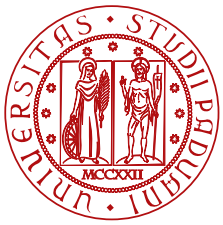


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

Stato dell'arte nella compressione di immagini senza perdita di informazione

Relatore
Prof. Giancarlo Calvagno

Laureando
Enrico Saro

ANNO ACCADEMICO 2023-2024

Data di laurea 26/09/2024

Sommario

La compressione di immagini lossless è fondamentale per l'archiviazione e la trasmissione di immagini quando la perdita di informazioni non è permessa. Nel corso degli anni sono stati sviluppati vari standard di compressione. Questo lavoro esamina il funzionamento delle diverse tecniche di compressione di immagini senza perdita di informazione, con un'attenzione particolare all'attuale stato dell'arte in questo campo. L'obiettivo è verificare se i progressi tecnologici degli ultimi anni hanno portato a miglioramenti significativi, ossia valutare di quanto gli standard più recenti sono superiori ai precedenti, in termini di rapporto di compressione e altri fattori chiave.

Indice

Capitolo 1

Introduzione

1.1 Compressione dati

Ogni giorno viene prodotta un'enorme quantità di dati digitali. La compressione di questi ultimi è una tecnologia cruciale nel mondo contemporaneo, dove l'efficienza nella trasmissione e nell'archiviazione delle informazioni riveste una importanza primaria. La compressione dei dati è una tecnica di elaborazione che, attraverso l'uso di opportuni algoritmi, permette di ridurre la quantità di bit necessari alla rappresentazione in forma digitale di un'informazione.

Questa compressione è possibile grazie alla presenza di una significativa componente di ridondanza nei dati di interesse comune, che può essere sfruttata per rappresentare la stessa informazione con un numero minore di bit.

1.1.1 Compressione Lossless e Lossy

La compressione dati può essere suddivisa in due tipologie: lossless e lossy.

Un algoritmo di compressione coinvolge in realtà due processi distinti: un algoritmo di encoding trasforma un input X nell'output X_C, mentre un algoritmo di decoding parte da X_C per ricostruire Y.

Se l'output Y è identico all'input X, allora il file originale può essere completamente ricostruito senza alcuna perdita di informazione e la tecnica di compressione dati si dice lossless (senza perdita). Viceversa, se l'output è diverso dall'input, si parla di compressione lossy (con perdita), in quanto non è possibile recuperare una parte dell'informazione originale.

1.1.2 Parametri di valutazione

Per valutare un algoritmo di compressione lossless possono essere presi in considerazione diversi parametri. I più importanti sono tre: rapporto di compressione, tempo di codifica (encoding

time) e di decodifica (decoding time)

Il rapporto di compressione (Compression Ratio, CR) è definito come il rapporto tra la dimensione del file originale e la dimensione del file compresso.

Il tempo di codifica e decodifica deve essere misurato sullo stesso hardware e con le stesse configurazioni software per garantire un confronto equo.

1.2 Compressione di immagini

Le immagini rappresentano una tipologia specifica di dati. Per comprimere le immagini in modo efficiente, gli algoritmi di compressione devono tenere in considerazione le peculiarità delle immagini come tipo di dato. Gli algoritmi di compressione di immagini possono essere sia con perdita sia senza perdita.

La compressione dati lossy può fornire rapporti di compressione da 10:1 fino a 50:1, mantenendo una buona percezione dell'immagine ricostruita [1].

Al contrario, la compressione lossless solitamente non permette un rapporto così elevato, ma ci sono aree di utilizzo, come medicina [2] [3], archiviazione, astronomia [4], settori scientifici e industriali, in cui avere una replica esatta dell'immagine originale è preferibile o addirittura necessario.

In questo lavoro l'attenzione sarà concentrata sulla codifica per immagini lossless.

1.2.1 Panoramica sui metodi di compressione

La compressione di immagini senza perdita può essere sempre modellata come una procedura articolata in due fasi: decorrelazione e codifica.

La prima fase ha l'obiettivo di eliminare la ridondanza spaziale o inter-pixel, e viene attuata principalmente attraverso due approcci: trasformazione e predizione. Questa fase è la più importante e divide gli algoritmi di compressione in "transform coding" se viene effettuata una trasformazione e "predictive coding" se sono basati su predizione.

La seconda fase riguarda la codifica. Essa viene eseguita utilizzando algoritmi basati su dizionario (come LZW), o tramite codifica entropica (come Huffmann e arithmetic coding).
Da decenni le prestazioni delle tecniche di codifica sono vicine ai limiti teorici, per questo le attività di ricerca si concentrano maggiormente sulla fase di decorrelazione.

In questa sezione verranno trattati in modo generale il "transform coding" e il "predictive coding". Nei prossimi due capitoli verranno studiati in modo approfondito i principali standard di compressione basati sulla predizione, storici e recenti.

Transform coding

La strategia base del transform coding consiste nel trasformare la sequenza di input in una diversa sequenza, in cui è più semplice codificare l'informazione (o eliminare l'informazione meno significativa, se la compressione è lossy). Questo approccio si basa sui tre seguenti step [5]:

- **Trasformazione**: La sequenza di dati originale viene suddivisa in blocchi. Ogni blocco viene poi mappato in una sequenza mediante una trasformazione, spesso lineare e reversibile. Esempi di trasformazione sono Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT) o Discrete Wavelet Transform (DWT).
- **Quantizzazione**: I coefficienti risultanti dalla trasformata vengono quantizzati, riducendo la precisione di rappresentazione e quindi il numero di bit necessari per rappresentarli. Questo step introduce una perdita di informazione e per questo è da evitare nel caso di codifica lossless.
- **Codifica**: I valori quantizzati vengono codificati utilizzando tecniche di codifica entropica o basate su dizionario.

Il transform coding è ampiamente utilizzato nella codifica lossy di immagini; un esempio è lo standard JPEG, che utilizza la DCT [6].

La maggior parte delle trasformate, come la DCT e la DFT, sono difficili da applicare alla codifica lossless dei segnali, perché i loro coefficienti di trasformazione sono a valore reale o complesso e la quantizzazione porta inevitabilemente a una perdita di informazione.
Esistono degli standard di compressione lossless basati su trasformazione, ad esempio JPEG 2000 o JPEG-XR. Questi ultimi devono rendere l'intero processo reversibile; ciò è possibile tramite una trasformazione che mappa da interi a interi per evitare il passo della quantizzazione [7] [8].

Predictive coding

La codifica predittiva è composta da due fasi distinte:

- **Predizione dei pixel**: In un'immagine i pixel sono altamente correlati tra loro. A causa di questa correlazione, un pixel può essere predetto con una buona accuratezza dai pixel vicini. Particolare attenzione va posta su quali pixel possono essere usati per la predizione: questi ultimi devono essere già visitati secondo la modalità di scansione scelta. Solitamente, la scansione scelta è Row-major, quindi i pixel noti sono quelli posti alla sinistra e nelle righe superiori rispetto al pixel su cui va fatta la predizione, come si può vedere in Figura 1.1.

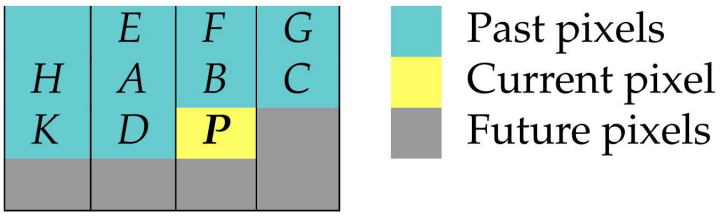


Figura 1.1: I pixel disponibili per la predizone di P. Credit: *The Impact of State-of-the-Art Techniques for Lossless Still Image Compression* [9].

- **Codifica dell'errore**: Il valore predetto viene sottratto dal valore effettivo del pixel corrispondente, generando un errore di predizione. Questo errore viene quindi codificato per poi essere trasmesso o archiviato.
  Se la predizione è buona, l'errore ha una alta probabilità di essere vicino a zero. Per questo la codifica entropica o basata su dizionario dell'errore residuo è particolarmente efficace e permette un grande risparmio di bit.

Capitolo 2

Standard di compressione lossless

Negli ultimi decenni sono stati sviluppati diversi standard di compressione di immagini lossless. Questo capitolo si sofferma su alcuni dei più importanti e utilizzati, focalizzandosi sul loro funzionamento, i loro vantaggi e i loro punti deboli.

2.1 GIF

Il formato GIF (Graphics Interchange Format) è un formato di compressione di immagini rilasciato nel 1987.
GIF presenta una eccezione tra i principali standard di compressione di immagini, in quanto non prevede né la fase di predizione, né di trasformazione, ma solo una compressione tramite dizionario LZW (Lempel-Ziv-Welch) [10].
L'algoritmo LZW è una tecnica di compressione senza perdita, che funziona creando un dizionario di sequenze di pixel durante la lettura dell'immagine, risparmiando spazio soprattutto in presenza di sequenze ripetute.

Il formato GIF supporta un massimo di 256 colori, tra cui il canale alfa, il canale che descrive il grado di trasparenza/opacità di ogni determinato pixel. Per comprimere un'immagine con più colori tramite GIF, l'immagine originale deve essere suddivisa in regioni più piccole con non più di 256 colori diversi. Ciascuna di queste regioni viene quindi memorizzata come blocco immagine separato con la propria palette locale [11].

Sebbene GIF funzioni bene con grafiche generate al computer o immagini composte da pochi colori, in genere non è il modo più efficiente per comprimere fotografie o scene naturali, in cui non sono presenti molte sequenze ripetute. Questo si può vedere in Tabella 2.1, tratta da *Introduction to Data Compression* [5], dove sono paragonate le dimensioni in byte di quattro immagini, compresse sia tramite lo standard GIF, sia tramite la codifica aritmetica dei valori dei pixel e tramite la codifica aritmetica della differenza tra i due pixel vicini. Si nota che se le

immagini sono foto naturali, la codifica GIF è paragonabile ad un semplice arithmetic coding ed ha risultati ben peggiori rispetto alla codifica aritmetica della differenza tra due pixel vicini.

Image	GIF	Arithmetic Coding of Pixel Values	Arithmetic Coding of Pixel Differences
Sena	51,085	53,431	31,847
Sensin	60,649	58,306	37,126
Earth	34,276	38,248	32,137
Omaha	61,580	56,061	51,393

Tabella 2.1: Comparazione di GIF con arithmetic coding. [5].

2.2 JPEG (old standard)

JPEG lossless è stato sviluppato come aggiunta a JPEG nel 1993, utilizzando una tecnica completamente diversa dallo standard JPEG lossy. Questo standard è ormai considerato obsoleto ed è stato sostituito da altri più efficienti, come JPEG-LS.

La predizione dei pixel si basa sui tre pixel causali, cioè già analizzati in precedenza, più vicini. Come si può vedere in figura 2.3, X è il pixel su cui va fatta la predizione, W il pixel a sinistra (ovest), NW il pixel in alto a sinistra (nordovest), N il pixel in alto (nord).

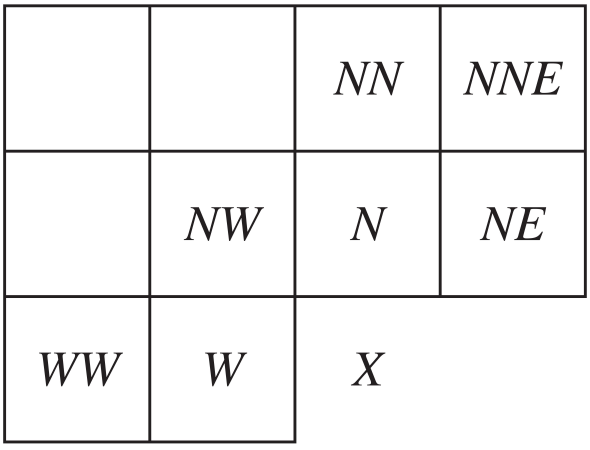


Figura 2.1: Pixel utilizzabili per la predizione di X. Credit: *Introduction to Data Compression* [5].

Questo standard offre otto diversi schemi di predizione selezionabili dall'utente. Il primo non effettua alcuna predizione. Gli altri sette sono qui elencati:

1. $\hat{X} = W$
2. $\hat{X} = N$
3. $\hat{X} = NW$
4. $\hat{X} = W + N - NW$
5. $\hat{X} = W + \frac{N-NW}{2}$
6. $\hat{X} = N + \frac{W-NW}{2}$
7. $\hat{X} = \frac{W+N}{2}$

Immagini diverse possono avere strutture diverse, che possono essere sfruttate al meglio da una delle otto modalità di predizione.

Se la compressione non viene eseguita in un ambiente in tempo reale, ad esempio per l'archiviazione, si possono provare tutte le otto modalità di predizione e utilizzare quella che offre la maggiore compressione. La modalità utilizzata per eseguire la predizione può essere memorizzata in un'intestazione a 3 bit insieme al file compresso.

Gli autori del libro *Introduction to Data Compression* [5] hanno codificato quattro immagini di prova utilizzando le varie modalità JPEG. L'errore residuo è stato codificato utilizzando la codifica aritmetica. I valori sono riportati nella Tabella 2.2. I risultati migliori, cioè le dimensioni minori dei file compressi, sono indicati in grassetto nella tabella. Dalla tabella si evince che immagini diverse provocano risultati altamente variabili tra i vari schemi di predizione, non è possibile utilizzarne solo uno senza rischiare di incorrere in un pessimo rapporto di compressione.

Il principale svantaggio di questa tecnica è il tempo necessario a provare tutti gli otto metodi e la necessità di sceglierne uno nelle applicazioni real-time. Inoltre, tutti gli schemi di predizione sono lineari e una immagine reale ha spesso una struttura non lineare [9].

Image	JPEG 0	JPEG 1	JPEG 2	JPEG 3	JPEG 4	JPEG 5	JPEG 6	JPEG 7
Sena	53,431	37,220	31,559	38,261	31,055	**29,742**	33,063	32,179
Sensin	58,306	41,298	37,126	43,445	**32,429**	33,463	35,965	36,428
Earth	38,248	32,295	**32,137**	34,089	33,570	33,057	33,072	32,672
Omaha	56,061	**48,818**	51,283	53,909	53,771	53,520	52,542	52,189

Tabella 2.2: Dimensioni del file compresso (in byte) delle immagini ottenute utilizzando le varie modalità di predizione JPEG [5].

2.3 PNG

Il formato PNG (Portable Network Graphics) è uno standard di compressione di immagini lossless, introdotto nel 1996 come alternativa migliorata e priva di brevetti rispetto al formato GIF.
PNG supporta una vasta gamma di profondità di colore, dalla scala di grigi a immagini a colori di 24 e 48 bit, e include anche un canale alfa per la trasparenza [12].

Lo standard PNG predice il valore di un pixel in base ai suoi vicini causali N, W e NW. La differenza modulo 256 viene quindi codificata al posto del pixel originale.
Esistono quattro modi diversi per ottenere la stima (cinque se si include nessuna predizione):

- Il primo metodo utilizza il pixel della riga precedente come stima ($\hat{X} = N$).
- Il secondo metodo prevede l'utilizzo del pixel a sinistra come stima ($\hat{X} = W$).
- Il terzo metodo utilizza la media del pixel sopra e del pixel a sinistra ($\hat{X} = \frac{N+W}{2}$).
- L'ultimo metodo è più complesso: una stima iniziale del pixel viene fatta sommando il pixel a sinistra e quello sopra e sottraendo il pixel in alto a sinistra: $X' = W + N - NW$. Quindi si prende come stima il pixel più vicino alla stima iniziale (in alto, a sinistra o in alto a sinistra).

PNG codifica la differenza tra stima e valore effettivo tramite la compressione *DEFLATE*, che combina la codifica LZ77 e la codifica Huffman.

Una caratteristica dello standard PNG è la possibilità di utilizzare l'interlacciamento (*interlacing*), una tecnica di visualizzazione progressiva che consente di visualizzare una versione sempre più dettagliata dell'immagine, man mano che vengono ricevuti più dati.
Lo scopo dell'interlacciamento è quello di poter ricostruire progressivamente un'immagine compressa, senza dover aspettare l'intero flusso di bit, in particolare se si tratta di un'immagine di grandi dimensioni di cui è necessaria solo un'anteprima.

La tecnica di interlacing di PNG è nota come Adam7 e funziona dividendo l'immagine originale in sette sotto-immagini più piccole.
Ognuno dei sette passaggi è progettato per fornire progressivamente più dettagli e copertura dell'immagine, replicando lo schema 8×8 visibile in Figura 2.2 sull'intera immagine, a partire dall'angolo superiore sinistro.

```
1 6 4 6 2 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
```

Figura 2.2: I pixel scelti ad ogni passaggio, numerato da 1 a 7. Immagine presa da *Portable Network Graphics (PNG) Specification (Third Edition)* [12].

2.4 CALIC

Lo standard CALIC (Context-based, Adaptive, Lossless Image Codec) è stato presentato per la prima volta nel 1996 [13]. CALIC si basa sulla predizione dei valori dei pixel, ma anche sulla modellizzazione del contesto.

Nella fase di predizione, CALIC impiega uno schema di predizione non lineare basato sul gradiente, chiamato GAP (gradient-adjusted predictor), che regola i coefficienti di predizione in base alle stime dei gradienti locali.
L'obiettivo è cercare di predire, tramite l'utilizzo dei pixel causali, se è presente un "*edge*" orizzontale o verticale. Con *edge*, o bordo, si intende un cambiamento locale significativo nei valori dei pixel dell'immagine [14].

In Figura 2.3 si vedono i pixel utilizzati nella predizione di X, tutti disponibili anche dal decoder.

		NN	NNE
	NW	N	NE
WW	W	X	

Figura 2.3: Pixel utilizzabili per la predizione di X. Credit: *Introduction to Data Compression* [5].

Si calcola

$dh = |W{-}WW| + |N{-}NW| + |NE{-}N|$

$dv = |W{-}NW| + |N{-}NN| + |NE{-}NNE|$

Se il valore di dh è molto più alto del valore di dv, significa che è presente una grande quantità di variazione orizzontale (un bordo orizzontale) e viene scelto N come predizione iniziale.
Se, invece, dv è molto più grande di dh, si ha una predominante variazione verticale e la predizione iniziale è data da W.
Se le differenze sono più moderate o più piccole, il valore predetto è una media ponderata dei pixel vicini.

Le prestazioni del predittore GAP possono essere migliorate in modo significativo attraverso la modellazione del contesto, poiché i gradienti da soli possono non essere in grado di caratterizzare adeguatamente alcune delle relazioni più complesse tra il pixel X e la sua predizione $\hat{X}$.

Inizialmente viene quantificata l'informazione dei pixel causali vicini, formando un vettore che ha 144 possibili configurazioni.
Successivamente si calcola una quantità che incorpora le variazioni verticali e orizzontali e il precedente errore di predizione ($\hat{N}$ è il valore predetto di N):

$\delta = dh + dv + 2N - \hat{N}$

L'insieme di valori di δ è suddiviso in quattro intervalli. Queste quattro possibilità, insieme ai 144 descrittori di texture, creano $144 \times 4 = 576$ contesti per X.

Mentre la codifica procede, la predizione iniziale viene compensata con una quantità dipendente dal contesto. Inoltre viene tenuto traccia dell'errore di predizione: per questo CALIC è *Adaptive*: permette di modificare l'offset dato dal contesto in base ai valori degli errori precedenti.

Si ottiene così il valore finale predetto. L'errore di predizione viene codificato utilizzando o la codifica di Huffmann o la codifica aritmetica [9].

2.5 JPEG-LS

JPEG-LS è nato per essere particolarmente adatto alle implementazioni hardware di complessità moderata, fornendo allo stesso tempo prestazioni di compressione lossless competitive [15].

Lo standard JPEG-LS è simile a CALIC, ma utilizza un modello predittivo più semplice per diminuire il tempo di codifica e di decodifica: il Median Edge Detection (MED) [1].
Questo approccio di predizione si basa sull'analisi dei pixel N, W e NW.

Se è presente una forte correlazione nella direzione orizzontale, allora vale come approssimazione
$X - W = N - NW$
Quindi si sceglie come predizione del valore del pixel:
$\hat{X} = W + N - NW$

Similmente, se si riscontra una forte correlazione nella direzione verticale vale
$X - N = W - NW$
che porta alla stessa predizione
$\hat{X} = W + N - NW$.

La presenza di un bordo verticale implica elevata correlazione tra pixel lungo la stessa linea verticale, quindi viene scelta la predizione
$\hat{X} = N$

mentre in caso di bordo orizzontale la predizione è
$\hat{X} = W$

Per determinare la presenza di un bordo non si effettua, come in CALIC, il calcolo di dh e dv, ma un procedimento più semplice descritto dall'algoritmo in Figura 2.4.

$$
\begin{aligned}
&\text{if } NW \geqslant \max(W, N)\\
&\widehat{X} = \max(W, N)\\
&\text{else}\\
&\{\\
&\quad \text{if } NW \leqslant \min(W, N)\\
&\quad \widehat{X} = \min(W, N)\\
&\quad \text{else}\\
&\quad \widehat{X} = W + N - NW\\
&\}
\end{aligned}
$$

Figura 2.4: Algoritmo utilizzato nel Median Edge Detection (MED) predictor. Credit: *Introduction to Data Compression* [5].

La predizione iniziale viene poi raffinata utilizzando il valore medio dell'errore in quel particolare contesto. Come in CALIC, anche in JPEG-LS i contesti riflettono le variazioni locali dei valori dei pixel. Tuttavia, sono calcolati in modo diverso da CALIC [5].
In primo luogo, le misure delle differenze D1, D2 e D3 sono calcolate come segue:
$D1 = NE - N$
$D2 = N - NW$
$D3 = NW - W$
I valori di queste differenze definiscono un vettore di contesto a tre componenti **Q**.
Le componenti di **Q** (D1, D2 e D3) sono mappate su 9 possibili valori, ottenendo 9 × 9 × 9 = 729 contesti possibili.
L'errore di predizione viene mappato in un intervallo di dimensioni pari a quelle dell'intervallo occupato dai valori dei pixel originali e quindi codificato.

Nella Tabella 2.3 tratta da *Introduction to Data Compression* [5], vengono confrontate le dimensioni in byte delle immagini compresse usando il vecchio standard JPEG, JPEG-LS e CALIC.
Si può notare che, per la maggior parte delle immagini, il nuovo standard JPEG ha prestazioni molto vicine a CALIC e supera il vecchio standard del $6\% - 18\%$.
Sebbene il miglioramento delle prestazioni in questi esempi possa non sembrare altamente significativo, bisogna considerare che viene scelto il miglior risultato su otto per il vecchio JPEG. Questo significa provare tutti gli otto predittori JPEG e scegliere il migliore. D'altra parte, sia CALIC che il nuovo standard JPEG sono algoritmi singlepass, necessitano cioè di una singola scansione.

Image	Old JPEG	New JPEG	CALIC
Sena	31,055	27,339	26,433
Sensin	32,429	30,344	29,213
Earth	32,137	26,088	25,280
Omaha	48,818	50,765	48,249

Tabella 2.3: Confronto delle dimensioni dei file ottenute utilizzando il vecchio standard JPEG, JPEG-LS e CALIC.

2.6 WebP

WebP è stato sviluppato da Google nel 2010. Il suo scopo è quello di comprimere immagini occupando meno spazio di archiviazione rispetto ai formati JPEG, PNG o GIF.
Lo standard è nato per essere utilizzato per la compressione di immagini sul Web ed è supportato da molti browser, tra cui Google Chrome, Microsoft Edge, Mozilla Firefox, Opera [16] [17].

WebP supporta il canale colore a 24 bit e il canale alfa a 8 bit con compressioni sia lossy che lossless.

In WebP esistono 13 diverse modalità di predizione possibili. Quelli prevalenti sono i pixel di sinistra, in alto, in alto a sinistra e in alto a destra. Le rimanenti sono combinazioni lineari di sinistra, in alto, in alto a sinistra e in alto a destra.

La codifica WebP-lossless comprime le immagini usando non solo la predizione spaziale, ma anche una trasformazione dello spazio colore, la pacchettizzazione di più pixel in uno e una cache dei colori, per utilizzare frammenti di immagine già visti per ricostruire nuovi pixel [18].
L'obiettivo della trasformazione del colore è la de-correlazione dei valori R, G e B di ogni pixel. La trasformazione del colore mantiene il valore verde (G) così com'è, trasforma il rosso (R) in base al verde e trasforma il blu (B) in base al verde e poi al rosso.
Per la codifica entropica viene utilizzata una variante della codifica LZ77-Huffman.

Nello studio *Comparative performance analysis of web image compression* [16] sono state misurate e comparate, su sei immagini di test, le performance di PNG e WebP.
La Tabella 2.4 mostra il rapporto di compressione dei due formati. WebP ha un miglioramento del CR di almeno il 10% rispetto a PNG e l'aumento medio della compressione è del 25%.

Tests images [Resolution]	**Compression Ratio**		
	PNG	***WebP lossless***	***CR Increase***
Wiki012 [1202 x 804]	1.34	1.69	26%
baboon [512 x 512]	1.23	1.36	10%
Kodim021 [768 x 512]	1.85	2.40	29%
Kodim14 [768 x 512]	1.70	2.28	34%
Kodim15[768 x 512]	1.93	2.70	40%
peppers [512 x 384]	2.05	2.28	11%

Tabella 2.4: CR delle immagini di test [16].

Anche il tempo della codifica e della decodifica di PNG e WebP è stato calcolato sulle immagini di prova e mostrato in Figura 2.5.

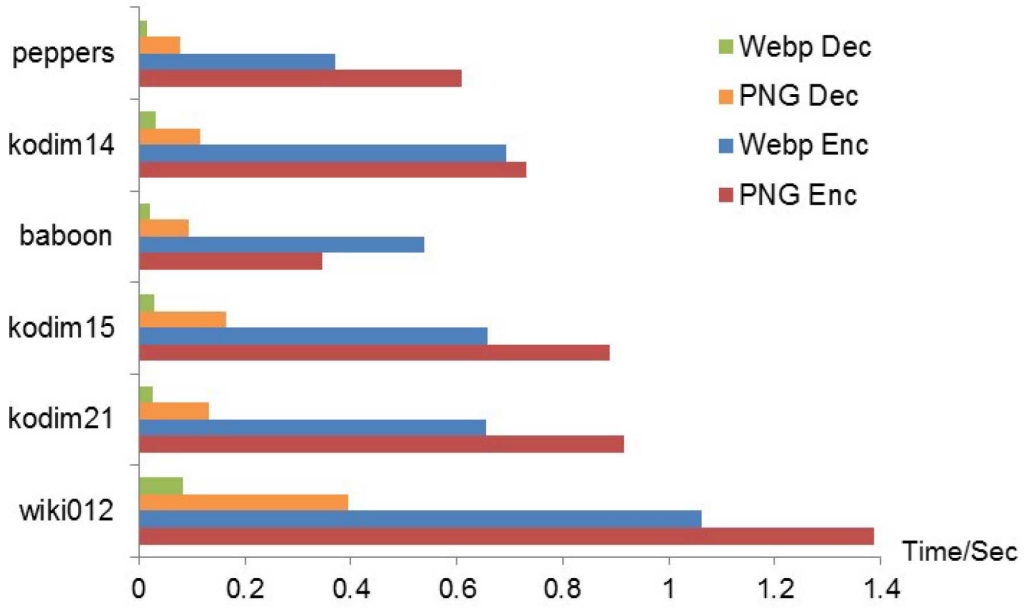


Figura 2.5: Tempo di codifica e decodifica di PNG e WebP [16].

Si può notare che i due algoritmi hanno una velocità di codifica simile, con una superiorità media di WebP del 10%.

Per quanto riguarda la decodifica, secondo questo studio WebP impiega solo il 20% del tempo impiegato da PNG per decodificare la stessa immagine, il che significa che la velocità di decodifica di WebP è 5 volte superiore a quella di PNG.

Anche altri studi, analizzati alla fine del capitolo successivo, confermano che WebP offre una compressione superiore rispetto a PNG e in generale una ottima compressione.
Tuttavia, tali studi smentiscono gli ottimi risultati di WebP in termini di tempo di decodifica. In essi WebP ha risultati inferiori rispetto a PNG, il formato migliore sotto questo aspetto.

Alcuni svantaggi dello standard WebP sono la risoluzione massima di 16383×16383 [19] e il mancato supporto del progressive decoding [9].

Capitolo 3

State of the art

In questo capitolo vengono descritti in breve alcuni degli standard emergenti per la codifica di immagini lossless, evidenziando eventuali particolarità. Ci si sofferma in particolare sul funzionamento del formato FLIF, l'unico sviluppato per essere esclusivamente lossless.
Nell'ultima sezione sono analizzati vari studi comparativi tra i vari standard, per valutare le differenze nel rapporto di compressione e i tempi di codifica e decodifica su varie immagini di test.

3.1 BPG

Better Portable Graphics (BPG) è un formato di file per la codifica di immagini digitali, creato dal programmatore Fabrice Bellard nel 2014 come sostituto del formato immagine JPEG.

Il formato è stato progettato per funzionare in ambienti con poca memoria e per essere utilizzato in dispositivi portatili e nell'IoT.
Sebbene non esista un supporto nativo integrato per il BPG in nessuno dei principali browser, i siti web possono comunque utilizzare immagini BPG includendo una libreria JavaScript scritta da Bellard [20].

BPG si basa sulla codifica intra-frame dello standard di compressione video High Efficiency Video Coding (HEVC) [21] e permette una compressione sia lossy che lossless [22].
In seguito è presente un riassunto del funzionamento della codifica intra-frame di HEVC, basato su *Intra compression efficiency in VP9 and HEVC* [23] e *HEVC still image coding and high efficiency image file format* [24].

HEVC adotta una struttura di codifica ibrida e include sia unità di predizione che unità di trasformazione.
Il frame di un video è inizialmente partizionato in blocchi. La predizione viene effettuata su

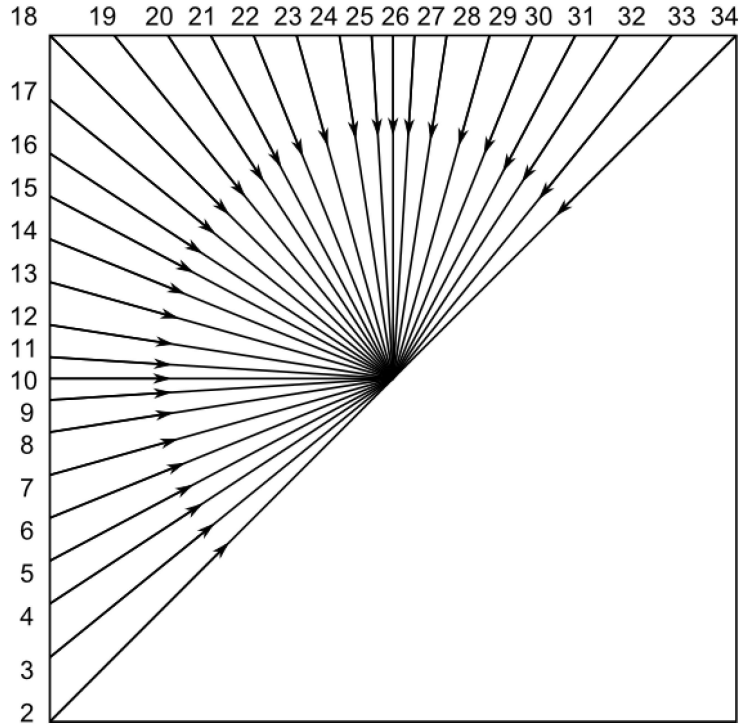


Figura 3.1: Le direzioni delle 33 modalità di predizione Angular in HEVC. Credit: *Intra compression efficiency in VP9 and HEVC* [23]

ciascun blocco basandosi sui pixel nei blocchi vicini disponibili, in 35 modalità differenti: DC, Planar e 33 diverse modalità Angular.

La predizione DC è la più semplice: tutti i pixel dell'unità di predizione sono impostati uguali alla media dei valori dei pixel a sinistra e in alto del blocco.

La predizione Planar è la più dispendiosa computazionalmente. Essa è una interpolazione lineare a due dimensioni.

Le modalità Angular sono interpolazioni lineari effettuate in 33 direzioni diverse, visibili in Figura 3.1.

Una volta effettuate le predizioni, queste ultime sono sottratte dal valore effettivo dei pixel. I residui sono ulteriormente compressi tramite Forward Discrete-Fourier Transform e quantizzazione.

La codifica entropica utilizzata da HEVC è chiamata "*syntax-based context-adaptive binary arithmetic coder*" (SBAC) e ha 27 modelli di contesto che cambiano basandosi sui dati già processati.

BPG supporta il progressive decoding, le animazioni e gli spazi di colore grayscale, RGB, YCgCo, con un canale alfa per la trasparenza. Ogni componente può avere una color depth da 8 a 14 bit [25].

HEVC è soggetto a numerosi brevetti che limitano la diffusione di BPG, in quanto gli sviluppatori e i fornitori di System on chip (SoC) devono pagare per l'implementazione di algoritmi brevettati.

3.2 FLIF

Free Lossless Image Format (FLIF) è un formato di immagini lossless annunciato nel settembre 2015 e rilasciato un anno dopo.
Uno dei principali vantaggi di FLIF è che, al contrario di altri standard di compressione lossless, ha un ottimo rapporto di compressione per ogni tipo di immagine, anche non fotografica. Questo permette, secondo gli sviluppatori, di creare un formato universale utilizzabile da chiunque e per qualunque scopo [26].
Lo sviluppo di FLIF è stato interrotto, in quanto FLIF è stato sostituito da FUIF e poi ancora da JPEG XL, che si basa su una combinazione di Pik e FUIF [27].

Alcune particolarità di FLIF sono:

- trasformazione reversibile del colore per aumentare il rapporto di compressione;
- supporto del progressive decoding usando l'interlacing Adam∞, una versione migliorata di Adam7, usato da PNG [9];
- supporto di grayscale, RGB and RGBA (con il canale alfa), con una color depth da 1 a 16 bit per canale, supporto di animazioni [28];
- utilizzo di MANIAC, un algoritmo per la codifica entropica basato sul machine learning.

3.2.1 Trasformazione dei colori

FLIF utilizza la trasformazione del colore YCoCg, inoltre supporta tavolozze di dimensioni arbitrarie e palette a 4 canali (YCoCgA) e a 3 canali (YCoCg), codificando il canale alfa separatamente se necessario.

Il modello di colore YCoCg è lo spazio colore formato da una trasformazione di uno spazio colore RGB associato in un valore di luminanza (indicato come Y), che rappresenta la quantità di luce dell'immagine, e due valori chiamati crominanza verde (Cg) e crominanza arancio (Co), rispettivamente la differenza tra rosso e verde e e la[29].
Le formule di trasformazione da RGB a YCoCg sono le seguenti:
$Y = (R + 2G + B)/4$
$Co = (R - B)/2$
$Cg = (2G - R - B)/4$

In genere, un'immagine utilizza solo un frammento dell'intero spazio colore. Si può tenere conto di questo aspetto per migliorare la compressione, utilizzando palette di colori.
L'approccio basato su palette funziona se è utilizzata solo una piccola parte dello spazio di colori, ad esempio 2^{10} su 2^{24} o 2^{32} colori [26]. Il meccanismo proposto dagli sviluppatori di FLIF, chiamato *color buckets*, generalizza l'idea delle palette di colori. Funziona come segue.
Per ogni valore di Y, viene tenuta traccia dei valori di Co corrispondenti. Finché il loro numero è piccolo, viene mantenuto un elenco di valori discreti (un "*discrete bucket*"); se questo elenco diventa troppo grande (secondo una soglia arbitraria), viene sostituito da un intervallo, memorizzando solo i limiti inferiore e superiore (un "*continuous bucket*"). Successivamente, per ogni combinazione di Y e Co, si registrano i valori di Cg in modo analogo.
Per mantenere il numero totale di bucket ragionevole, viene utilizzata la quantizzazione e il meccanismo viene disattivato per le immagini ad alta profondità di bit.

3.2.2 Scansione dell'immagine e predizione dei pixel

FLIF permette due metodi per scannerizzare una immagine [26].

Il primo metodo è il classico scan row-major, riga dopo riga e da sinistra a destra. Facendo riferimento alla notazione visibile in Figura 3.2, la predizione di ogni pixel è data dalla mediana di T, L e $T + L - TL$.

Il secondo metodo è una generalizzazione dell'interlacciamento Adam7 presente in PNG, chiamato Adam∞. Invece di avere un numero fisso di passaggi, Adam∞ utilizza un numero teoricamente infinito di passaggi. L'idea è di aggiungere continuamente dettagli all'immagine, migliorandola in modo uniforme e progressivo. In ogni fase di interlacciamento, il numero di pixel raddoppia. Il primo passo è semplicemente un pixel: il pixel nell'angolo in alto a sinistra. Poi, in ogni passo di interlacciamento, il numero di righe raddoppia (step orizzontale) o il numero di colonne raddoppia (step verticale). Il passo finale è sempre uno step orizzontale, che attraversa tutte le righe dispari dell'immagine.

		$\boldsymbol{TT}$	
	$\boldsymbol{TL}$	$\boldsymbol{T}$	$\boldsymbol{TR}$
$\boldsymbol{LL}$	$\boldsymbol{L}$	?	R
	$\boldsymbol{BL}$	B	$\boldsymbol{BR}$

Figura 3.2: Pixel attorno al pixel ignoto "?". Credit: *FLIF: Free lossless image format based on MANIAC compression* [26]

I pixel indicati in grassetto nella Figura 3.2 sono noti al momento della decodifica. In uno step orizzontale, è noto anche B, mentre in uno step verticale è noto R. Sono definiti tre predittori:

- la media $\frac{T+B}{2}$, se è uno step orizzontale, o la media $\frac{L+R}{2}$;
- la mediana dei tre pixel vicini noti (L, T, B o R);
- la mediana di: la media indicata sopra, il gradiente $T + L{-}TL$, il gradiente $L + B{-}BL$ o $T + R{-}TR$ (in base al tipo di step).

In entrambi i metodi sono codificate le differenze tra i valori dei pixel predetti e i valori effettivi.

Può sembrare che il loro intervallo sia doppio rispetto a quello dei valori originali, tuttavia, solo la metà di questa gamma è valida. Inoltre, non tutte le triple YCoCg corrispondono a triple RGB valide, riducendo ulteriormente l'intervallo. Anche i color bucket vengono usati per ridurre l'intervallo e per far coincidere il valore indovinato con un valore valido: se il color bucket è discreto, la predizione viene arrotondata al valore più vicino dell'elenco.

3.2.3 Entropy coding e contesto: MANIAC

FLIF utilizza per la codifica entropica MANIAC, Meta-Adaptive Near-zero Integer Arithmetic Coding, dove il modello stesso del contesto è adattivo: la predizione iniziale viene affinata dal contesto e successivamente codificata utilizzando la codifica aritmetica.

In FLIF il contesto è dato principalmente da differenze tra i pixel vicini al pixel da predire, ma in base alla scansione il contesto è diverso e comprende anche valori dei pixel vicini e la mediana [26].

È difficile trovare il numero di contesti giusto [26]: l'uso di un numero eccessivo danneggia la compressione perché l'adattamento al contesto è limitato (pochi pixel per contesto); ma anche con un numero insufficiente di contesti la compressione ne risente, perché pixel con proprietà diverse finiscono nello stesso. Inoltre, quando i contesti sono definiti in modo statico, molti di essi non vengono utilizzati affatto, poiché la corrispondente combinazione di proprietà non è presente nell'immagine di partenza.

Basandosi su queste considerazioni, gli sviluppatori di FLIF propongono una struttura dati dinamica come modello di contesto. Si tratta di un albero decisionale (in realtà un albero per canale), che cresce durante la codifica e permette un modello di contesto specifico per l'immagine da comprimere.

Ogni nodo interno ha una condizione di test, una disuguaglianza riguardante una delle proprietà del contesto. I nodi figli corrispondono alle due opzioni.

Ogni contesto attraversa l'albero fino a raggiungere una foglia. Ogni foglia contiene un contatore che decresce: a zero, la foglia diventa un nodo interno e l'albero "cresce".

3.3 AVIF

AVIF è un formato di immagine aperto e non proprietario, introdotto a partire dal 2019, che comprende sia codifica lossy che lossless. Il termine AVIF è un acronimo di AV1 Image File Format e può essere descritto come la versione immagine del formato video AV1. Oggi quasi tutti i browser supportano AVIF [30].

Il formato AVIF è simile concettualmente al formato BPG. Esso utilizza per la modalità lossless uno schema di codifica basato sia sulla predizione che sulla trasformazione. L'immagine viene divisa in blocchi, e ogni blocco viene predetto dai valori dei pixel all'interno dei blocchi vicini. AV1 (e di conseguenza AVIF) consente come predizione l'uso di 56 modalità direzionali e 13 modalità non direzionali [31].

I residui vengono sottoposti a una trasformazione 2D e successivamente codificati utilizzando la codifica aritmetica. [32].

I dati dell'immagine sono memorizzati in un file AVIF come una sequenza di frame compressi codificati con il codec AV1. Ogni frame è memorizzato in un proprio box, chiamato *Compressed Image Item Box*, ed è accompagnato da un *decoder configuration record*, che fornisce informazioni sui parametri di codifica utilizzati per quello specifico frame.

Questa struttura modulare consente ai file AVIF di memorizzare più immagini, come una serie di miniature o diverse risoluzioni della stessa immagine, all'interno di un unico file [33].

Alcune funzionalità di AVIF sono:

- supporto di SDR, Wide Color Gamut (WCG) e High Dynamic Range (HDR) [34];
- supporto del *multi-channel*, incluso il canale alfa per la trasparenza [35];
- supporto di immagini con profondità di 8, 10, e 12-bit [36].

Sono presenti limiti che rendono il formato AVIF non ideale:

- le tecniche di codifica utilizzate dal codec AV1 comportano tempi di codifica più lunghi rispetto ad altri standard di compressione, come verrà mostrato nel dettaglio in seguito;
- non supporta il progressive rendering, nè esiste un codificatore in grado di implementarlo facilmente [37] [38];

- mancanza di retrocompatibilità: le vecchie versioni di Browser Web non supportano AVIF [37] [33], inoltre esistono ancora browser, seppur poco popolari, non compatibili con AVIF;

- necessità di più potenza della CPU per comprimere immagini rispetto ad altri formati [39].

3.4 JPEG XL

JPEG XL è un formato che supporta sia compressione con perdita di dati che compressione senza perdita di dati. Si basa sulle idee del formato PIK di Google e del formato FUIF, a sua volta basato su FLIF, ed è il più recente tra tutti i formati analizzati in questo lavoro: il sistema di codifica principale è stato ufficialmente standardizzato nel marzo 2022 [40].

L'obiettivo degli sviluppatori di JPEG XL è di creare uno standard duraturo e adatto per ogni utilizzo.

JPEG XL permette una color depth di 32 bit per canale, inoltre supporta il progressive decoding, il canale alfa, HDR e animazioni [41].

Una particolarità è che i file JPEG esistenti possono essere transcodificati senza perdite in file JPEG XL, riducendone significativamente le dimensioni [42]. Questi possono essere ricostruiti nello stesso identico file JPEG, garantendo la retrocompatibilità con le applicazioni precedenti. Ciò consente di passare senza problemi dalle piattaforme JPEG tradizionali al moderno JPEG XL.

JPEG XL è costituito da due modalità principali: la modalità VarDCT, utilizzata nella compressione lossy, e la modalità Modular, utilizzata nella compressione lossless.
L'architettura di JPEG XL è mostrata in Figura 3.3.

Entrambe le modalità eseguono fasi simili o identiche nel processo di compressione, ma la VarDCT utilizza fasi aggiuntive per eseguire una Discrete Cosine Transform (DCT) di dimensioni variabili. I coefficienti risultanti vengono poi quantizzati, con conseguente perdita di informazioni e quindi con perdita di dati [43].

La modalità modulare è responsabile di un'efficiente compressione lossless e combina, secondo gli sviluppatori, le migliori idee da FLIF e da lossless WebP [44].
Essa prevede trasformazione di colore, predizione e selezione del contesto.

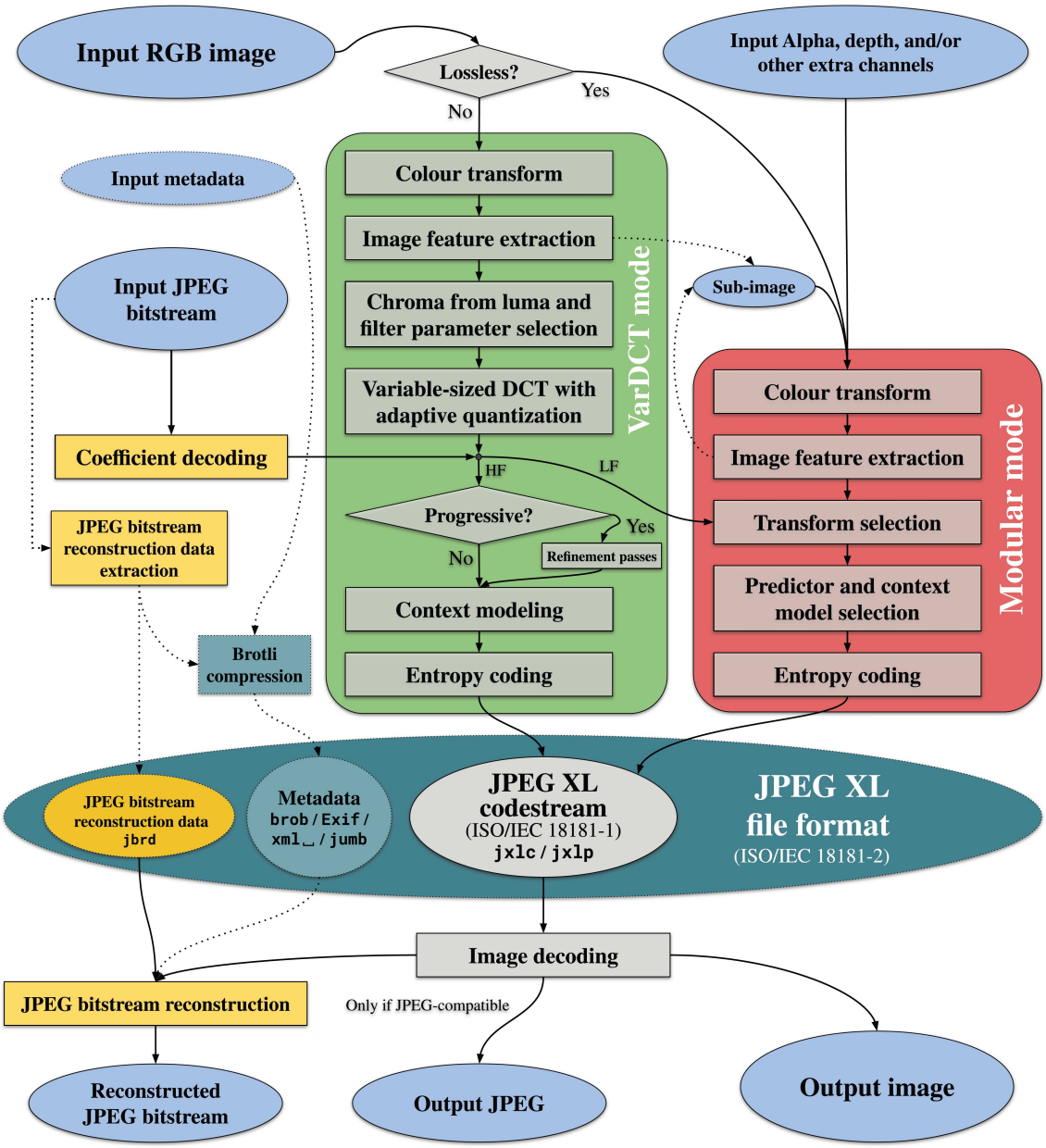


Figura 3.3: Architettura di JPEG XL. Credit: *JPEG XL overview* [44].

Lo spazio di colori di destinazione è chiamato "*XYB*" ed è basato sullo spazio di colori LMS (long, medium, short) che descrive la risposta dei tre tipi di coni dell'occhio umano.

Le varie predizioni di JPEG XL possono essere scelte in base al contesto e il contesto stesso è dato da una versione migliorata dell'albero decisionale utilizzato da FLIF.
In Tabella 3.1 sono indicati i predittori utilizzati da PNG, WebP, FFV1 (un formato di codifica intra-frame lossless), FLIF e JPEG XL.

JPEG XL utilizza "Asymmetric Numeral Systems", un codificatore basato sulla codifica arit-

Predictor	*PNG*	*Lossless WebP*	*FFV1*	*FLIF*	*JPEG XL*
Zero (none)	■	■			■
Left (W), Top (N)	■	■			■
LeftLeft (WW)					■
Avg(Left, Top)	■	■			■
Paeth	■				
Select		■			■
Gradient		■	■	(non-interlaced)	■
TopLeft (NW), TopRight (NE)		■			■
Avg(Left, TopLeft)		■			■
Avg(Top, TopLeft)		■			■
Avg(Top, TopRight)		■			■
Avg(Left, TopLeft, Top, TopRight)		■			
Avg(Top, Avg(Left, TopRight))		■			
Weighted sum of W, WW, N, NN, NE, NEE					■
Predictors involving Bottom				(interlaced)	
Self-correcting (Weighted) Predictor					■

Tabella 3.1: Predittori utilizzati da PNG, WebP, FFV1, FLIF e JPEG XL, colorati in verde. Credit: *JPEG XL overview* [44].

metica sviluppato recentemente, che raggiunge rapporti di compressione simili a quest'ultima, ma è più veloce nella fase di decodifica [43].

Al momento JPEG XL è poco supportato dai browser Web [45], ma diversi brand noti del settore hanno espresso pubblicamente il loro supporto per JPEG XL come opzione preferita, tra cui Facebook, Adobe e Intel [40].

3.5 Comparazione

In questo paragrafo sono riportati e analizzati alcuni studi che comparano le prestazioni dei vari standard di compressione lossless. Purtroppo non esistono studi che analizzano ogni formato citato in questo lavoro nelle stesse condizioni e con le stesse immagini di test. Questo rende difficile la valutazione del miglior rapporto di compressione e minore tempo di codifica e decodifica.

Lo studio *The Impact of State-of-the-Art Techniques for Lossless Still Image Compression* [9] è uno dei più completi e riguarda JPEG-LS, JPEG 2000, lossless JPEG, PNG, JPEG XR, CALIC, AVIF, WebP e FLIF.

JPEG 2000 è uno standard di compressione basato sulla discrete wavelet transform, sviluppato dal 1997 al 2000 con lo scopo di offrire più funzioni rispetto al classico JPEG.

JPEG XR è un formato rilasciato nel 2009 con lo scopo di raggiungere una ottima compressione e bassa complessità di codifica e decodifica [46].
Quest'ultimo utilizza una trasformata reversibile chiamata "lifting-based reversible hierarchical lapped biorthogonal transform" (LBT).

Gli autori hanno applicato le varie tecniche di compressione a quattro diverse tipologie di immagine: scale di grigio con profondità di 8-bit e 16-bit e immagini RGB da 8 e 16 bit. Per ogni tipologia sono state utilizzate le stesse nove immagini di test, prese da un database.
I risultati sono sintetizzati nelle tre figure successive.

La Figura 3.4 mostra il rapporto di compressione medio per ogni tipo di immagine. Si può vedere che FLIF fornisce il più alto CR per ogni tipologia, mentre il vecchio JPEG lossless ha il rapporto di compressione peggiore.
Tranne nelle immagini in scala di grigi a 8 bit, il secondo e il terzo standard migliore sono rispettivamente WebP e AVIF. Nelle immagini a 16 bit si vede nettamente il miglioramento tra gli standard recenti e quelli antecedenti.

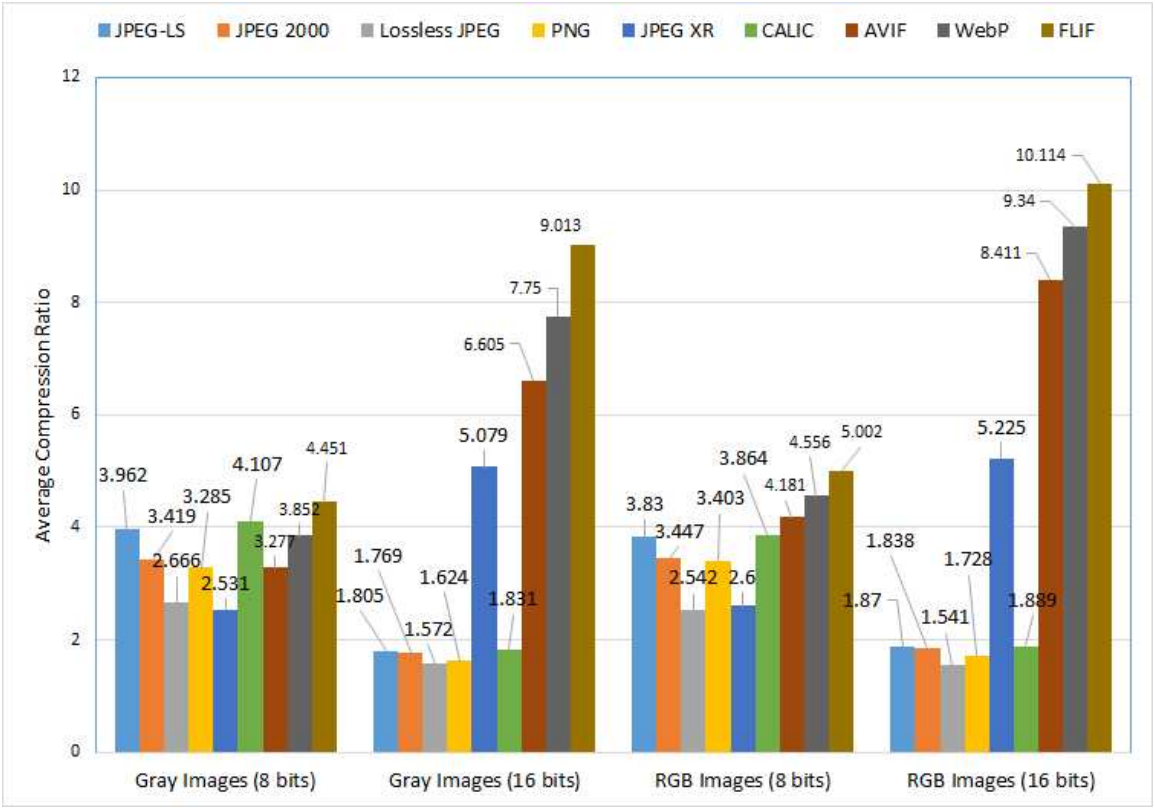


Figura 3.4: Comparazione del rapporto di compressione medio [9].

La Figura 3.5 mostra il tempo di codifica medio. Si può vedere come JPEG XR richieda il minor tempo per ogni tipologia di immagine, mentre FLIF sia nettamente il peggiore in questa metrica. In generale, gli standard più recenti tendono ad avere un tempo di codifica elevato, così come JPEG 2000, PNG e CALIC.

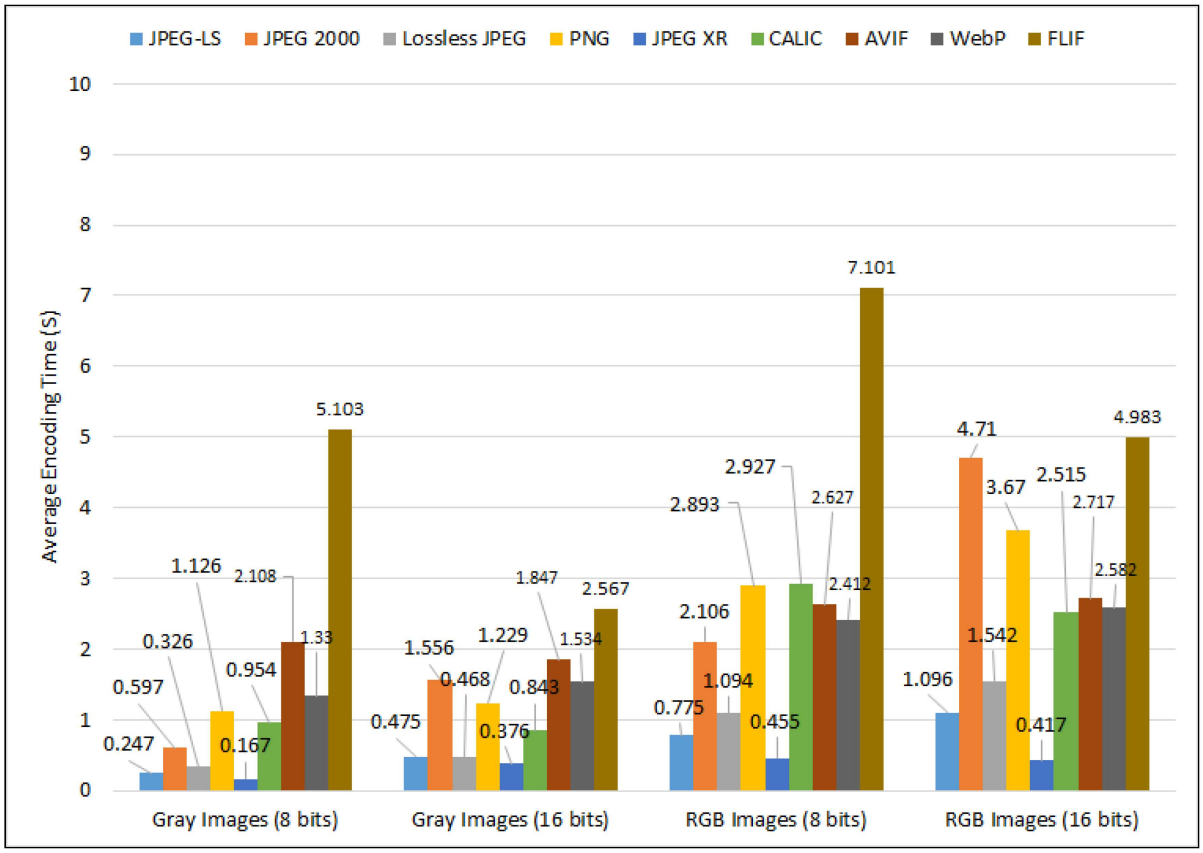


Figura 3.5: Comparazione del tempo di codifica medio [9].

Infine, in Figura 3.6 sono indicati i tempi di decodifica medi per ogni tipologia di immagine. PNG necessita del minor tempo di decodifica per tutte le tipologie tranne le immagini RGB a 16 bit, dove JPEG XR risulta migliore.
JPEG 2000 ha bisogno di tempi estremamente elevati, tranne nella prima tipologia.
Anche in questo caso i formati più recenti tendono a richiedere tempi maggiori, così come CALIC, standard noto per la sua complessità.

Questo studio evidenzia come i formati di compressione più recenti offrano una compressione significativamente superiore, ma senza miglioramenti nei tempi di elaborazione.
Se il tempo è un fattore critico, come nel caso delle applicazioni in tempo reale, formati più

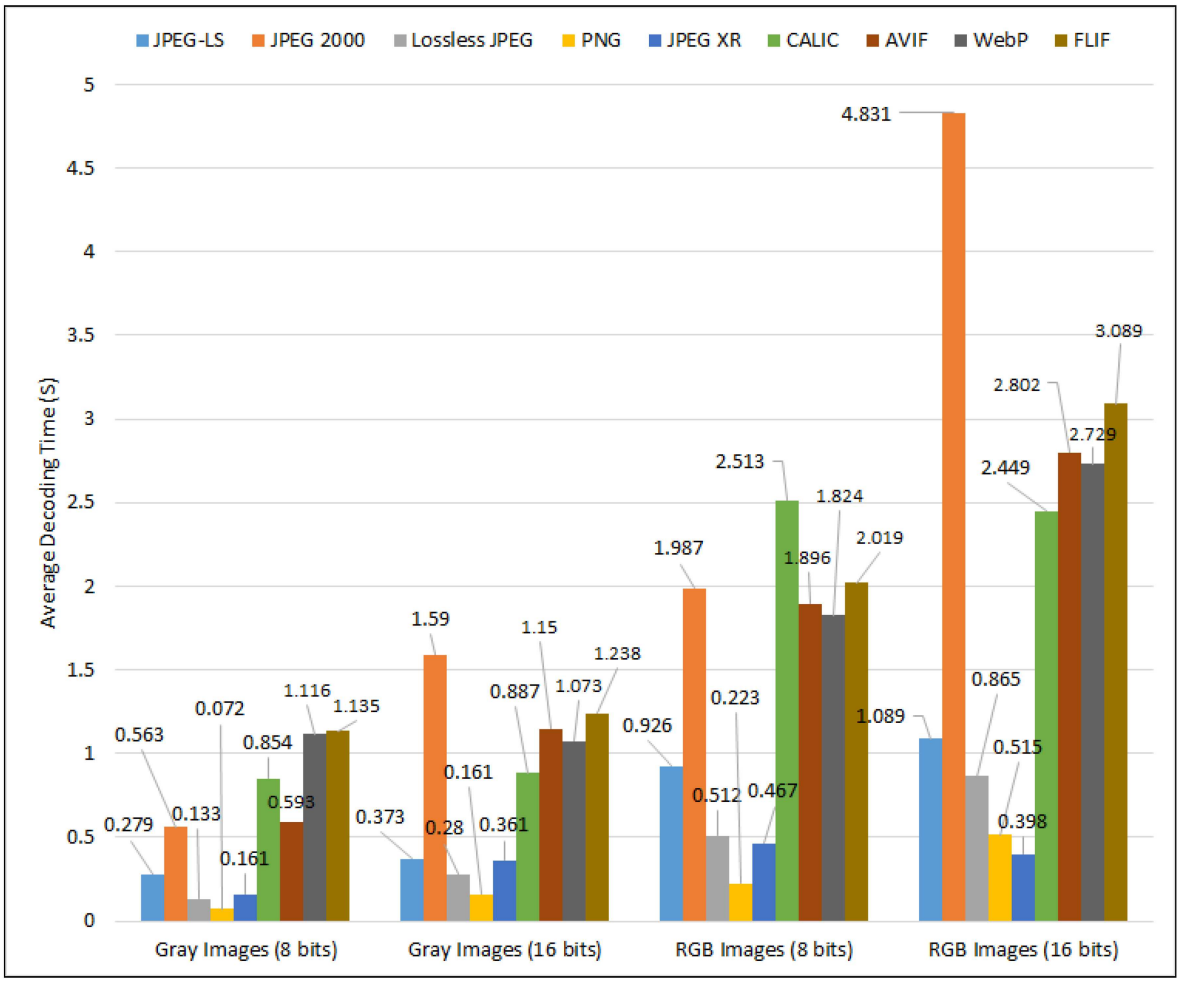


Figura 3.6: Comparazione del tempo di decodifica medio [9].

semplici come JPEG-LS e JPEG XR possono garantire prestazioni migliori, perchè in grado di codificare e decodificare una immagine in tempi nettamente minori.

Nel caso di archiviazione di immagini, dove il tempo necessario non è un fattore critico, la compressione lossless preferibile tra quelle analizzate è quella offerta da FLIF.

Anche nei test effettuati da Cloudinary in *FLIF, the new lossless image format that outperforms PNG, WebP and BPG* [47] FLIF risulta il vincitore, in questo caso tra i formati PNG, WebP e BPG.

Il test è stato eseguito su 200,000 immagini in formato PNG caricate sul servizio cloud dell'azienda. Come si può vedere in Figura 3.7, la dimensione media delle immagini convertite in FLIF è il 43% più piccola dei file PNG originali. WebP lossless è solo marginalmente inferiore a FLIF, mentre BPG e anche una versione ottimizzata di PNG offrono una compressione nettamente minore.

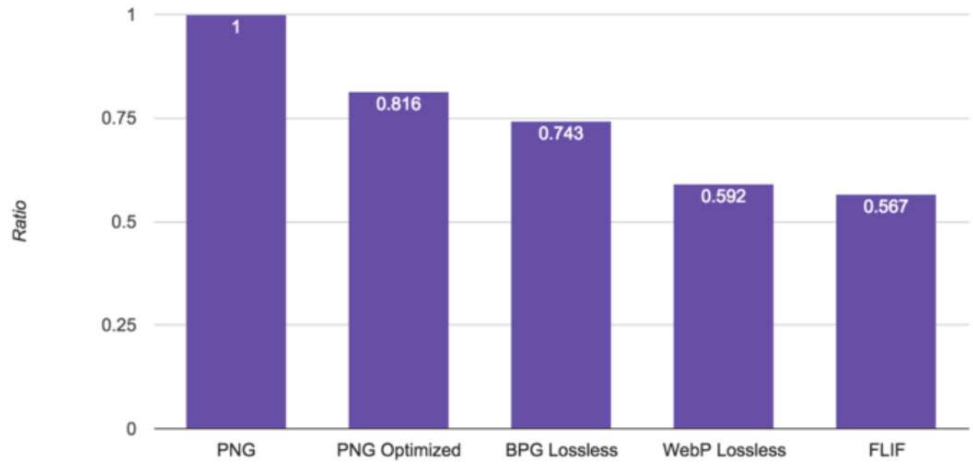


Figura 3.7: Dimensione dei file rapportata alla dimensione in formato PNG [47].

Lo studio *A Comparative Study on Lossless compression mode in WebP, Better Portable Graphics (BPG) and JPEG XL Image Compression Algorithms* [19] divide le immagini di test in due categorie:

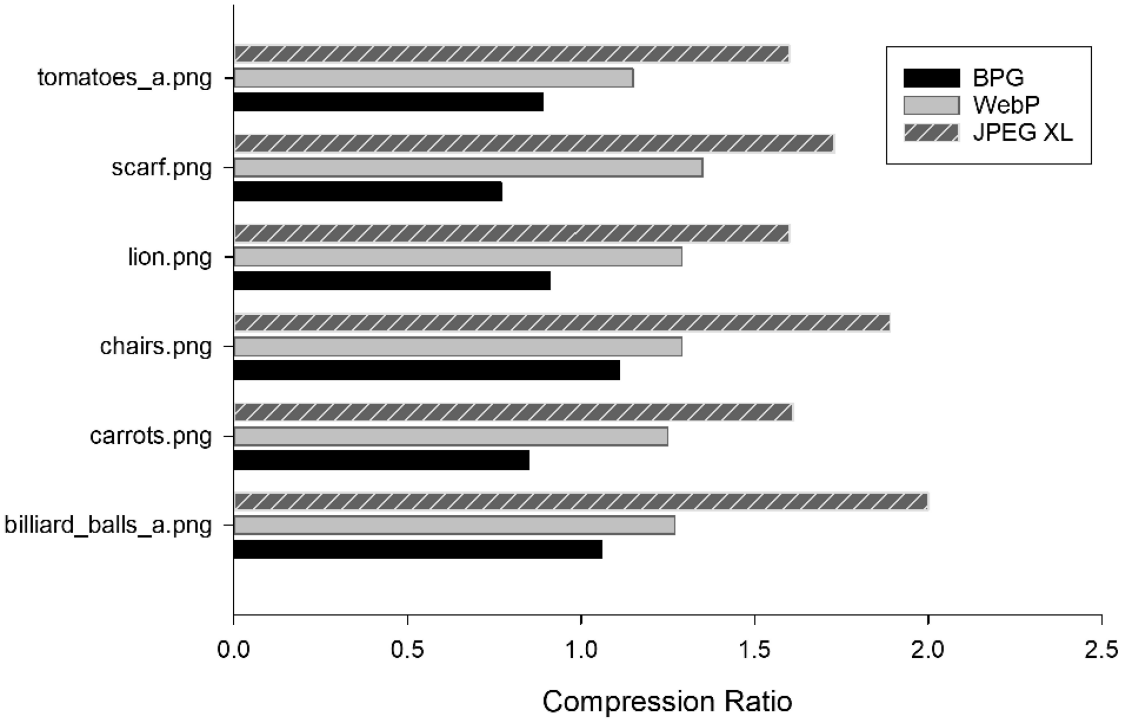
- *Sampling*, cioè immagini fotografiche, ognuna 2400x2400, spazio di colori RGB, 16 bit e HDR;
- *Pattern*, immagini in scala di grigio generate al computer che contengono semplici schemi geometrici.

Hanno inoltre creato tipologie di immagini diverse riducendo la risoluzione, cambiando lo spazio di colori in scala di grigio e la profondità di colore. Questo per misurare l'effetto di questi cambiamenti nella performance degli algoritmi.

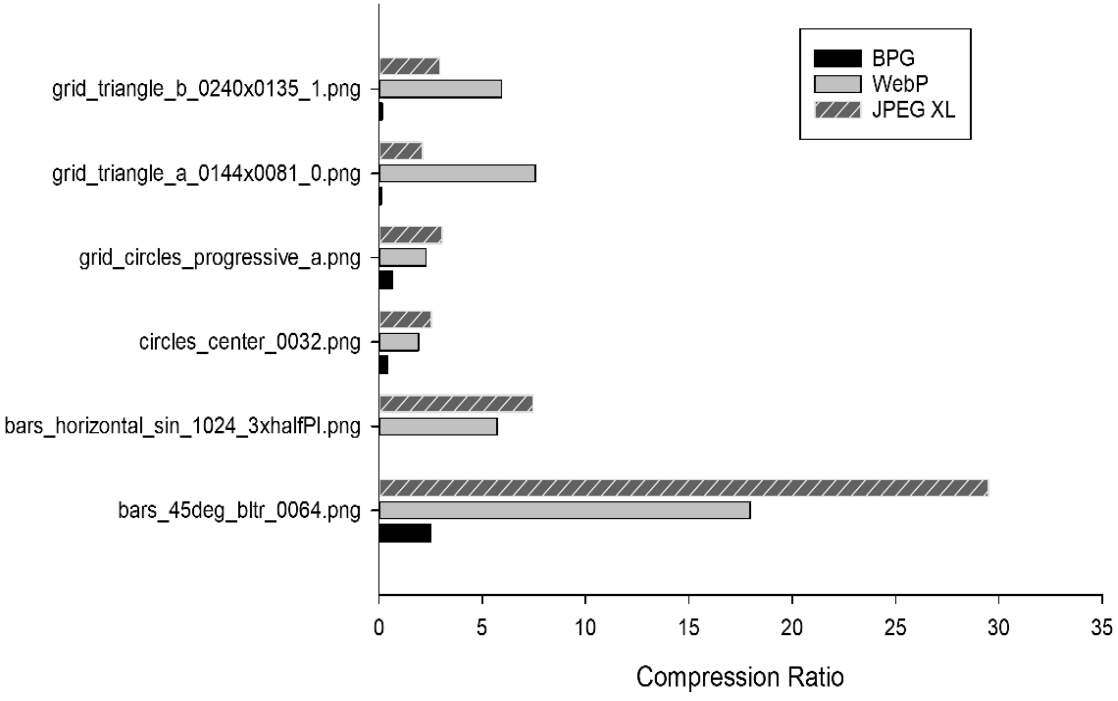
I risultati per la categoria *Sampling* sono visibili in Figura 3.8a, mentre i risultati per la categoria *Pattern* sono in Figura 3.8b.

Questa ricerca dimostra che JPEG XL produce i risultati migliori rispetto a BPG e WebP sia quando si comprimono immagini naturali, sia quando la compressione avviene su immagini sintetiche ad alta risoluzione.

Al contrario, il formato BPG ha mostrato scarse prestazioni in termini di CR in tutti i test. Questo era in parte prevedibile, in quanto BPG è stato progettato per essere poco complesso, sacrificando compressione per velocità e semplicità.



(a) Rapporto di compressione per immagini *Sampling* con dimensione 2400x2400 [19].



(b) Rapporto di compressione per immagini *Pattern* con dimensione 2880x1620 [19].

Figura 3.8

In *Performance Evaluation of JPEG Standards, WebP and PNG in Terms of Compression Ratio and Time for Lossless Encoding* [48] gli autori hanno analizzato sia il rapporto di compressione, sia il tempo di codifica e di decodifica. In questo caso il test è stato eseguito su immagini 8-bit SDR, 16-bit HDR e 16-bit SDR.

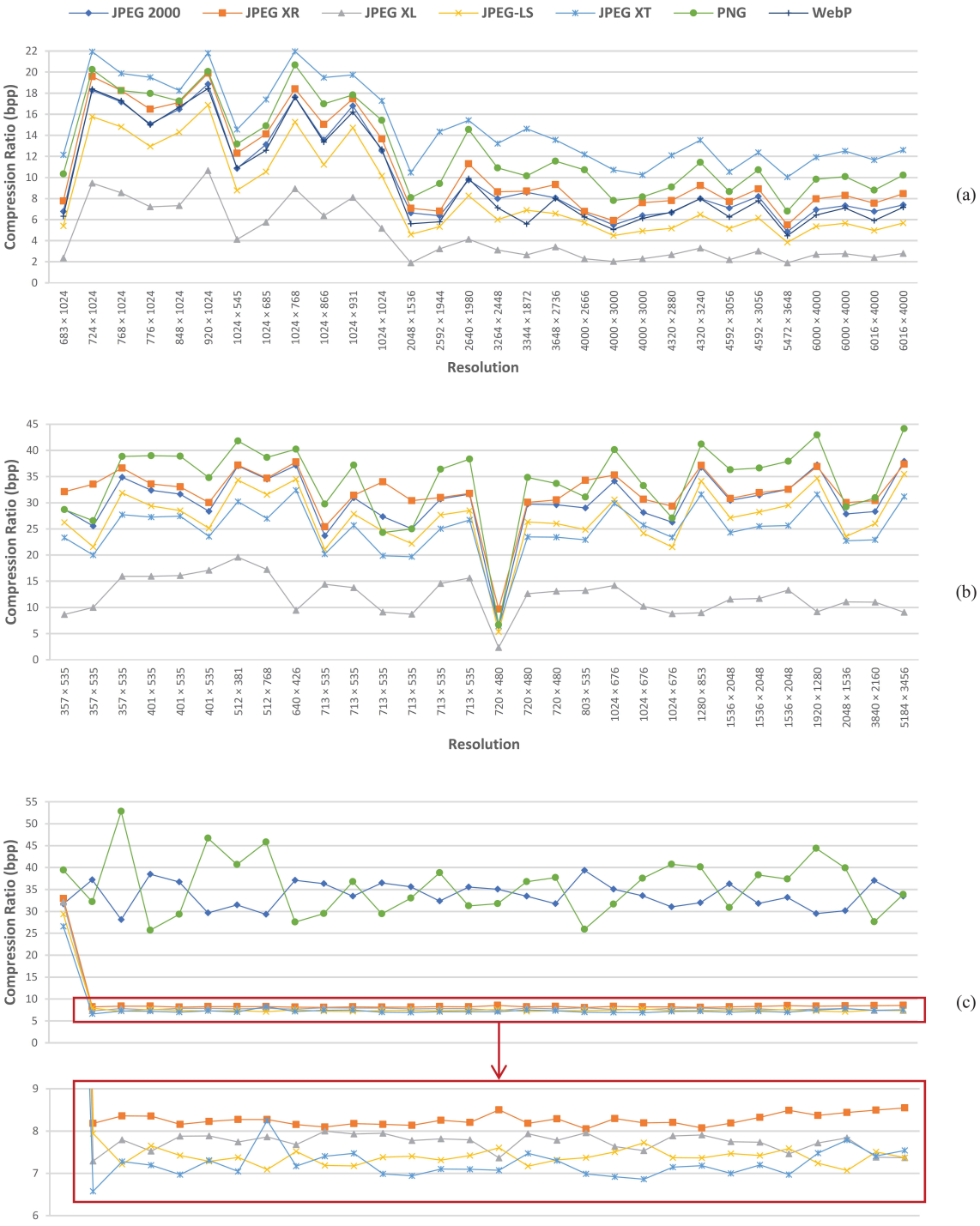


Figura 3.9: Bpp di immagini 8-bit SDR (a), 16-bit HDR (b) e 16-bit SDR (c) [48].

La Figura 3.9 mostra la compressione, espressa in bit per pixel, di immagini 8-bit SDR (a), 16-bit HDR (b) e 16-bit SDR (c) di diversa risoluzione.
Il bit per pixel (bpp) è una misura della compressione di immagini e rappresenta il numero medio di bit necessari per esprimere un pixel dell'immagine.
Più un formato riesce a diminuire i bpp, migliore è la compressione.

Nelle Figure 3.9 (a) e (b) si vede che il formato JPEG XL offre il miglior rapporto di compressione nelle immagini SDR a 8 bit e HDR a 16 bit. PNG e JPEG 2000 offrono una compressione scarsa nelle immagini SDR a 16 bit, mentre gli altri metodi hanno rapporti molto vicini tra loro. Poiché quando è stato effettuato lo studio il metodo WebP non era ancora in grado di codificare immagini a 16 bit, WebP non è stato incluso nei grafici di HDR e SDR a 16 bit.

Considerando tutte le immagini in generale, gli autori dello studio affermano che tra gli stadard analizzati JPEG XL ha il miglior rapporto di compressione e PNG il peggiore.

Come si vede in Figura 3.10, mentre JPEG XL impiega il tempo minore di codifica nelle immagini SDR a 8 bit, JPEG XR ottiene i risultati migliori nelle immagini SDR a 16 bit e HDR a 16 bit.

Quando la risoluzione supera 1 megapixel nelle immagini SDR a 8 bit, WebP, PNG e JPEG XR hanno risultati decisamente peggiori rispetto agli altri algoritmi. Nelle immagini HDR, i metodi basati su JPEG diversi da JPEG 2000 riescono a gestire meglio l'aumento della risoluzione.

Infine, la Figura 3.11 mostra i tempi di decodifica dei vari formati.
JPEG XR ha i migliori tempi di decodifica nelle immagini SDR a 8 bit, mentre PNG ottiene i risultati migliori nelle immagini SDR a 16 bit e HDR a 16 bit. A causa della complessità dell'implementazione, il JPEG 2000 impiega i tempi di decodifica maggiori in tutte le immagini.

In conclusione, JPEG XL non impiega tempi significativamente peggiori rispetto agli altri formati analizzati, come invece accade a FLIF nello studio già analizzato *The Impact of State-of-the-Art Techniques for Lossless Still Image Compression* [9].

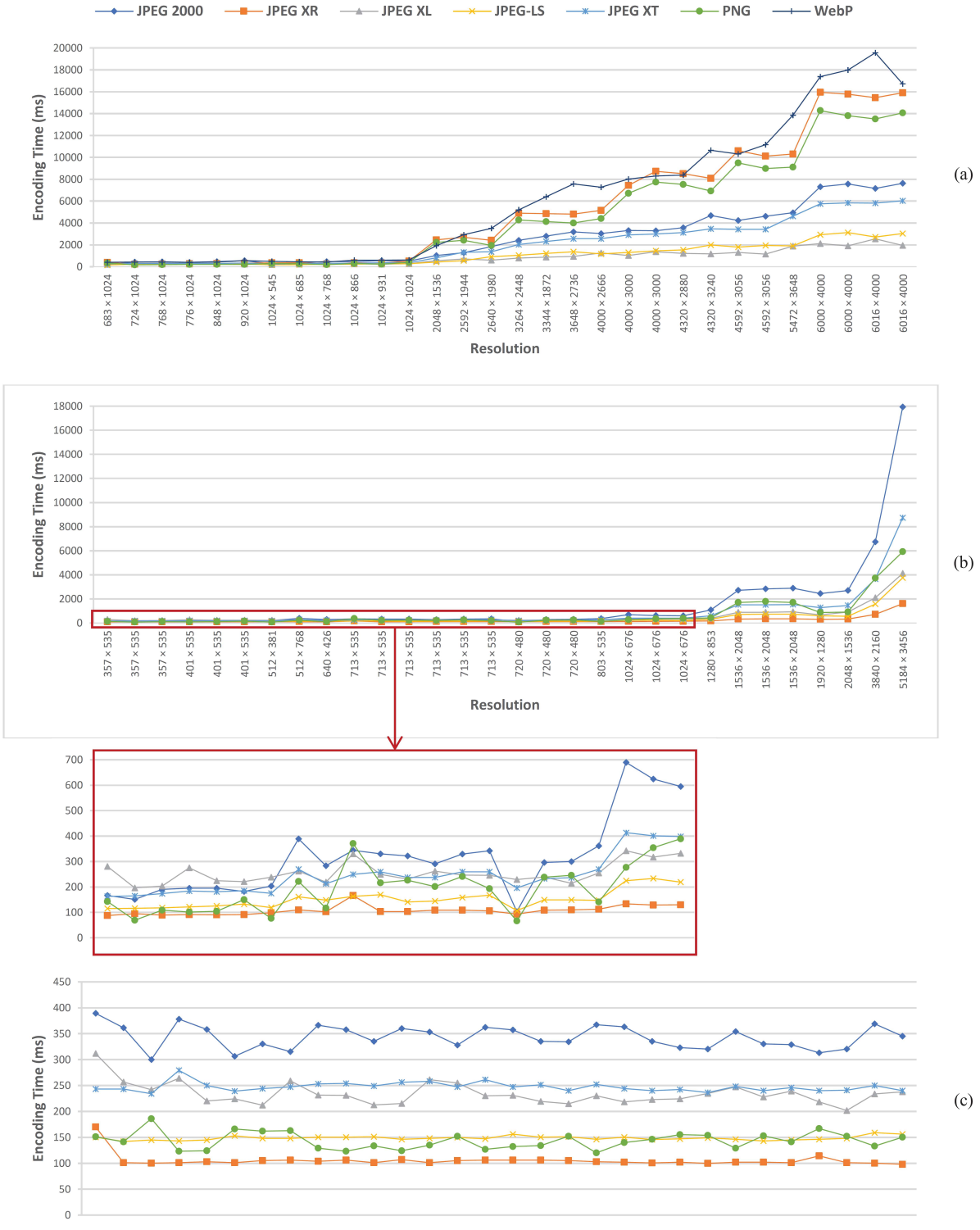


Figura 3.10: Tempo di codifica per immagini 8-bit SDR (a), 16-bit HDR (b) e 16-bit SDR (c) [48].

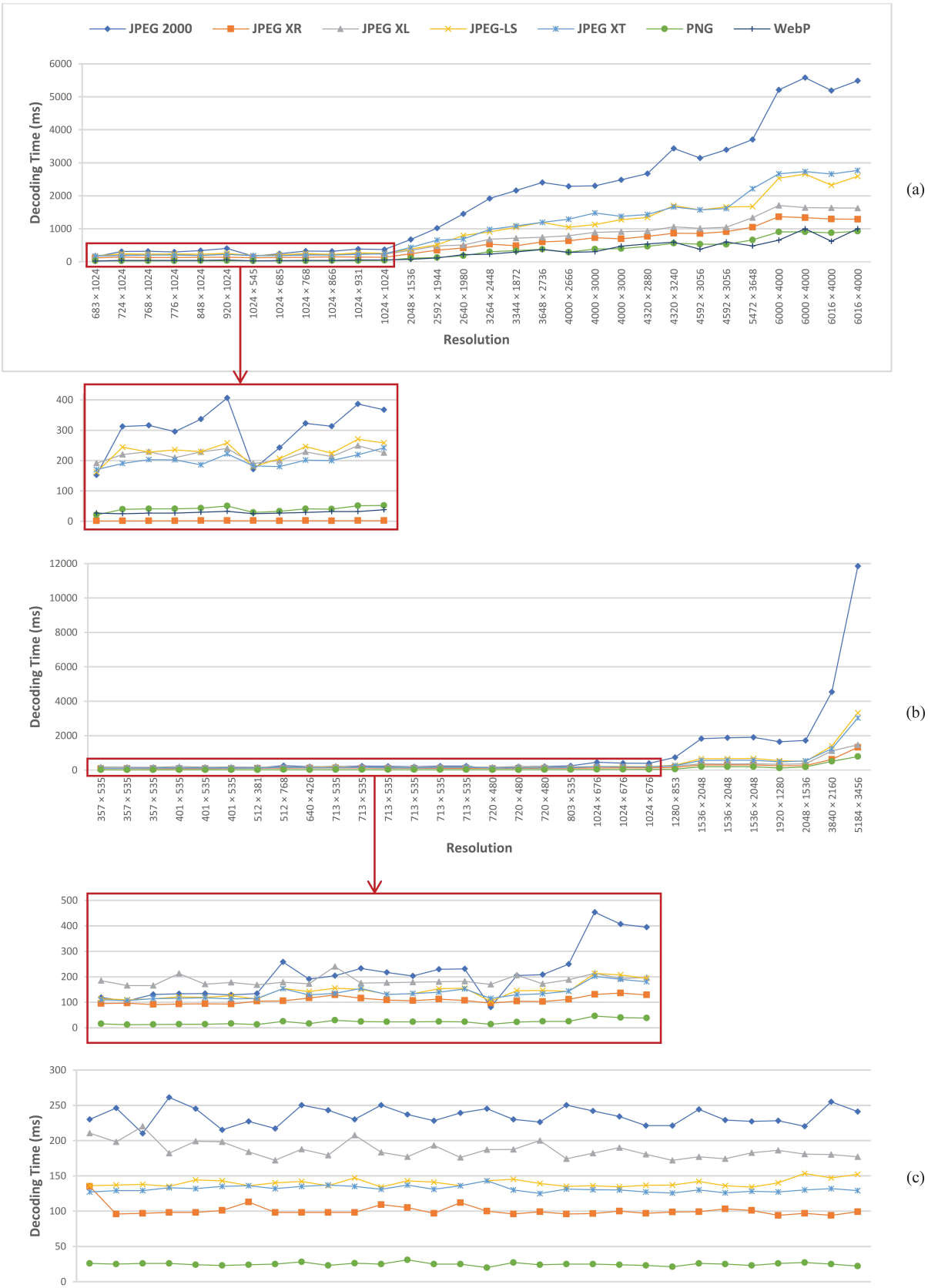


Figura 3.11: Tempo di decodifica per immagini 8-bit SDR (a), 16-bit HDR (b) e 16-bit SDR (c) [48].

Infine, lo studio *Comparison of Lossless Image Formats* [32] paragona i due formati che, dagli studi analizzati in precedenza, forniscono il miglior rapporto di compressione: FLIF e JPEG XL.

Il confronto viene effettuato su tre dataset. Il primo è composto da foto ad alta risoluzione ("*Photos*"), il secondo è costituito da immagini artificiali ("*Illustrations*"), mentre il terzo è dato da pagine scansionate di libri ("*Books*").

I rapporti di compressione e i rispettivi bpp di ogni standard sono visibili in Tabella 3.2.

Format	Bitrate [bpp]	Compression Ratio
PNG	11.461131	2.094034 : 1
JPEG-LS	10.588847	2.266535 : 1
JPEG 2000	10.620645	2.259749 : 1
JPEG XR	11.575239	2.073391 : 1
WebP	10.619910	2.259906 : 1
H.265	12.460111	1.926146 : 1
FLIF	**9.318886**	**2.575415** : 1
AVIF	12.039541	1.993431 : 1
WebP 2	10.007292	2.398251 : 1
JPEG XL	9.433458	2.544135 : 1

(a) Rapporto di compressione e bpp sul dataset *Photos*. Il risultato migliore è in grassetto.

Format	Bitrate [bpp]	Compression Ratio
PNG	4.628656	5.185090 : 1
JPEG-LS	6.994191	3.431419 : 1
JPEG 2000	6.094012	3.938292 : 1
JPEG XR	8.097638	2.963827 : 1
WebP	4.294325	5.588771 : 1
H.265	7.158238	3.352780 : 1
FLIF	**3.394439**	**7.070387** : 1
AVIF	8.198868	2.927233 : 1
WebP 2	3.621495	6.627097 : 1
JPEG XL	3.473148	6.910157 : 1

(b) Rapporto di compressione e bpp sul dataset *Illustrations*. Il risultato migliore è in grassetto.

Format	Bitrate [bpp]	Compression Ratio
PNG	8.694597	2.760334 : 1
JPEG-LS	8.394430	2.859038 : 1
JPEG 2000	7.303011	3.286315 : 1
JPEG XR	8.989639	2.669740 : 1
WebP	7.451733	3.220727 : 1
H.265	10.326125	2.324201 : 1
FLIF	**6.084763**	**3.944278** : 1
AVIF	10.398287	2.308072 : 1
WebP 2	6.890983	3.482812 : 1
JPEG XL	6.216347	3.860788 : 1

(c) Rapporto di compressione e bpp sul dataset *Books*. Il risultato migliore è in grassetto.

Tabella 3.2: Prestazione dei vari formati nei tre dataset [32].

I risultati di questo studio sono inaspettati: è emerso che il formato più efficiente è FLIF per ogni tipologia di immagine. Questo malgrado lo sviluppo di FLIF sia stato interrotto in favore di JPEG XL, il quale ha comunque un rapporto di compressione simile a FLIF.
La terza migliore scelta è WebP, mentre AVIF ha fornito in tutti i dataset una compressione particolarmente bassa.

Capitolo 4

Conclusioni

In questa tesi sono stati analizzati diversi formati di compressione lossless, partendo da quelli storici come GIF, il vecchio standard JPEG, PNG, CALIC e JPEG-LS, passando per quelli sviluppati negli anni 2000, tra cui JPEG 2000, JPEG XR e WebP, fino ad arrivare ai più recenti BPG, AVIF, FLIF e JPEG XL.

Dall'analisi emerge una tendenza nello sviluppo di soluzioni più universali e versatili, capaci di comprimere efficacemente qualsiasi tipo di immagine, sia SDR che HDR, con diverse profondità di colore e in grado di supportare canale alfa e animazioni.

Le valutazioni successive sono fatte ignorando il supporto o meno di varie funzionalità, includendo solo come parametri il rapporto di compressione e il tempo necessario per la codifica e la decodifica.

Come menzionato all'inizio del paragrafo precedente, non esistono studi che analizzino ogni formato nelle stesse condizioni; questo introduce inevitabilmente un margine di incertezza nelle considerazioni successive.

Tuttavia, sulla base degli studi precedentemente esaminati, si può affermare con ragionevole certezza che i nuovi formati JPEG XL e FLIF offrono un rapporto di compressione significativamente migliore rispetto agli altri standard. L'unico standard con una compressione paragonabile, anche se inferiore, è WebP, che tuttavia gode del vantaggio di essere già supportato dai principali browser.

Tale superiorità risulta particolarmente evidente se confrontata con il noto formato PNG, il quale si colloca tra i formati peggiori in termini di rapporto di compressione.

Secondo l'ultimo studio analizzato, *Comparison of Lossless Image Formats* [32], FLIF presenta un rapporto di compressione leggermente superiore rispetto a JPEG XL, ciò lo rende particolarmente indicato nelle applicazioni in cui la compressione efficiente è l'unico criterio determinante. Questo sebbene il suo sviluppo sia stato arrestato in favore proprio di JPEG XL.

Tra gli altri formati sviluppati negli ultimi anni, BPG ha dimostrato di avere una compressione modesta, mentre AVIF presenta risultati variabili: nello studio *The Impact of State-of-the-Art Techniques for Lossless Still Image Compression* [9] si colloca come terzo miglior formato in tre categorie, invece in *Comparison of Lossless Image Formats* [32] mostra una compressione pessima in tutti i dataset.

Se anche il tempo di elaborazione risulta un fattore determinante, la scelta diventa più complessa.

FLIF, WebP e AVIF richiedono tempi elevati sia nella fase di codifica che in quella di decodifica. FLIF permette in parte di ovviare a questo problema tramite l'interlacciamento e la conseguente possibilità di visionare anteprime sempre più accurate senza dover aspettare l'intero tempo di decodifica.

Al contrario, JPEG XL impiega tempi paragonabili a quelli di standard meno complessi e offre una compressione decisamente superiore rispetto a questi ultimi. Questo lo rende una scelta ottimale per bilanciare compressione e velocità.

Se il tempo è la metrica fondamentale, allora JPEG XR, un formato rilasciato nel 2009 con il preciso scopo di avere una bassa complessità di codifica e decodifica, può rappresentare la soluzione ideale. Esso è particolarmente indicato in caso di immagini RGB a 16 bit, dove emerge come il migliore sia in *The Impact of State-of-the-Art Techniques for Lossless Still Image Compression* [9] che in *Performance Evaluation of JPEG Standards, WebP and PNG in Terms of Compression Ratio and Time for Lossless Encoding* [48]. Sebbene la sua capacità di compressione non sia eccellente, si colloca comunque a un livello accettabile.

Il caso del formato PNG è interessante: oltre a offrire una scarsa compressione, richiede un tempo elevato per la codifica. Tuttavia, il suo tempo di decodifica è particolarmente basso, risultando spesso il più rapido. Pertanto, se il tempo di decodifica rappresenta l'unico criterio rilevante (come ad esempio nella trasmissione di immagini da un server ad un utente, in cui il tempo di codifica non è importante) PNG è la scelta migliore. Il tempo di decodifica può essere anche minore se interessa solo una anteprima dell'immagine, funzionalità ammessa da PNG tramite l'interlacciamento Adam7.

In conclusione, i formati di compressione più recenti FLIF e JPEG XL offrono maggiori funzionalità e garantiscono una compressione significativamente superiore rispetto agli standard precedenti. Sebbene FLIF garantisca una compressione leggermente migliore rispetto a JPEG XL, esso richiede tempi di elaborazione elevati sia in fase di codifica che di decodifica. Al contrario, JPEG XL si distingue per tempi di elaborazione competitivi, paragonabili ai migliori formati sotto questo aspetto.

Bibliografia

[1] J. Shukla, M. Alwani e A. K. Tiwari, «A survey on lossless image compression methods,» in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 6, 2010, pp. V6–136–V6–141. doi: 10.1109/ICCET.2010.5486344.

[2] L. F. R. Lucas, N. M. M. Rodrigues, L. A. da Silva Cruz e S. M. M. de Faria, «Lossless Compression of Medical Images Using 3-D Predictors,» *IEEE Transactions on Medical Imaging*, vol. 36, n. 11, pp. 2250–2260, 2017. doi: 10.1109/TMI.2017.2714640.

[3] S. Haddad, G. Coatrieux, M. Cozic e D. Bouslimi, «Joint watermarking and lossless JPEG-LS compression for medical image security,» in *Proceedings of the International Conference on Watermarking and Image Processing*, ser. ICWIP 2017, Paris, France: Association for Computing Machinery, 2017, pp. 16–21, isbn: 9781450353076. doi: 10.1145/3150978.3150985. indirizzo: https://doi.org/10.1145/3150978.3150985.

[4] Y. Deigant, V. Akshat, H. Raunak, P. Pranjal e J. Avi, «A proposed method for lossless image compression in nano-satellite systems,» in *2017 IEEE Aerospace Conference*, 2017, pp. 1–11. doi: 10.1109/AERO.2017.7943682.

[5] K. Sayood, *Introduction To Data Compression*. Morgan Kaufmann, 2012, isbn: 978-0124157965.

[6] G. Wallace, «The JPEG still picture compression standard,» *IEEE Transactions on Consumer Electronics*, vol. 38, n. 1, pp. xviii–xxxiv, 1992. doi: 10.1109/30.125072.

[7] M. Marcellin, M. Gormish, A. Bilgin e M. Boliek, «An overview of JPEG-2000,» in *Proceedings DCC 2000. Data Compression Conference*, 2000, pp. 523–541. doi: 10.1109/DCC.2000.838192.

[8] F. Dufaux, G. J. Sullivan e T. Ebrahimi, «The JPEG XR image coding standard [Standards in a Nutshell],» *IEEE Signal Processing Magazine*, vol. 26, n. 6, pp. 195–204, 2009. doi: 10.1109/MSP.2009.934187.

[9] M. A. Rahman, M. Hamada e J. Shin, «The Impact of State-of-the-Art Techniques for Lossless Still Image Compression,» *Electronics*, vol. 10, n. 3, 2021, issn: 2079-9292. doi: 10.3390/electronics10030360. indirizzo: https://www.mdpi.com/2079-9292/10/3/360.

[10] C. Incorporated. «Graphics Interchange Format (GIF) Specification.» (1987), indirizzo: https://www.w3.org/Graphics/GIF/spec-gif87.txt.

[11] info@ar kleinert.de. «GIF 24 Bit (truecolor) extensions.» (2007), indirizzo: https://web.archive.org/web/20120316215949/http://uk.aminet.net/docs/misc/GIF24.readme.

[12] «Portable Network Graphics (PNG) Specification (Third Edition).» (2023), indirizzo: https://www.w3.org/TR/png/.

[13] X. Wu e N. Memon, «CALIC-a context based adaptive lossless image codec,» in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 4, 1996, 1890–1893 vol. 4. doi: 10.1109/ICASSP.1996.544819.

[14] GeeksforGeeks. «Image Edge Detection Operators in Digital Image Processing.» (), indirizzo: https://www.geeksforgeeks.org/image-edge-detection-operators-in-digital-image-processing/.

[15] Jpeg.org. «JPEG - JPEG LS.» (), indirizzo: https://jpeg.org/jpegls/index.html.

[16] J. Hu, S. Song e Y. Gong, «Comparative performance analysis of web image compression,» in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2017, pp. 1–5. doi: 10.1109/CISP-BMEI.2017.8301939.

[17] Adobe.com. «Introduzione ai file WebP | Formato di immagine web di Google | Adobe.» (2024), indirizzo: https://www.adobe.com/it/creativecloud/file-types/image/raster/webp-file.html#:~:text=Il%20formato%20WebP%20%C3%A8%20altamente,file%20di%20dimensioni%20pi%C3%B9%20gestibili.

[18] G. for Developers. «Compression Techniques.» (2024), indirizzo: https://developers.google.com/speed/webp/docs/compression.

[19] T. H. Mandeel, M. Imran Ahmad, N. A. A. Khalid e M. N. Md Isa, «A Comparative Study on Lossless compression mode in WebP, Better Portable Graphics (BPG), and JPEG XL Image Compression Algorithms,» in *2021 8th International Conference on Computer and Communication Engineering (ICCCE)*, 2021, pp. 17–22. doi: 10.1109/ICCCE50029.2021.9467224.

[20] Theindexproject.org. «BPG IMAGE FORMAT - The Index Project.» (2015), indirizzo: https://theindexproject.org/award/nominees/265.

[21] F. Li, S. Krivenko e V. Lukin, «An Approach to Better Portable Graphics (BPG) Compression with Providing a Desired Quality,» in *2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT)*, 2020, pp. 13–17. doi: 10.1109/ATIT50783.2020.9349289.

[22] W. Contributors. «Better Portable Graphics.» (2024), indirizzo: https://en.wikipedia.org/wiki/Better_Portable_Graphics.

[23] M. Sharabayko, O. Ponomarev e R. Chernyak, «Intra compression efficiency in VP9 and HEVC,» *Applied Mathematical Sciences*, vol. 7, pp. 6803–6824, nov. 2013. doi: 10.12988/ams.2013.311644.

[24] J. Lainema, M. M. Hannuksela, V. K. M. Vadakital e E. B. Aksu, «HEVC still image coding and high efficiency image file format,» in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 71–75. doi: 10.1109/ICIP.2016.7532321.

[25] F. Bellard. «BPG Specification.» (2014), indirizzo: https://bellard.org/bpg/bpg_spec.txt.

[26] J. Sneyers e P. Wuille, «FLIF: Free lossless image format based on MANIAC compression,» in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 66–70. doi: 10.1109/ICIP.2016.7532320.

[27] Flif.info. «FLIF - Free Lossless Image Format.» (2024), indirizzo: https://flif.info/.

[28] J. Sneyers. «FLIF16 Specification.» (2016), indirizzo: https://flif.info/spec.html#_overview_of_the_format.

[29] W. Contributors. «YCoCg.» (2023), indirizzo: https://en.wikipedia.org/wiki/YCoCg.

[30] Caniuse.com. «AVIF image format.» (2024), indirizzo: https://caniuse.com/avif.

[31] M. Saldanha, M. Corrêa, G. Corrêa et al., «An Overview of Dedicated Hardware Designs for State-of-the-Art AV1 and H.266/VVC Video Codecs,» in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2020, pp. 1–4. doi: 10.1109/ICECS49266.2020.9294862.

[32] D. Barina. «Comparison of Lossless Image Formats.» (2021), indirizzo: https://arxiv.org/abs/2108.02557.

[33] «AVIF Format: A Next-Gen Image Format to Rule them All?» (2024), indirizzo: https://cloudinary.com/guides/image-formats/avif-format-a-next-gen-image-format-to-rule-them-all#avif-file-format-structure-and-components.

[34] Wikipedia.org. «AVIF.» (2022), indirizzo: https://it.wikipedia.org/wiki/AVIF.

[35] A. W. Group. «AV1 Image File Format (AVIF).» (2024), indirizzo: https://aomediacodec.github.io/av1-avif/.

[36] LambdaTest. «AVIF Image Format - The Next-Gen Compression Codec.» (2020), indirizzo: https://www.lambdatest.com/blog/avif-image-format/.

[37] U. F. S. Elia. «Formato AVIF: come migliorare qualità immagini con minimo peso.» (2023), indirizzo: https://www.unidformazione.com/formato-avif-migliorare-peso-immagini.

[38] Imgix.com. «AVIF: The Best Image Format That You Might Be Missing.» (2023), indirizzo: https://www.imgix.com/blog/avif-best-image-format.

[39] G. Mancini. «Formato immagine AVIF: storia, caratteristiche e vantaggi SEO.» (2023), indirizzo: https://www.seozoom.it/avif-formato-immagine-guida/.

[40] Wikipedia.org. «JPEG XL.» (2021), indirizzo: https://it.wikipedia.org/wiki/JPEG_XL.

[41] J. Alakuijala, R. van Asseldonk, S. Boukortt et al., «JPEG XL next-generation image compression architecture and coding tools,» in *Applications of Digital Image Processing XLII*, A. G. Tescher e T. Ebrahimi, cur., International Society for Optics e Photonics, vol. 11137, SPIE, 2019, 111370K. doi: 10.1117/12.2529237. indirizzo: https://doi.org/10.1117/12.2529237.

[42] B. S. C. L. V. G. J. S. J. Alakuijala Google J. Switzerland e S. W. Google. «JPEG White Paper: JPEG XL Image Coding System.» (2023), indirizzo: https://ds.jpeg.org/whitepapers/jpeg-xl-whitepaper.pdf.

[43] M. E.-S. R. Mamedov e N. Madhavji. «Analysis and Enhancement of Lossless Image Compression in JPEG XL Final Progress Report.» (2024), indirizzo: https://arxiv.org/pdf/2404.19755.

[44] https://jpegxl.info/. «JPEG XL - overview.» (2019), indirizzo: https://docs.google.com/presentation/d/1LlmUR0Uoh4dgT3DjanLjhlXrk_5W2nJBDqDAMbhe8v8/edit#slide=id.gde87dfbe27_0_25.

[45] Caniuse.com. «JPEG XL image format | Can I use... Support tables for HTML5, CSS3, etc.» (2021), indirizzo: https://caniuse.com/jpegxl.

[46] Jpeg.org. «JPEG - JPEG XR.» (2024), indirizzo: https://jpeg.org/jpegxr/.

[47] Cloudinary.com. «Lossless Image Formats Comparison: FLIF vs.PNG, WebP and BPG.» (2015), indirizzo: https://cloudinary.com/blog/flif_the_new_lossless_image_format_that_outperforms_png_webp_and_bpg.

[48] E. Ozturk e A. Mesut, «Performance Evaluation of JPEG Standards, WebP and PNG in Terms of Compression Ratio and Time for Lossless Encoding,» in *2021 6th International Conference on Computer Science and Engineering (UBMK)*, 2021. doi: https://doi.org/10.1109/ubmk52708.2021.9558922.