,27	43,04	42,63	30,35
4	27,95	29,31	31,36	38,16	36,99	36,79	40,15	30,09
5	21,66	24,02	27,23	33,24	34,38	36,07	35,86	24,28
6	22,54	24,65	27,49	34,25	33,49	35,77	36,20	25,28
7	27,71	29,24	31,73	39,55	37,33	41,52	41,51	29,56
8	18,78	21,52	24,80	30,32	32,02	31,38	32,06	22,30
9	26,96	28,77	31,33	38,72	37,63	37,40	40,46	29,52
10	26,90	28,84	31,33	38,60	37,86	37,74	40,73	29,54
11	24,12	26,13	28,74	34,48	35,36	35,77	36,35	26,44
12	27,44	29,33	31,87	39,65	38,90	41,12	41,43	30,62
13	19,07	21,48	23,45	27,54	30,65	29,14	29,52	20,86
14	24,00	25,69	28,56	34,56	33,83	36,21	36,60	26,06
15	26,02	27,48	30,01	36,93	35,10	38,63	38,83	28,47
16	25,65	26,92	29,78	38,32	34,23	40,06	40,08	28,15
17	27,24	29,79	32,59	36,89	39,45	37,61	38,89	28,93
18	23,17	25,72	27,76	31,53	34,76	33,54	33,71	25,07
19	23,37	26,17	30,00	34,89	37,75	36,24	36,75	26,38
20	25,23	26,32	28,24	36,44	33,57	37,83	38,15	28,00
21	23,49	25,70	28,17	33,01	34,55	34,14	34,55	25,87
22	25,53	27,60	30,10	35,04	36,39	36,43	36,84	27,80
23	29,29	30,50	32,91	41,09	38,12	43,12	42,85	31,83
24	22,00	24,54	26,54	31,15	33,42	33,51	33,47	24,21

Tabella 3.12: PSNR [dB] delle immagini kodak in scala di grigio, algoritmi 0-7

...	[15]	[9]	[16]	[17]	[18]	[19]	[20]	[24]	[11]	Best
...	36,55	33,49	33,50	34,76	36,45	32,63	34,25	35,14	26,27	[15]
...	39,68	36,39	40,12	41,73	43,02	39,21	40,52	41,39	30,91	[18]
...	40,58	37,06	42,45	44,88	46,73	41,68	43,59	44,56	31,56	[18]
...	40,93	37,81	40,63	42,70	43,44	39,54	40,66	41,44	31,74	[18]
...	38,37	35,42	36,45	39,19	39,68	34,64	37,12	38,36	27,87	[18]
...	37,12	34,86	34,45	38,17	39,30	35,37	36,75	37,69	27,18	[18]
...	41,87	37,98	42,69	43,94	45,31	40,53	42,97	43,94	31,83	[18]
...	34,83	32,23	31,82	32,47	35,44	31,46	32,88	33,73	23,84	[18]
...	41,57	38,25	40,67	42,20	43,99	39,85	40,67	41,37	31,16	[18]
...	41,44	38,48	41,36	42,61	44,23	40,02	41,01	41,79	31,35	[18]
...	39,38	36,02	36,46	38,44	39,57	35,77	37,30	38,27	28,92	[18]
...	42,68	39,74	40,43	42,36	44,86	40,60	42,00	42,83	31,11	[18]
...	34,45	31,07	29,92	32,06	33,07	29,14	30,61	31,61	24,58	[15]
...	37,77	34,51	37,32	39,07	40,15	35,81	37,62	38,67	28,76	[18]
...	38,34	35,71	38,84	41,17	42,28	38,53	39,99	40,86	29,69	[18]
...	37,48	35,33	37,49	42,10	43,56	39,15	40,37	41,25	29,21	[18]
...	40,72	39,93	39,30	41,20	42,06	37,90	39,77	40,78	32,88	[18]
...	37,96	34,52	34,02	36,13	37,09	33,02	34,76	35,80	28,81	[15]
...	39,10	38,24	36,86	39,04	40,40	36,35	37,34	38,17	28,86	[18]
...	38,87	33,80	38,97	39,80	42,19	37,93	39,74	40,75	28,29	[18]
...	38,19	34,79	35,32	36,46	38,08	34,50	36,20	37,11	28,49	[15]
...	38,95	36,91	37,50	38,98	40,19	36,55	38,02	38,91	30,44	[18]
...	42,97	38,84	43,27	44,99	46,59	42,10	44,46	45,32	33,23	[18]
...	36,00	34,20	32,91	35,81	36,33	32,45	34,22	35,33	27,84	[18]

Tabella 3.13: PSNR [dB] delle immagini kodak in scala di grigio, algoritmi 8-16

Image\Algh	Bil.	[1]	[5]	[7]	[10]	[8]	[13]	[14]
1	5,041	3,756	2,980	1,092	1,289	1,400	1,425	4,355
2	2,447	1,854	1,537	0,572	0,752	0,726	0,760	2,176
3	1,713	1,335	1,086	0,343	0,524	0,419	0,470	1,568
4	2,238	1,716	1,451	0,535	0,692	0,766	0,689	2,009
5	5,478	3,962	3,217	1,085	1,298	1,282	1,433	5,250
6	3,688	2,767	2,139	0,646	0,977	0,780	0,849	3,081
7	2,038	1,581	1,228	0,421	0,562	0,523	0,570	2,316
8	5,873	4,322	3,193	1,100	1,344	1,414	1,454	4,982
9	1,938	1,524	1,198	0,427	0,568	0,637	0,548	1,834
10	1,946	1,504	1,207	0,444	0,568	0,644	0,565	1,788
11	3,596	2,698	2,168	0,762	0,977	0,968	1,018	3,285
12	1,633	1,274	1,000	0,321	0,462	0,395	0,423	1,399
13	6,773	4,908	4,461	1,805	2,011	2,266	2,368	6,070
14	3,924	2,917	2,383	0,847	1,104	1,050	1,127	3,664
15	2,631	1,979	1,705	0,630	0,816	0,772	0,832	2,460
16	2,932	2,213	1,724	0,516	0,829	0,615	0,684	2,475
17	2,655	1,998	1,679	0,743	0,798	0,930	0,966	2,693
18	4,520	3,313	2,925	1,271	1,315	1,559	1,636	4,225
19	3,266	2,444	1,879	0,759	0,781	0,946	0,978	2,929
20	2,254	1,739	1,489	0,529	0,706	0,678	0,716	2,100
21	3,446	2,582	2,182	0,850	1,008	1,061	1,124	3,129
22	2,760	2,084	1,764	0,697	0,830	0,859	0,909	2,452
23	1,412	1,127	0,951	0,359	0,468	0,435	0,480	1,347
24	3,461	2,493	2,053	0,799	0,873	0,978	1,043	3,034

Tabella 3.14: (LAB) delle immagini kodak in scala di grigio, algoritmi 0-7

...	[15]	[9]	[16]	[17]	[18]	[19]	[20]	[24]	[11]	Best
...	0,945	1,168	1,312	1,125	0,909	1,586	1,229	1,120	2,980	[18]
...	0,627	0,686	0,690	0,608	0,448	0,830	0,665	0,611	1,537	[18]
...	0,408	0,438	0,432	0,369	0,239	0,525	0,399	0,359	1,055	[18]
...	0,583	0,574	0,624	0,520	0,436	0,793	0,614	0,557	1,330	[18]
...	0,945	0,956	1,240	0,932	0,857	1,753	1,125	1,003	3,095	[18]
...	0,716	0,751	0,961	0,667	0,559	0,999	0,786	0,713	2,201	[18]
...	0,438	0,461	0,491	0,426	0,330	0,650	0,439	0,400	1,228	[18]
...	1,026	1,222	1,398	1,281	0,948	1,636	1,305	1,201	3,545	[18]
...	0,479	0,508	0,520	0,453	0,358	0,619	0,525	0,483	1,234	[18]
...	0,476	0,507	0,508	0,454	0,366	0,641	0,520	0,476	1,187	[18]
...	0,696	0,846	0,967	0,818	0,626	1,154	0,896	0,820	2,237	[18]
...	0,348	0,403	0,413	0,382	0,250	0,473	0,377	0,345	1,036	[18]
...	1,336	1,769	2,143	1,789	1,473	2,624	2,024	1,834	4,064	[15]
...	0,825	0,894	1,015	0,817	0,677	1,298	0,944	0,852	2,317	[18]
...	0,685	0,705	0,750	0,625	0,472	0,905	0,703	0,641	1,649	[18]
...	0,685	0,651	0,813	0,540	0,408	0,790	0,645	0,586	1,821	[18]
...	0,676	0,693	0,842	0,734	0,610	1,098	0,776	0,703	1,654	[18]
...	0,945	1,162	1,460	1,202	1,043	1,865	1,332	1,205	2,644	[15]
...	0,664	0,710	0,890	0,748	0,610	1,088	0,838	0,763	1,945	[18]
...	0,556	0,646	0,648	0,588	0,412	0,780	0,586	0,530	1,465	[18]
...	0,732	0,912	1,034	0,864	0,708	1,227	0,923	0,841	2,096	[18]
...	0,693	0,738	0,831	0,683	0,572	0,988	0,747	0,681	1,645	[18]
...	0,367	0,420	0,426	0,365	0,257	0,509	0,376	0,343	0,882	[18]
...	0,650	0,743	0,943	0,803	0,628	1,202	0,870	0,782	1,948	[18]

Tabella 3.15: (LAB) delle immagini kodak in scala di grigio,algoritmi 8-16

Capitolo 4

Demosaicking e campionamento

In questo capitolo viene affrontata una problematica abbastanza comune in questo ambito: il ridimensionamento di un immagine acquisita con CFA di tipo Bayer con un numero di pixel $W_1 \times H_1$, per poterla visualizzare su di un display con formato standard di dimensione $W_2 \times H_2$.

Il problema che mi è stato posto è ancora più specifico, si vuole passare da un immagine composta da 2880x1620 pixel ad una di dimensioni 1920x1080x3 pixel (formato *Full Hd*). Si nota subito che, avendo acquisito un'immagine con un *color filter array* e volendola ricostruire a colori, più che un *downscaling* si effettua in realtà il processo inverso.

Oltretutto si vuole realizzare un algoritmo che esegua questa operazione per immagini acquisite in *high speed mode* ossia aventi un *frame rate* variabile tra 60 e 120 fps; lo scopo ultimo di questa operazione è permettere la visione di anteprime "al volo" di riprese cinematografiche su di un display di risoluzione inferiore a quella finale: nasce quindi la necessità che l'algoritmo sia veloce e poco complesso.

4.1 Primo metodo

L'approccio usuale in figura 4.1.1 prevede che, data un'immagine Bayerizzata, venga ricostruita un'immagine delle stesse dimensioni a tre canali con un dato algoritmo di demosaicking e successivamente venga applicato il campionamento opportuno. Ovviamente tale approccio sarebbe il più efficace dal punto di vista qualitativo, tuttavia si effettuerebbero operazioni su matrici complete di pixel per poi scartare determinate zone arrivando quindi alla dimensione richiesta, si avrebbe così un occupazione di memoria maggiore ed anche dei tempi di elaborazione maggiori.

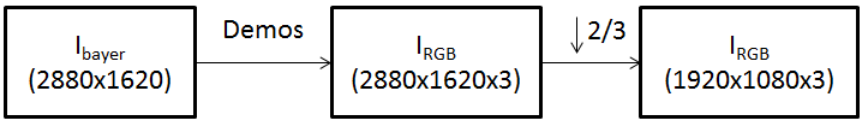


Figura 4.1.1: Schema del processo

Dalle specifiche del problema si nota subito che il campionamento richiesto è di tipo fratto: in figura 4.1.2 l'immagine acquisita con CFA di tipo Bayer viene campionata con fattore $\frac{2}{3}$, producendo così un ulteriore immagine nel dominio di Bayer, quindi viene applicato l'algoritmo di demosaicking scelto producendo un'immagine RGB.

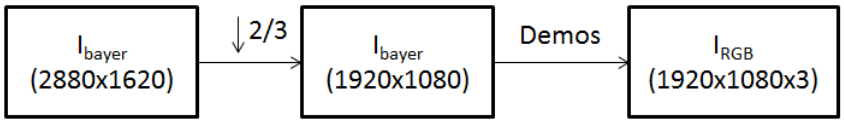


Figura 4.1.2: Schema del processo: fattore di campionamento fratto

Un campionamento di tipo fratto di fattore $\frac{2}{3}$ viene scomposto nell'interpolazione di fattore 2 seguita da una decimazione di fattore 3 entrambe eseguite nel "dominio di Bayer".

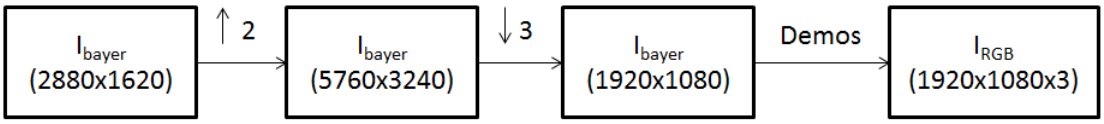


Figura 4.1.3: Schema del processo: interpolazione e decimazione

In questa sezione viene seguito lo schema a blocchi descritto in figura 4.1.3, ora vediamo più nel dettaglio il filtro interpolatore ed il filtro decimatore.

4.1.1 Filtro interpolatore

Il filtro interpolatore prende come ingresso dati di dimensione 2880x1920 e ne raddoppia la dimensione, il tutto però nel dominio di Bayer, ovvero nel quale la posizione di ciascun pixel è significativa nella ricostruzione dell'immagine finale a colori.

Si nota subito che il CFA di Bayer è una ripetizione periodica del blocco di pixel di dimensioni 2x2 strutturato dove si susseguono verde, rosso nella prima riga e blu, verde nella seconda riga.

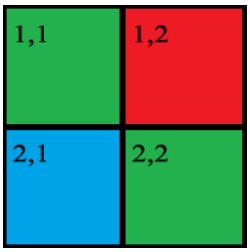


Figura 4.1.4: Blocco di 2x2 pixel di Bayer

Per mantenere questa successione invariata anche nell'immagine di dimensione doppia interpolata, eseguo un'interpolazione basata su blocchi di pixel, per ogni blocco di pixel strutturato come in figura 4.1.4, verrà prodotto un blocco di 4x4 pixel in uscita.

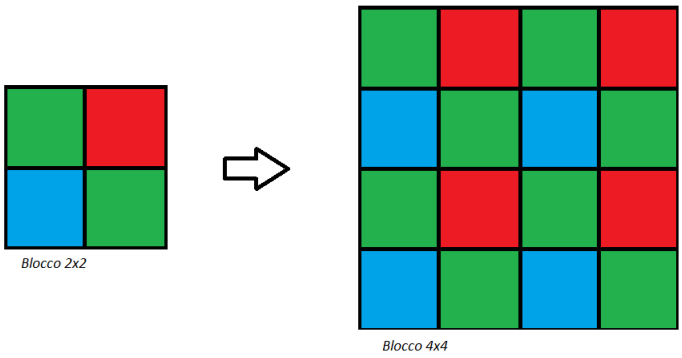


Figura 4.1.5: Interpolazione su blocchi di pixel

Fatte queste considerazioni, l'interpolazione può venire eseguita con diversi metodi, il più semplice ma sicuramente meno efficace prevede la ripetizione del blocco 2x2 nel blocco 4x4, nella pratica è una operazione di copia ed incolla del blocco 2x2.

Un altro metodo è basato su un approccio "bilineare", ogni pixel del blocchetto 2x2 viene ricopiato nella posizione opportuna a seconda del suo colore nel blocchetto 4x4, successivamente vengono ricostruiti i pixel mancanti dei blocchi 4x4 mediando sui valori dello stesso colore più vicini.

4.1.2 Il filtro decimatore

Il filtro decimatore prende in ingresso dati di dimensione 5760x3240 e produce in uscita dati di dimensione 1920x1080 sempre restando nel dominio di Bayer.

Ricordando come già detto che complessivamente l'algoritmo deve esser tanto veloce da permettere una visualizzazione rapida delle immagini, sono stati implementati due metodi di decimazione. Entrambi partono dall'analisi di blocchi di pixel di dimensione 6x6 per arrivare a blocchi di dimensione 2x2.

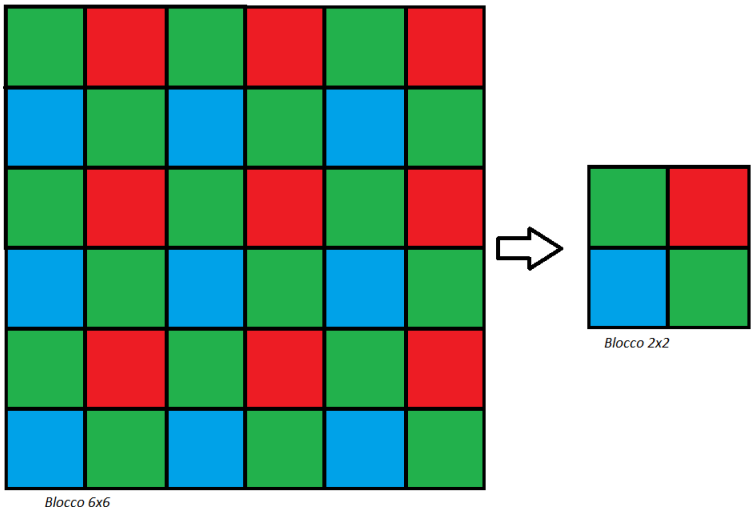


Figura 4.1.6: Decimazione di blocchi di pixel

Il primo metodo esegue una semplice media sui componenti presenti, il rosso così come il blu del blocco in uscita è calcolato come media dei nove componenti del blocco 6x6, per quanto riguarda i due verdi, si divida il blocco 6x6 in due triangoli con una diagonale passante per il vertice superiore destro sino a quello inferiore sinistro, di ciascuno dei triangoli viene quindi calcolata la media dei nove pixel verdi presenti in ciascuno di essi.

Un secondo metodo è sviluppato sempre secondo tale meccanismo, pensando però ciascun componente a seconda della posizione all'interno del blocco 6x6.

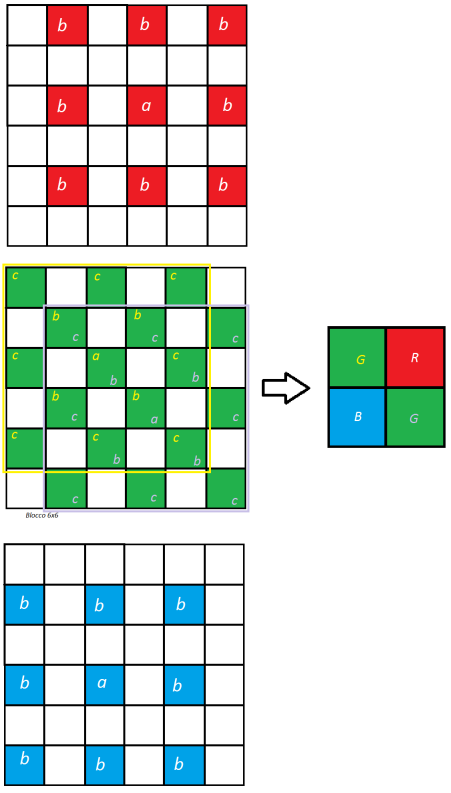


Figura 4.1.7: Decimazione con media pesata

Prendendo come riferimento la figura 4.1.7 ho applicato i seguenti pesi:

$$\begin{cases} a = \frac{1}{2} \\ b = \frac{1}{16} \end{cases} \quad rossi \qquad \begin{cases} a = \frac{1}{4} \\ b = \frac{1}{16} \\ c = \frac{1}{16} \end{cases} \quad verdi \qquad \begin{cases} a = \frac{1}{2} \\ b = \frac{1}{16} \end{cases} \quad blu$$

Ovviamente i due diversi verdi sono stati calcolati seguendo le due distinte zone di pixel.

A seguito di questa decimazione si trova un immagine di Bayer delle dimensioni cercate, ovvero di 1920x1080, ora non rimane altro che applicare uno dei tanti algoritmi di demosaicking noti descritti nei capitoli precedenti.

4.2 Secondo metodo

Alcune delle operazioni descritte nel primo metodo sono talvolta superflue o comunque aggirabili, ad esempio, aumentare le dimensioni delle immagini per poi comunque doverle ridurre sembra senza dubbio uno spreco di operazioni e di occupazione di memoria. Per quanto concerne il demosaicking, lavorare su una immagine di Bayer di dimensioni 1920x1080 sicuramente produrrà risultati meno soddisfacenti che da una di dimensioni 2880x1920, nasce quindi il desiderio di progettare un algoritmo che contemporaneamente riduca la dimensione e ricostruisca l'immagine a colori. Ovviamente tale algoritmo deve risultare semplice e veloce, quindi date le specifiche del problema vengono tralasciati algoritmi troppo pesanti quali gli adattativi.

In questo metodo si cerca di non eseguire il ridimensionemento dell'immagine di Bayer nei due passaggi descritti prima, l'idea di base è che dovendo interpolare con un fattore 2 e poi decimare con un fattore 3 in teoria è possibile sin da subito capire quali campioni restino inalterati, ad esempio in figura 4.2.1 si può osservare il caso mono dimensionale.

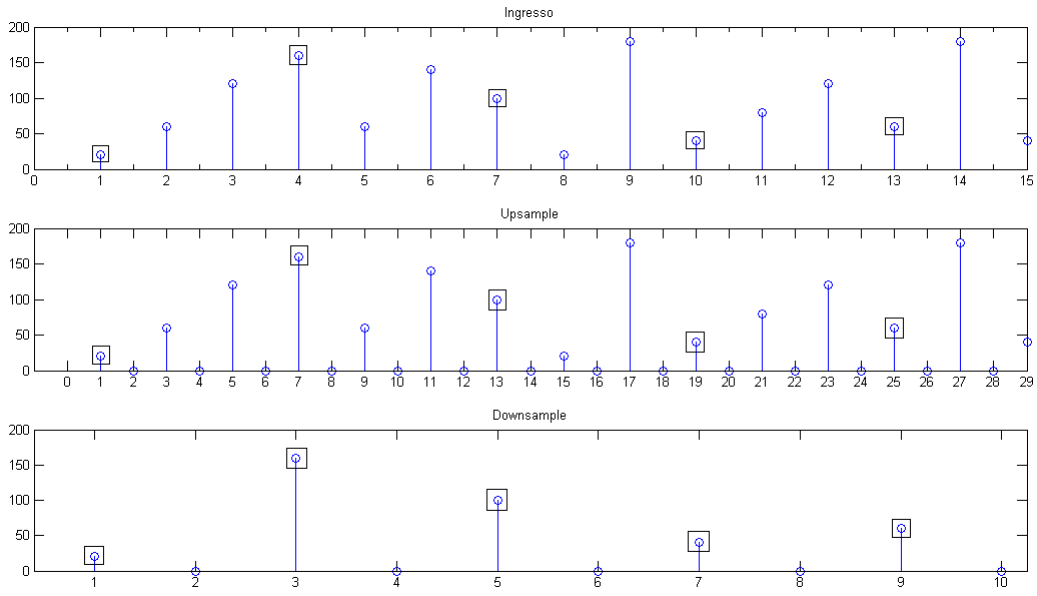


Figura 4.2.1: UpSampling & DownSampling

Nel primo grafico abbiamo i campioni d'ingresso, vengono quindi interpolati con fattore 2 (in questo esempio con l'aggiunta degli zeri per semplicità visiva) e successivamente decimati con fattore 3. Ci si accorge subito che alcuni di questi campioni, sono sin da subito individuabili nella serie di dati in ingresso con gli opportuni accorgimenti, questi campioni utili sono stati evidenziati con dei rettangolini.

Nel caso bidimensionale è possibile applicare una tecnica simile, ovvero ricercare nell'immagine di Bayer originale i valori utili e ricopiarli nell'immagine finale RGB delle dimensioni volute.

Per fare ciò si consideri la figura 4.2.2, la periodicità dei campioni utili sui dati in ingresso è di 6, che corrisponde ad una periodicità sui dati in uscita di 4, i restanti elementi vengono calcolati con semplici interpolazioni tra i campioni più vicini al pixel da ricostruire.

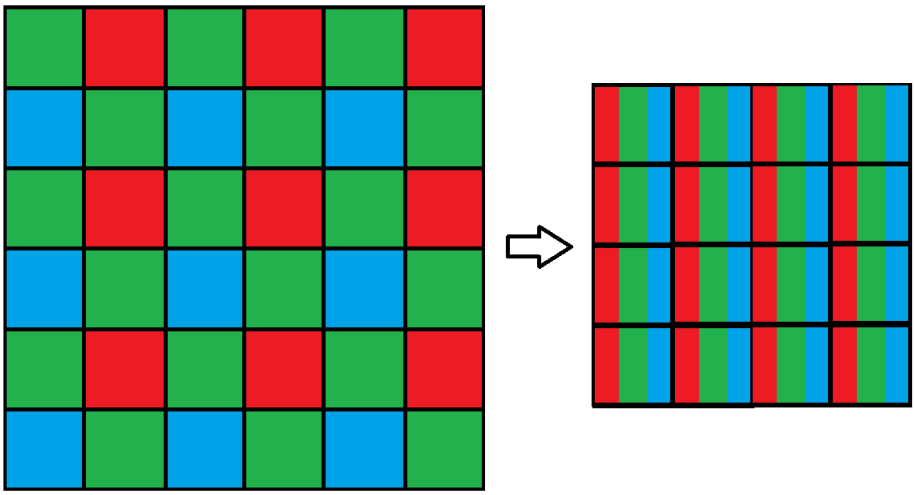


Figura 4.2.2: Upsampling & Downsampling su immagine di Bayer

Tale metodo non risulta tuttavia soddisfacente, le motivazioni si trovano nella sezione successiva.

4.3 Terzo metodo

Un modo diverso di pensare lo scalamento di un'immagine è considerare un'area fissata, e dentro di questa fare variare il numero di pixel presenti.

Per capire meglio questo approccio prendiamo ad esempio un'area di pixel significativa dell'immagine originale acquisita con CFA di Bayer, e sovrapponiamola all'immagine ricostruita e adeguatamente ridimensionata. Il risultato si può osservare in figura 4.3.1 dove è stato preso un blocco 9x9 dell'immagine di Bayer e uno di dimensioni scalate di $\frac{2}{3}$ ovvero 6x6.

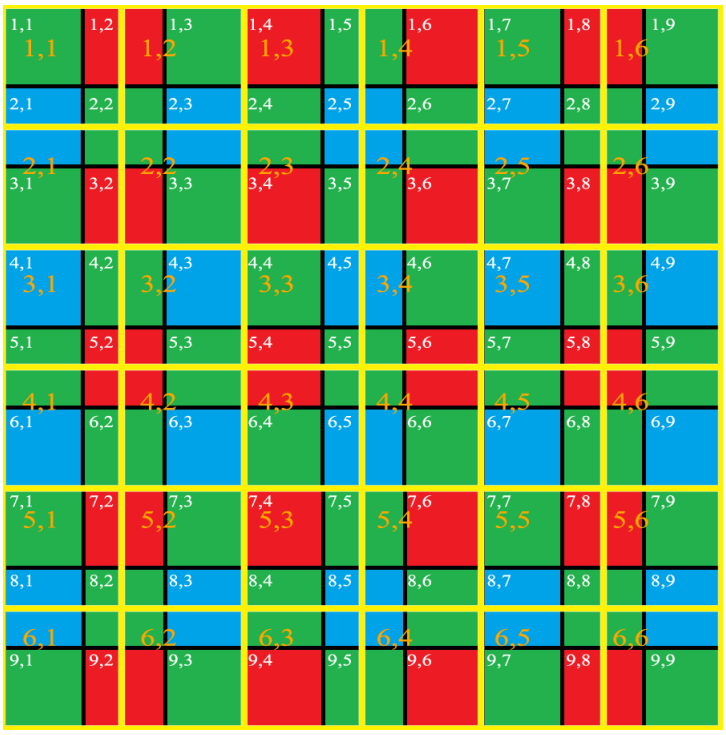


Figura 4.3.1: Joint demosaicking & downscaling

Con questa concezione un singolo pixel dell'immagine originaria non può mai apparire tale e quale nell'immagine ricostruita, sia dal punto di vista del suo valore, sia dal punto di vista della posizione. Si può osservare che ogni pixel costituente l'immagine finale è formato da una combinazione di 4 pixel appartenti all'immagine iniziale.

In particolare è possibile suddividere ciascun pixel ricostruito nella combinazioni delle quattro aree come in figura 4.3.2, ciascuna area ha un peso diverso a seconda della propria dimensione:

$$\begin{cases} A = \frac{4}{9} \\ B = \frac{1}{9} \\ C = \frac{2}{9} \end{cases}$$

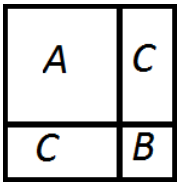


Figura 4.3.2: Suddivisione del pixel ricostruito

Quindi se si fossero già ricostruiti tutti e tre i canali per l'immagine delle dimensioni originarie scegliendo i quattro pixel e moltiplicandoli per i pesi si otterrebbero immediatamente tutti i pixel dell'immagine di risoluzione inferiore.

Volendo invece effettuare un demosaicking e downscaling contemporaneo, si possono costruire delle "maschere" sempre partendo dalle considerazioni fatte in precedenza. Consideriamo ad esempio con l'ausilio di figura 4.3.3 la ricostruzione del canale verde.

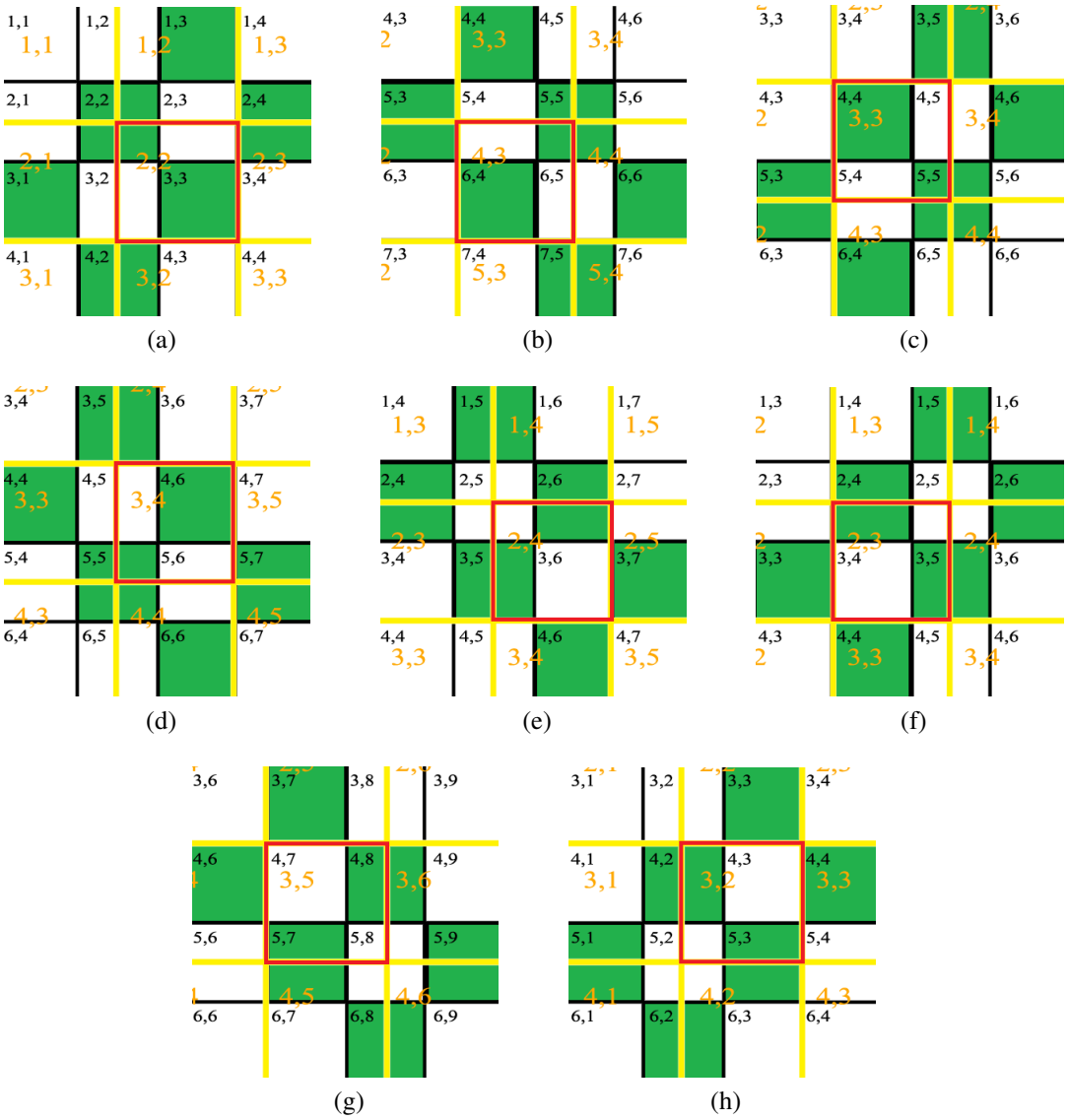


Figura 4.3.3: Maschere per il Green channel

In questo caso si usa lo stesso meccanismo di area pesata, prendiamo ad esempio la maschera (a), il pixel da ricostruire è segnalato con un contorno rosso, gli indici di colore nero appartengono all'immagine di dimensioni originali mentre gli indici di colore giallo all'immagine di dimensioni scalate, subito si nota (ricordando la figura 4.3.2) che sono presenti i componenti che danno i contributi A e B ma non quelli dei C, questi vengono calcolati dai pixel adiacenti:

$$\hat{G}(2,2) = A \cdot G(3,3) + B \cdot G(2,2) + C \cdot \left[\frac{G(3,1) + G(4,2)}{2}\right] + C \cdot \left[\frac{G(1,3) + G(2,4)}{2}\right]$$

dove è stato indicato con G il pixel verde appartente all'immagine originale Bayerizzata, con $\hat{G}$ il pixel verde ricostruito e con $\{A, B, C\}$ i pesi assegnati in precedenza. Lo stesso calcolo si applica alle maschere (b),(c),(d):

$$\hat{G}(4,3) = A \cdot G(6,4) + B \cdot G(5,5) + C \cdot \left[\frac{G(5,3)+G(4,4)}{2}\right] + C \cdot \left[\frac{G(7,5)+G(6,6)}{2}\right]$$

$$\hat{G}(3,3) = A \cdot G(4,4) + B \cdot G(5,5) + C \cdot \left[\frac{G(5,3)+G(6,4)}{2}\right] + C \cdot \left[\frac{G(3,5)+G(4,6)}{2}\right]$$

$$\hat{G}(3,4) = A \cdot G(4,6) + B \cdot G(5,5) + C \cdot \left[\frac{G(3,5)+G(4,4)}{2}\right] + C \cdot \left[\frac{G(5,7)+G(6,6)}{2}\right]$$

Per quanto riguarda le restanti maschere per il canale verde, le componenti mancanti sono quelle che hanno contributo A e B:

$$\hat{G}(2,4) = A \cdot \left[\frac{G(4,6)+G(3,7)}{2}\right] + B \cdot \left[\frac{G(2,4)+G(1,5)}{2}\right] + C \cdot G(2,6) + C \cdot G(3,5)$$

$$\hat{G}(2,3) = A \cdot \left[\frac{G(3,3)+G(4,4)}{2}\right] + B \cdot \left[\frac{G(2,6)+G(1,5)}{2}\right] + C \cdot G(2,4) + C \cdot G(3,5)$$

$$\hat{G}(3,5) = A \cdot \left[\frac{G(4,6)+G(3,7)}{2}\right] + B \cdot \left[\frac{G(5,9)+G(6,8)}{2}\right] + C \cdot G(5,7) + C \cdot G(4,8)$$

$$\hat{G}(3,2) = A \cdot \left[\frac{G(3,3)+G(4,4)}{2}\right] + B \cdot \left[\frac{G(5,1)+G(6,2)}{2}\right] + C \cdot G(4,2) + C \cdot G(5,3)$$

Dalla figura 4.3.4, si nota che le otto maschere presentate ricoprono tutte le casistiche presenti nel pattern (ad eccezione dei bordi), la periodicità tra l'immagine originale e quella ricostruita e scalata è ancora di 6 a 4.

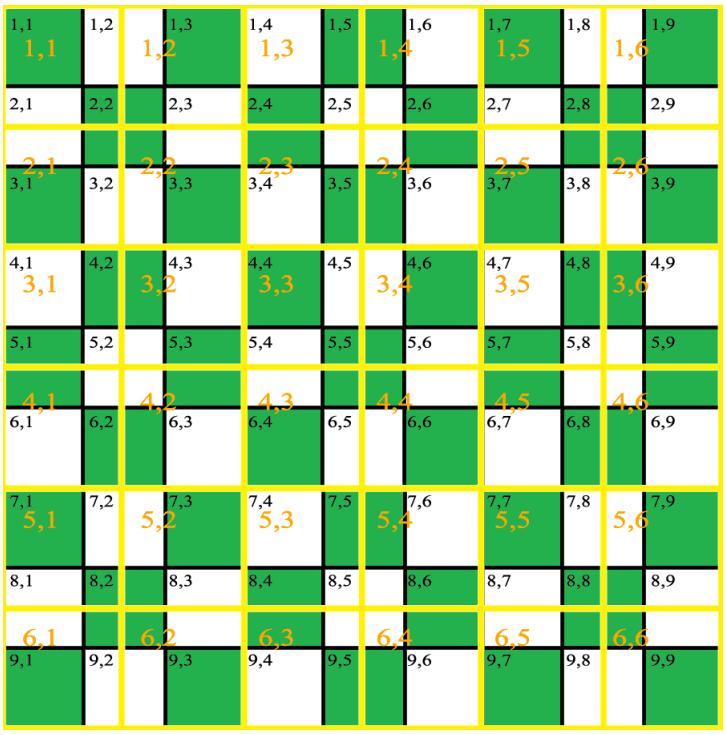


Figura 4.3.4: Il canale verde nel pattern Bayer originale

Esaminando ora i canali rosso e blu, si nota innanzitutto che sono tra loro simili, e che le operazioni che si effettueranno per un canale varranno pure per l'altro scambiando solamente le righe con le colonne; prendiamo in esame il canale rosso in figura 4.3.5, come in precedenza il contorno nero dei pixel si riferisce all'immagine originale e quello con contorno giallo a quella scalata.

Seguendo lo stesso filone di idee, ciascun pixel viene ricostruito con gli opportuni pesi $\{A, B, C\}$ moltiplicati per i valori dei pixel adiacenti e quindi sommati, balza subito all'occhio che disponendo della metà delle infomazioni rispetto al canale verde bisogna eseguire *in primis* una interpolazione. Consideriamo ad esempio il pixel :

$$\hat{R}(2,3) = A \cdot R(3,4) + B \cdot R(2,5) + C \cdot R(2,4) + C \cdot R(3,5)$$

dove $\{R(2,5), R(2,4), R(3,5)\}$ non sono noti dal pattern Bayer originale, per ricostruirli si possono utilizzare vari metodi, ad esempio l'interpolazione bilineare. Si può scrivere quindi $\hat{R}(2,3)$ come combinazione dei soli componenti di rosso presenti nell'immagine di Bayer:

$$\hat{R}(2,3) = \left[A + \frac{B}{4} + C\right] \cdot R(3,4) + \left[\frac{B}{4} + \frac{C}{2}\right] \cdot R(1,4) + \left[\frac{B}{4}\right] \cdot R(1,6) + \left[\frac{B}{4} + \frac{C}{2}\right] \cdot R(3,6)$$

Sviluppiamo anche il termine $\hat{R}(2,2)$:

$$\hat{R}(2,2) = \left[A + \frac{B}{4} + \frac{C}{4}\right] \cdot R(3,2) + \left[\frac{B}{4} + \frac{C}{2}\right] \cdot R(1,2) + \left[\frac{A}{2} + \frac{C}{4}\right] \cdot R(3,4) + \left[\frac{C}{4}\right] \cdot R(1,4)$$

e così via.

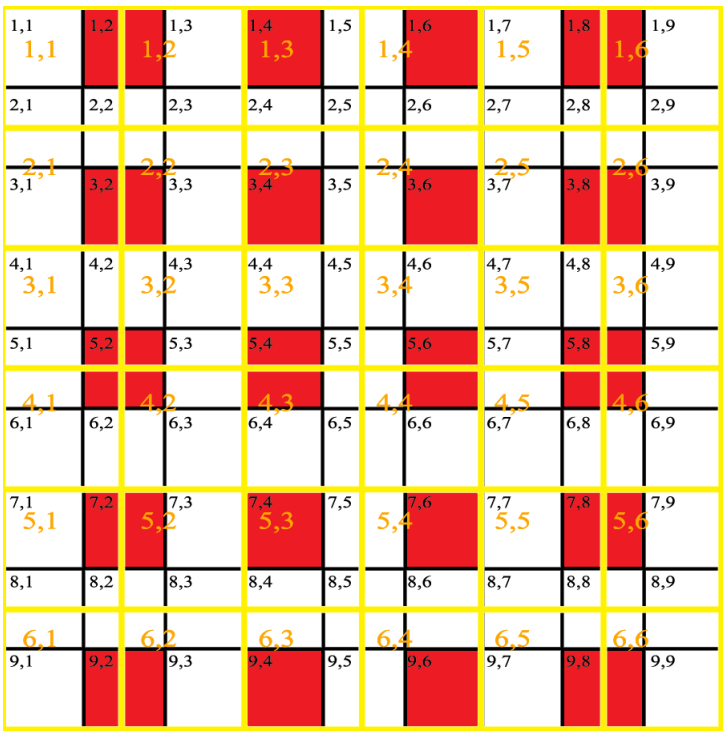


Figura 4.3.5: Il canale rosso nel pattern Bayer originale

4.4 Quarto metodo

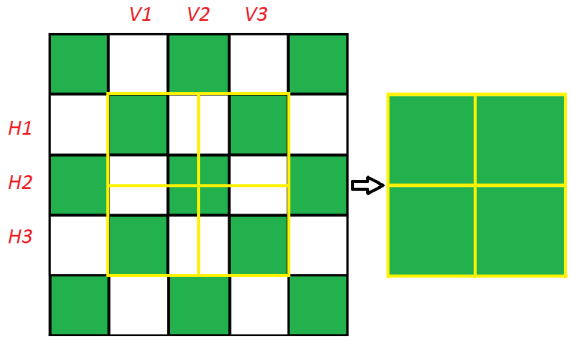
In questo metodo si calcola un *downscaling* e un *demosaicking* ancora in maniera congiunta, semplificando l'intera analisi in due blocchi di pixel di dimensione 3x3, e applicando un algoritmo *edge directed* eseguito solo sulla componente verde.

Un metodo di questo tipo, ovvero in grado di determinare la direzione di un contorno di una figura, ha inevitabilmente un grado di complessità maggiore rispetto ai precedenti, tuttavia porta a migliorie visive ed a bordi più marcati.

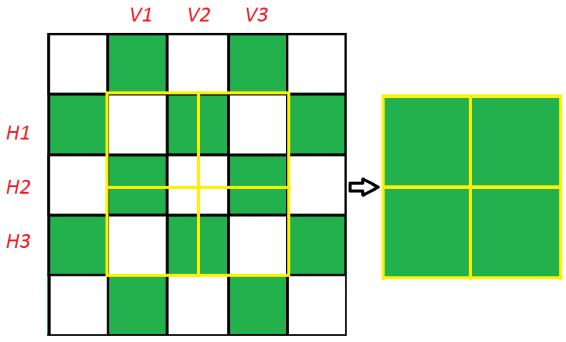
Esaminiamo la ricostruzione del canale verde, prendiamo la sola componente verde dell'immagine in ingresso decomposta con un CFA di Bayer, tale componente è formata da due tipologie di blocchi di dimensioni 3x3 pixel (contorno giallo in figura 4.4.1a e 4.4.1b), per ognuno di questi blocchi devo ricostruire un blocco di dimensione 2x2 pixel del colore verde completo.

Dovendo applicare un algoritmo di tipo *edge oriented* esamino un blocco più ampio dell'immagine originale, ovvero di dimensioni 5x5. In tale blocco (figura 4.4.1a) si identificano 3 colonne verticali e 3 colonne orizzontali, per ciascuno di questi elementi ne

stimo un ”gradiente”. L’elemento che ottiene un valore di gradiente minore di conseguenza identifica un bordo o un contorno, quindi costituisce una direzione da privilegiare in fase di interpolazione. Ad esempio un gradiente piccolo su V1 identifica un bordo sullo stesso, in questo caso ho scelto di mantere inalterati i due pixel presenti in V1 e copiarli nei due pixel sinistri dell’immagine ricostruita, mentre i pixel destri vengono ricostruiti con lo stesso algoritmo presentato nel metodo precedente. Lo stesso meccanismo viene appicato per V3, H1, H3, se invece viene trovato il gradiente minore per la serie di pixel V2 (o H2), siamo in presenza di un bordo verticale (o orizzontale) passante per il centro dei 4 pixel in uscita, questo caso può essere interpretato nella ricostruzione, tuttavia ho escluso l’eventualità di scartare tale informazione determinante, ed ho ripiegato con lo spostamento dei due pixel interpolati da V2 (o H2) con un’assegnazione sui due pixel di sinistra (o in basso) nel blocco 2x2 in uscita.



(a)

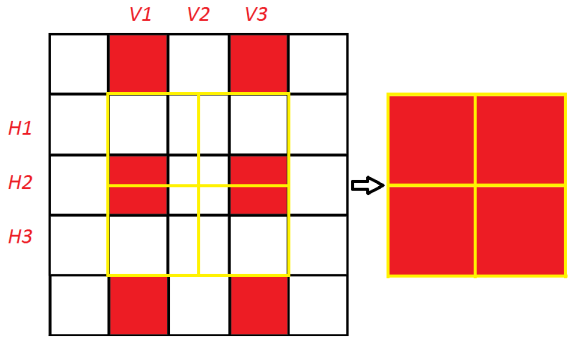


(b)

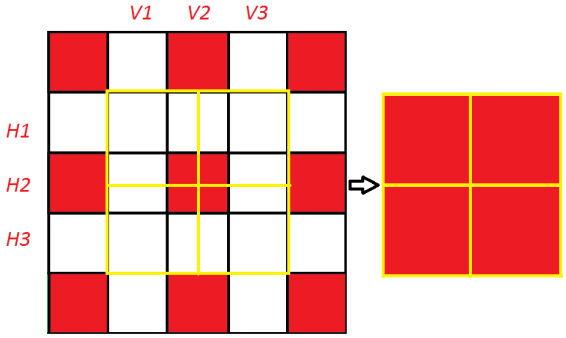
Figura 4.4.1: Maschere per il verde con metodo *edge directed*

Per quanto concerne la seconda maschera in figura 4.4.1b, ci si trova di fronte ad una situazione complementare. In questo caso una volta determinata la direzione privilegiata col gradiente minimo, se questa risultasse ad esempio V1, i pixel di sinistra del blocco 2x2 in uscita vengono ricostruiti con interpolazione sui soli componenti di V1 mentre quelli di destra con il solito metodo; lo stesso si applica a V3, H1, H3.

Le direzioni centrali V2, H2 vengono ancora una volta spostate sui pixel di sinistra o in basso, ricopiando i valori presenti su V2 o H2.



(a)

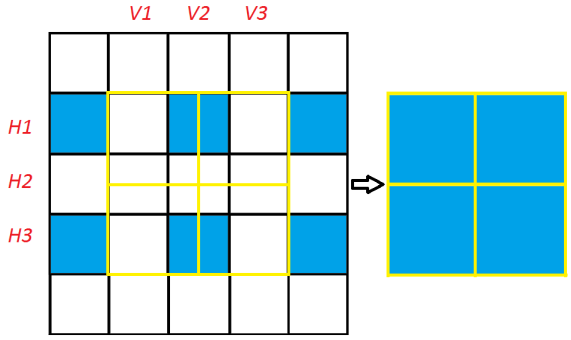


(b)

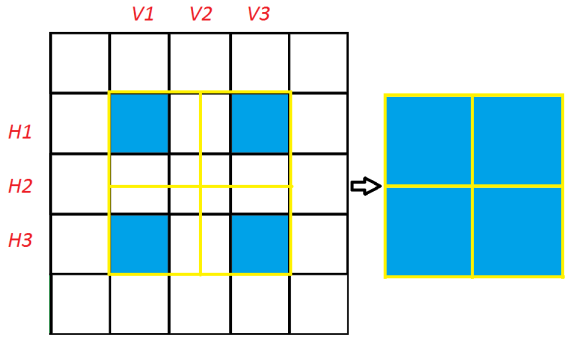
Figura 4.4.2: Maschere per il rosso con metodo *edge directed*

Passiamo ora in esame il canale rosso in figura 4.4.2a. Inizialmente vengono utilizzate le informazioni riguardo al tipo di bordo già calcolate per il canale verde in figura 4.4.1a, quindi ad esempio se si era trovato un gradiente minimo in V1 si utilizza un'interpolazione verticale tra i pixel presenti in V1, per le direzioni V2 si può utilizzare ad esempio il metodo bilineare e spostarlo con lo stesso criterio usato per i verdi cioè sui pixel in uscita a sinistra, per H2 invece vengono ricopiati i due valori nella riga in basso. Per la figura 4.4.2b, nel caso di V1, V3, H1, H3 si utilizza un'interpolazione bilineare; per V2 e H2, un'interpolazione verticale o orizzontale seguita dallo spostamento.

Per il canale blu, si effettuano le stesse considerazioni ma sulle figure 4.4.3a e 4.4.3b.



(a)



(b)

Figura 4.4.3: Maschere per il rosso con metodo *edge directed*

Per ricostruire la totalità dell'immagine si nota subito che bisogna applicare in successione le maschere (a) e (b) per i tre canali nella successione indicata in figura 4.4.4, con (a') e (b') vengono indicate le maschere (a) e (b) con inversione tra i colori blu e rossi.

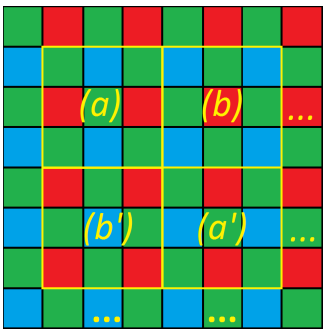


Figura 4.4.4: Applicazione maschere sull'immagine intera

4.5 Confronto tra i metodi

Nelle immagini che seguono è possibile osservare un confronto tra i vari metodi proposti. Le immagini di partenza sono state rese disponibili da ARRI, e sono immagini a 16 bit di dimensione 2880x1620 pixel. Inizialmente in figura 4.5.1, si è solamente ricostruita l'immagine seguendo il metodo DDFAPD[13], lasciando inalterata la risoluzione.

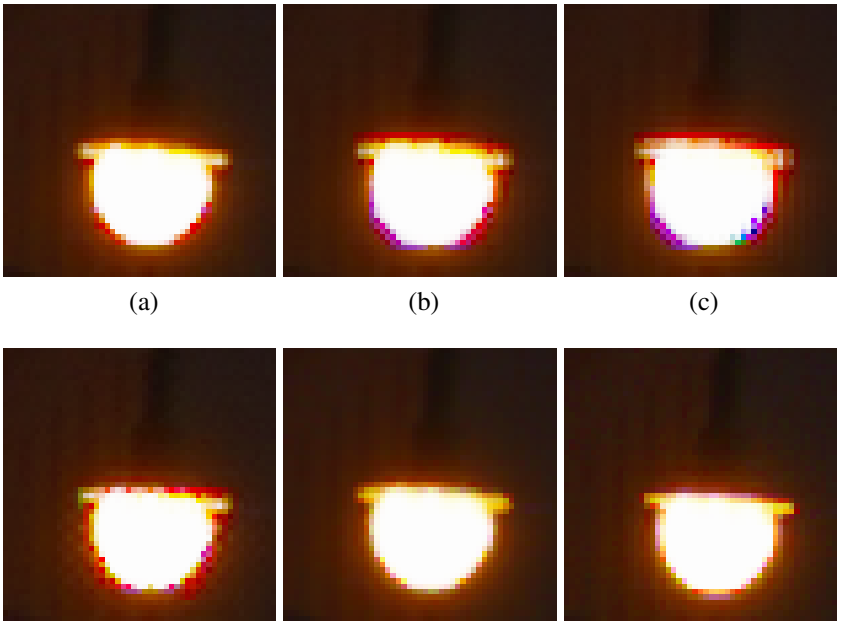
Nelle figure 4.5.2, 4.5.3, 4.5.5 e 4.5.4 invece si possono confrontare i vari metodi, partendo da in alto a sinistra e procedendo verso destra si trovano: (a) demosaicking con l'algoritmo DDFAPD seguito dal processo di "area pesata" (fig 4.3.2) che può essere usata come immagine di riferimento per il confronto, (b) upsampling con ripetizione del blocco (sez4.1.1) e downsampling con media uniforme (sez 4.1.2) sul pattern di Bayer seguito dal demosaicking DDFAPD, (c) upsampling (sez 4.1.1) e downsampling (sez 4.1.2) e demosaicking DDFAPD, (d) ridimensionamento dell'immagine di Bayer (sez 4.2) e demosaicking DDFAPD, (e) metodo congiunto di ricostruzione e ridimensionamento (sez 4.3) e (f) metodo congiunto di ricostruzione e ridimensionamento *edge directed* (sez 4.4).

(a)

(b)

(c)

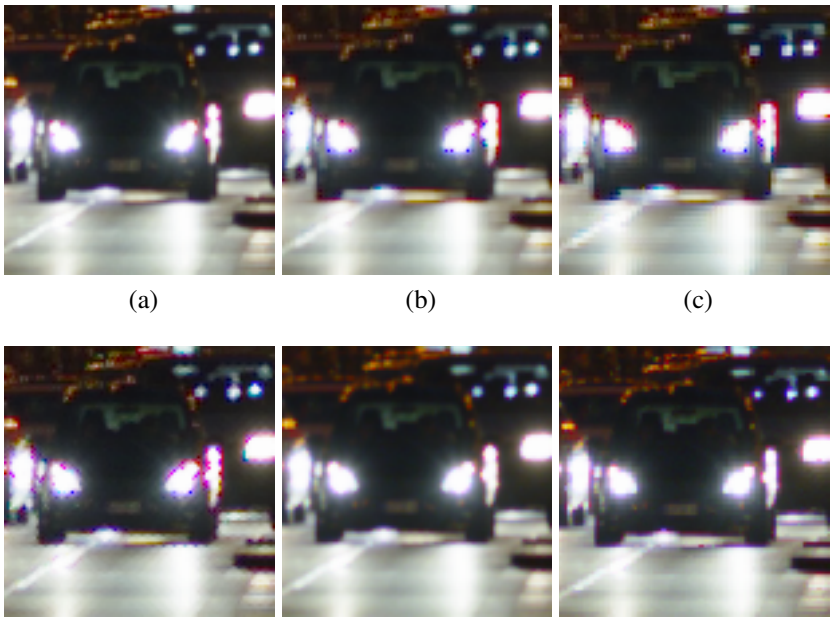
Figura 4.5.1: Immagini originali ricostruite senza ridimensionamento



(a) (b) (c)

(d) (e) (f)

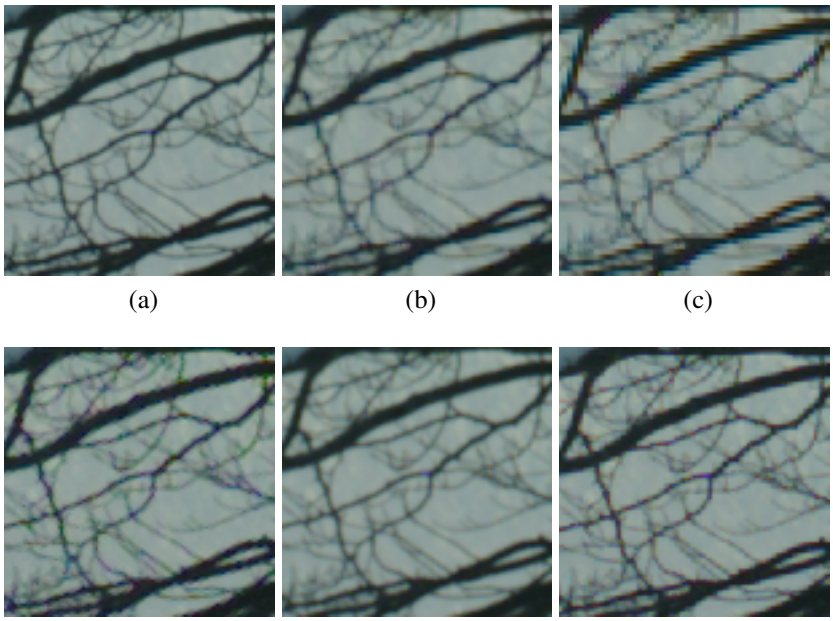
Figura 4.5.2: Confronto sulla prima immagine



(a) (b) (c)

(d) (e) (f)

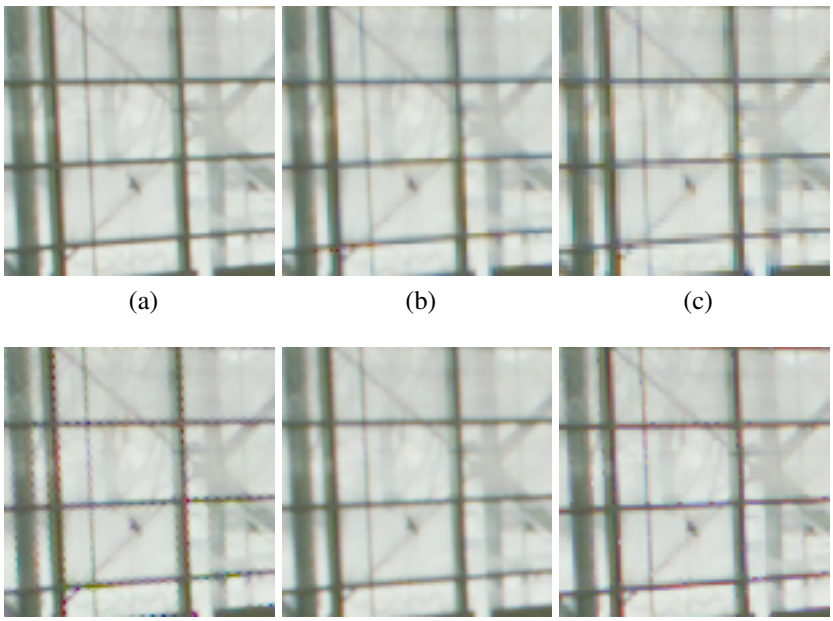
Figura 4.5.3: Confronto sulla prima immagine



(a) (b) (c)

(d) (e) (f)

Figura 4.5.4: Confronto sulla seconda immagine



(a) (b) (c)

(d) (e) (f)

Figura 4.5.5: Confronto sulla terza immagine

Conclusioni

Nella stesura di questa tesi ho cercato di trattare gli aspetti più importanti nel processo di ricostruzione di un'immagine a colori. Innanzitutto si è visto che non esiste un unico metodo per acquisire un'immagine: anteponendo differenti tipologie di CFA ad un sensore è possibile raccogliere la luce in modi diversi, ottenendo migliorie a livello qualitativo. Avendo a disposizione un sensore con filtri intercambiabili sarebbe stato interessante condurre degli studi su quali matrici di filtri aumentino le prestazioni qualitative.

Nel corso degli anni sono stati sviluppati molti metodi, che appartengono a cinque principali categorie: metodi euristici, metodi *edge-directed*, metodi nel dominio della frequenza, metodi basati sulle *wavelet*, metodi di ricostruzione. Alcuni sono semplici, tal altri complessi e con artefatti più o meno visibili. Bisogna però chiarire che molti di questi algoritmi sono stati sviluppati per ottenere degli ottimi risultati sul campionario di foto proposto da Kodak [23], infatti simulando la ricostruzione con immagini *raw* scattate in condizioni reali (messe a disposizione da ARRI) si è visto che non sempre un buon algoritmo per le immagini del campionario Kodak è un buon algoritmo per le immagini reali.

Nel corso delle mie analisi ho cercato il più possibile di comprendere cosa fosse in realtà una buona ricostruzione di un'immagine a colori, ho sviluppato i principali metodi di analisi delle prestazioni suggerite in letteratura, spaziando dal semplice calcolo del MSE, agli spazi uniformi, al calcolo dello *zipper effect*, al MMSIM, a seguito di molte simulazioni di ricostruzione delle immagini con gli algoritmi di demosaicking in mio possesso, le analisi qualitative di rado hanno suggerito gli stessi algoritmi vincitori. Non credo che i metodi di analisi quantitativi debbano essere presi come verità assolute ma solamente come un metro di riferimento che dia una chiave di lettura iniziale al modo di ricostruire un'immagine; l'ultima analisi qualitativa deve essere fatta dall'uomo solamente con l'osservazione vera e propria e il confronto diretto tra le immagini ricostruite, osservando ogni singolo bordo, colore e pixel.

Il fatto che la fotografia digitale sia arrivata ad un così grande successo e venga inserita nei dispositivi più vari, fa anche si che non sempre venga scelto l'algoritmo di demosaicking migliore e poiché in questo campo raramente qualità e velocità viaggiano nella stessa direzione, per ogni singola applicazione si utilizzi il metodo più opportuno.

L'esempio più lampante viene riportato nell'ultimo capitolo di questa tesi, dovendo fornire un'anteprima al volo di fotogrammi acquisiti con telecamere cinematografiche, i

metodi di ricostruzioni scelti non possono essere troppo pesanti dal punto di vista computazionale ma tuttavia efficienti a livello qualitativo, per questo motivo ho scelto la strada euristica ed al più *edge-directed*.

Ringraziamenti

Vorrei ringraziare tutte le persone che mi hanno accompagnato in questi duri anni di studio universitario, in primo luogo la mia famiglia: mamma, papà e sorella, che mi hanno sempre sostenuto ed aiutato.

La mia dolce metà Monica, che mi è sempre stata vicina, in tutti i momenti belli e brutti che fossero.

Il Professor Calvagno, per la disponibilità e l'aiuto nella compilazione di questa tesi.

L'Ing. Daniele Menon, e l'Ing. Stefano Andriani per il materiale resomi disponibile ed i consigli dati.

Gli amici dell'università , che hanno saputo alleggerire ed allietare le interminabili giornate di studio.

Gli amici del CUS, per tutti i bei momenti passati insieme.

Un ringraziamento speciale a tutti voi, non vi elenco solo perché non vorrei dimenticare nessuno!

Elenco delle figure

1.0.1 Sistema di acquisizione con CFA 6
1.0.2 Confronto tra sistema con CFA e un sensore Foveon X3 7
1.0.3 Processo di ricostruzione [30] 8
1.0.4 Tipologie di *color filter array* 9
1.0.5 La struttura dell'occhio 10
1.0.6 Struttura nervosa nella retina 11
1.0.7 Coni e Bastoncelli 12

2.0.1 Green channel 14
2.0.2 Red channel 14
2.0.3 Esempio interpolazione bilineare 15
2.1.1 Bayer pattern 16
2.1.2 Esempio metodo di Cok 16
2.1.3 La tipica immagine Bayerizzata tratta da un CCD singolo 17
2.1.4 Coefficienti proposti per il filtro lineare 19
2.1.5 Immagine ricostruita 21
2.2.1 Bayer pattern 24
2.2.2 Esempio di interpolazione Hamilton-Adams 24
2.2.3 Esempio di interpolazione Hirakawa-Parks 25
2.2.4 Esempio di interpolazione Zhang-Wu 26
2.3.1 Densità spettrale di potenza dell'immagine Kodak del faro 28
2.3.2 Filtraggi ideali per l'interpolazione del CFA 29
2.3.3 Filtraggi ideali passa-alto per il canale verde 30
2.3.4 Trasformata di Fourier del filtro passa-basso proposto in [41] 30
2.3.5 Spettro del filtro passa-basso per la ricostruzione della luminanza del verde 31
2.3.6 Banco di filtri utilizzato in [10] 31
2.4.1 Banco di filtri utilizzato in [44] 34
2.6.1 Demosaicking dell'immagine Kodak nr.1 38
2.6.2 Demosaicking dell'immagine Kodak nr.8 39
2.6.3 Demosaicking dell'immagine Kodak nr.1 40

3.1.1 Interpretazione degli assi CIELAB 45

3.1.2 Modello S-CIELAB[36] 47
3.2.1 Blocco di pixel in esame 49
3.3.1 Schema generico di un sistema di valutazione basato sugli errori. 52
3.3.2 Diagramma della struttura del sistema SSIM. 53
3.4.1 Digramma per il calcolo del DI-PSNR 56
3.4.2 Esempio di *DI-difference map* (errori amplificati) 57
3.4.3 CFA adattato 58
3.4.4 Schema a blocchi utlizzato 58
3.4.5 Esempio tratto da Matlab 59
3.4.6 Risultati della simulazione 60
3.5.1 Esempio del metodo di analisi 65

4.1.1 Schema del processo 72
4.1.2 Schema del processo: fattore di campionamento fratto 72
4.1.3 Schema del processo: interpolazione e decimazione 72
4.1.4 Blocco di 2x2 pixel di Bayer 73
4.1.5 Interpolazione su blocchi di pixel 73
4.1.6 Decimazione di blocchi di pixel 74
4.1.7 Decimazione con media pesata 74
4.2.1 UpSampling & DownSampling 76
4.2.2 Upsampling & Downsampling su immagine di Bayer 77
4.3.1 Joint demosaicking & downscaling 78
4.3.2 Suddivisione del pixel ricostruito 78
4.3.3 Maschere per il Green channel 79
4.3.4 Il canale verde nel pattern Bayer originale 81
4.3.5 Il canale rosso nel pattern Bayer originale 82
4.4.1 Maschere per il verde con metodo *edge directed* 83
4.4.2 Maschere per il rosso con metodo *edge directed* 84
4.4.3 Maschere per il rosso con metodo *edge directed* 85
4.4.4 Applicazione maschere sull'immagine intera 85
4.5.1 Immagini originali ricostruite senza ridimensionamento 87
4.5.2 Confronto sulla prima immagine 88
4.5.3 Confronto sulla prima immagine 88
4.5.4 Confronto sulla seconda immagine 89
4.5.5 Confronto sulla terza immagine 89

Elenco delle tabelle

2.1 PSNR a confronto [dB] 20
2.2 Tempi di eleaborazione 22

3.1 Parametri per il modello S-CIELAB HVS 45
3.2 SCIELAB delle immagini Kodak, algoritmi 0-8 47
3.3 SCIELAB delle immagini Kodak, algoritmi 9-17 48
3.4 ZE delle immagini kodak, algoritmi 0-8 50
3.5 ZE delle immagini kodak,algoritmi 9-17 51
3.6 SSIM index delle immagini Kodak, algoritmi 0-8 54
3.7 SSIM index delle immagini Kodak,algoritmi 9-17 55
3.8 PSNR [dB] delle immagini kodak, algoritmi 0-8 60
3.9 PSNR [dB] delle immagini kodak,algoritmi 9-17 61
3.10 DI-PSNR [dB] delle immagini kodak, algoritmi 0-8 62
3.11 DI-PSNR [dB] delle immagini kodak,algoritmi 9-17 63
3.12 PSNR [dB] delle immagini kodak in scala di grigio, algoritmi 0-7 66
3.13 PSNR [dB] delle immagini kodak in scala di grigio, algoritmi 8-16 67
3.14 (LAB) delle immagini kodak in scala di grigio, algoritmi 0-7 68
3.15 (LAB) delle immagini kodak in scala di grigio,algoritmi 8-16 69

Bibliografia

[1] D.R. Cok, *Signal processing method and apparatus for producing interpolation and chrominance values in a sampled color image signal*, 1986.

[2] W.T. Freeman, *Method and apparatus for reconstructing missing color samples*, 1988

[3] R.H. Hibbard, *Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients*, 1995.

[4] J.E. Adams, J.F. Hamilton *Design of practical color filter array interpolation algorithms for digital cameras*, Feb. 1997

[5] J.E. Adams, J.F. Hamilton *Adaptive color plane interpolation in a single sensor color electronic cameras*, 1997

[6] K. Hirakawa, T.W. Parks, *Adaptive Homogeneity-directed demosaicing algorithm*, Proc. IEEE Trans. Image Processing. 2002.

[7] K. Hirakawa, T.W. Parks, *Adaptive homogeneity directed demosaicing algorithm*, IEEE Trans. Image Process. 14 (3) (2005) 360–369.

[8] X. Wu, N. Zhang, *Primary-Consistent Soft-Decision Color Demosaiking for Digital Cameras*, IEEE Trans. Image Processing, vol. 13, No. 9, Sept. 2004.

[9] X. Wu, N. Zhang, *Color demosaicing via directional linear minimum mean square-error estimation*, IEEE Trans. Image Processing, vol. 14, 2005.

[10] B.K. Gunturk, Y. Altunbasak, R.M. Mersereau, *Color plane interpolation using alternating projections*, IEEE Trans. Image Process. 11 (9) (2002) 997–1013.

[11] Henrique S. Malvar, Li-wei He, and Ross Cutler*, High-quality linear interpolation for demosaicing of bayer-patterned color images,* Microsoft Research One Microsoft Way, Redmond, WA 98052, USA

[12] S. Ferradans, M. Bertalmío, V. Caselles, *Correspondence Geometry-Based Demosaicking*, IEEE transactions on image processing, Vol. 18, No. 3, March 2009

[13] D. Menon, S. Andriani, and G. Calvagno, *Demosaicing with directional filtering and a posteriori decision,* IEEE Trans. Image Processing, vol. 16, no. 1, pp.132–14, Jan. 2007

[14] Snell & Wilcox LTD, *Method and apparatus for spatial interpolation of colour images*, Dec 2006

[15] X. Li, *Demosaicing by successive approximation*, IEEE Trans. Image Process. 14 (3) (2005) 370–379

[16] N.-X. Lian, L. Chang, Y.-P. Tan, V. Zagorodnov, *Adaptive filtering for color filter array demosaicking*, IEEE Trans. Image Process. 16 (10) (2007) 2515–2525.

[17] E. Dubois, *Frequency-domain methods for demosaicking of Bayer-sampled color images*, IEEE Signal Process. Lett. 12 (12) (2005) 847–850

[18] D. Menon, G. Calvagno, *Demosaicing based on wavelet analysis of the luminance component*, Proceedings of the IEEE International Conference on Image Processing, vol. 2, 2007, pp. 181–184.

[19] D. Menon, *Demosaicing based on wavelet analysis & directional filtering (metodo rifiutato a SPL e TIP)*

[20] D. Menon and G. Calvagno, *A regularization approach to demosaicking*, Proc. of IS&T/SPIE Conference on Visual Communication and Image Processing (VCIP), Jan. 2008

[21] Eric Dubois, *Frequency-Domain Methods for Demosaicking of Bayer-Sampled Color Images*, IEEE signal processing letters, vol. 12, no. 12, december 2005 847

[22] S.-C. Pei and I.-K. Tam, *Effective color interpolation in CCD color filter array using signal correlation*, Proc. ICIP, pp. 488–491, Sept. 2000.

[23] http://r0k.us/graphics/kodak/

[24] D. Menon and G. Calvagno, *Regularization approaches to demosaicking*, IEEE Trans. on Image Processing, vol. 18, no. 10, pp. 2209-2210, Oct. 2009.

[25] Yi-Nung Liu, Yi-Chun Lin, Shao-Yi Chien, *A No-reference Quality Evaluation Method for CFA Demosaicking,* 2010 IEEE

[26] http://it.wikipedia.org/wiki/Fotorecettore

[27] Colorimetria tristimolo e misurazione strumentale del colore della carta, Rosso Gabriele, Scuola Interregionale di Tecnologia per Tecnici Cartari di Verona

[28] R. Lukac, K.N. Plataniotis, *Color filter arrays: design and performance analysis*, IEEE Trans. Consum. Electron. 51 (4) (2005) 1260–1267.

[29] K. Hirakawa, P.J. Wolfe, *Spatio-spectral color filter array design for optimal image recovery*, IEEE Trans. Image Process. 17 (10) (2008) 1876–1890.

[30] G.Sharma, *Digital color imaging Handbook*, CRC Press LLC 2003

[31] D.H. Brainard, *Bayesian method for reconstructing color images from thrichromatic samples*, Proceedings of the IS&T 47th Annual Meeting1994, pp. 375–380.

[32] Wyszecki G. and W. S., Stiles, *Color Science*, John Wiley & Sons, New York, 1982.

[33] G. A. Gescheider, *Psychophysics: The Fundamentals*, 3rd ed., Lawrence Erlbaum Assoc., Mahwah, NJ, 1997.

[34] J. Mukherjee, R. Parthasarathi, S. Goyal, *Markov random field processing for color demosaicing*, Pattern Recognition Lett. 22 (3–4) (2001) 339–351

[35] D. Taubman, *Generalized Wiener reconstruction of images from colour sensor data using a scale invariant prior*, Proceedings of the IEEE International Conference on Image Processing, vol. 3, 2000, pp. 801–804

[36] http://white.stanford.edu/~brian/scielab

[37] W.Lu and Y.P. Tan, *Color Filter Array Demosaicking: New Method and Performance Measures*, 2003

[38] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, *Image quality assessment: From error visibility to structural similarity*, IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.

[39] Z. Wang and A.C. Bovik, *A Universal Image Quality Index*, IEEE Signal Processing Letters, vol. 9, no. 3, pp. 81-84, March 2002

[40] J.W. Glotzbach, R.W. Schafer, K. Illgner, *A method of color filter array interpolation with alias cancellation properties*, Proceedings IEEE International Conference on Image Processing, vol. 1, 2001, pp. 141–144.

[41] D. Alleysson, S. Susstrunk, and J. Herault, *Linear demosaicking inspired by human visual system*, IEEE Trans. on Image Processing, vol. 14, no. 4, pp. 439–449, 2005

[42] N.-X. Lian, L. Chang, Y.-P. Tan, V. Zagorodnov, *Adaptive filtering for color filter array demosaicking*, IEEE Trans. Image Process. 16 (10) (2007) 2515–2525.

[43] Y.M. Lu, M. Karzand, M. Vetterli, *Demosaicking by alternating projections: theory and fast one step implementation*, IEEE Trans. Image Process. 19 (8) (2010) 2085–2098

[44] J. Gu, P.J. Wolfe, K. Hirakawa, *Filterbank-based universal demosaicking*, Proceedings of the IEEE International Conference on Image Processing, 2010, pp. 1981–1984.

[45] D. Menon, G. Calvagno, *Color image demosaicking: An overview*, Elsevier, Signal Processing: Image Communication 26 (2011) 518–533

[46] J. Mairal, M. Elad and G. Sapiro, *Sparse Representation for Color Image Restoration,* IEEE Transactions on image processing, vol. 17, no. 1, January 2008 53