

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Corso di Laurea Magistrale in Informatica
Compressione Dati

Light Field Image Compression

Relatore:

**Prof.
Bruno CARPENTIERI**

Team:

Matteo Della Rocca
Mat. 0522501500
Luca Boffa
Mat. 0522501521
Vincenzo Di Leo
Mat. 0522501408

ANNO ACCADEMICO 2023/2024

INDICE

1	Introduzione	3
1.1	Gli ologrammi	3
1.2	Un pò di storia	3
1.3	Obiettivi dello studio	4
1.4	Organizzazione del paper	5
2	LIGHT FIELD IMAGE	6
3	Lavori correlati	8
4	Fase di ricerca	9
4.1	Tabella riassuntiva	10
4.1.1	Cirrus Logic AccuPak	10
4.1.2	FLV	10
4.1.3	Motion JPEG	11
4.1.4	MPEG4	11
4.1.5	ProRes	11
4.1.6	FFVHUFF	12
4.1.7	LCL-ZLIB	12
4.1.8	MagicYuv	12
4.2	Metriche selezionate per il confronto	13
4.2.1	SSIM	13
4.2.2	PSNR	14
4.2.3	Test indice SSIM con riferimento	15
4.2.4	Test indice PSNR con riferimento	15
5	Sistema proposto	16
6	Implementazione	17
6.1	Linguaggio, librerie principali e tool usati	17

INDICE

6.2	Organizzazione del progetto e codice	18
6.2.1	Moduli python principali	18
6.2.2	Directory principali	18
6.3	Metodologie esperimento	18
6.4	Fase di compressione	19
6.5	Fase di decompressione	20
7	Esperimenti svolti	21
7.1	Applicazione del sistema proposto sui dataset utilizzati in precedenza . .	21
7.2	Studio sulla correlazione tra light field contigui	21
7.3	Aggiunta di nuovi codec e nuovi dataset	22
7.3.1	Dataset utilizzati	22
8	Analisi dei risultati	26
8.1	Configurazione Hardware	26
8.2	Nomenclatura esperimenti svolti	26
8.3	Risultati ottenuti	27
8.3.1	Codec 2022 sui dataset 2022	27
8.3.2	Codec 2022 sui dataset 2022 (random)	29
8.3.3	Codec 2022 sui dataset 2023	30
8.3.4	Codec 2022 sui dataset 2023 (random)	32
8.3.5	Codec 2023 sui dataset 2022	34
8.3.6	Codec 2023 sui dataset 2022 (random)	36
8.3.7	Codec 2023 sui dataset 2023	38
8.3.8	Codec 2023 sui dataset 2023 (random)	41
8.4	Decompressione codec 2023 su Dataset 2022	44
8.5	Decompressione codec 2023 su Dataset 2022 (random)	46
8.6	Decompressione codec 2023 su Dataset 2023	48
8.7	Decompressione codec 2023 su Dataset 2023 (random)	51
8.8	Analisi riassuntiva dei risultati	54
8.8.1	Analisi sui dataset random	55
9	Conclusione e Sviluppi futuri	56
Riferimenti		57
Elenco delle figure		59
Elenco delle tabelle		61

CAPITOLO 1

INTRODUZIONE

1.1 Gli ologrammi

Gli ologrammi sono definiti come figure o pattern d'onda interferenti, ottenute tramite l'uso di un laser, aventi la specificità di creare un effetto fotografico tridimensionale: essi, a differenza delle normali fotografie, ci mostrano una rappresentazione tridimensionale dell'oggetto proiettato. Poiché le statistiche e le proprietà dei segnali olografici differiscono notevolmente dalle immagini naturali come le fotografie, le soluzioni di codifica convenzionali non sono ottime. Ad oggi si hanno problemi nel salvataggio e trasmissione degli ologrammi dovuti alla loro dimensione. Pertanto, sono necessarie nuove soluzioni di codifica e trasformazioni per comprimere gli ologrammi in modo più efficace.

1.2 Un pò di storia

Gli ologrammi nascono intorno alla prima metà degli anni '40. Dennis Gabor, famoso scienziato ungherese, sviluppò la teoria dell'olografia mentre lavorava per migliorare la risoluzione di un microscopio elettronico. Fu egli stesso a coniare il termine "ologramma", derivante dall'unione delle due parole greche **holos** (intero) e **gramma** (messaggio). Purtroppo, negli anni a seguire gli ologrammi non suscitarono grande successo, date le sorgenti luminose ancora poco sviluppate ai tempi.

Finalmente, negli anni '60, fu inventato il laser che, grazie al suo potente lampo di luce, dalla durata di pochi nanosecondi, si rivelò perfetto per creare ologrammi. Il laser, infatti, riesce a bloccare il movimento efficacemente, creando in tal modo ologrammi di eventi o persone ad alta velocità. In particolare, il primo ologramma di una persona è stato creato nel 1967. Da qui nasce la ritrattistica olografica pulsata.

Nel 1962 due studiosi americani decisero di superare la teoria di Gabor e utilizzare, oltre al laser, anche una tecnica messa in atto già nel loro lavoro (studiavano un modo per realizzare un radar a lettura laterale). Il risultato di tutto ciò fu la creazione dei primi ologrammi 3D in movimento: un trenino e un uccello.

Un ulteriore sviluppo nel mondo degli ologrammi è stato raggiunto nel '68 da Stephen A. Benton che ha inventato l'olografia a trasmissione di luce bianca, la quale permette di creare un'immagine “arcobaleno” dai sette colori che compongono la luce bianca. L'invenzione di Benton è importante in quanto ha reso possibile la produzione in serie di ologrammi attraverso una tecnica di goffratura.

Oggi gli strumenti necessari a creare ologrammi 3D (un laser a onda continua, dispositivi ottici come lenti o specchi per dirigere la luce laser, un supporto per pellicola e un tavolo isolante su cui vengono effettuate le esposizioni) sono posseduti da moltissimi laboratori e studi.[1]

1.3 Obiettivi dello studio

Questo studio rappresenta un'estensione di una ricerca precedente nel campo della compressione delle immagini Light Field, mantenendo lo stesso approccio metodologico. L'obiettivo principale è di confermare o smentire le conclusioni raggiunte nel precedente studio e di arricchire la ricerca mediante l'introduzione di nuovi codec, dataset e criteri di valutazione.

Le immagini Light Field sono particolari tipologie di immagini utilizzate per rappresentare gli ologrammi attraverso dispositivi di visualizzazione appositi. La loro compressione richiede particolare attenzione per mantenere la qualità dell'immagine e ridurre lo spazio di archiviazione necessario.

Per questo studio, sono stati selezionati algoritmi di compressione lossy e lossless applicati ai dataset delle immagini Light Field. La valutazione delle prestazioni di tali algoritmi si basa su criteri oggettivi come il rapporto di compressione e il tempo impiegato durante la compressione. Durante la decompressione, l'attenzione è focalizzata sull'accuratezza della ricostruzione dell'immagine, valutata tramite l'indice SSIM (Structural Similarity Index) e un nuovo indice, il PSNR (Peak Signal-to-Noise Ratio), specifico per i codec lossy.

L'inclusione di nuovi codec e dataset amplia la portata dello studio, consentendo di esplorare una gamma più ampia di opzioni di compressione e di valutare il loro impatto sulle prestazioni complessive degli algoritmi. Inoltre, è stato condotto un esperimento sulle relazioni inter-frame nelle immagini light field per comprendere meglio

il loro effetto sull'efficacia dei codec durante le fasi di compressione e decompressione.

1.4 Organizzazione del paper

La struttura del paper rispecchia gli argomenti affrontati per lo studio, la progettazione e l'implementazione della soluzione proposta:

- **LIGHT FIELD IMAGE:** raccolta dei dati relativi ai light field image e l'importanza della compressione di questi ultimi.
- **LAVORI CORRELATI:** studio dei paper precedentemente realizzati per la compressione dei light field.
- **FASE DI RICERCA:** approfondimento delle tecnologie e dei linguaggi necessari al progetto, nonché la ricerca degli algoritmi di compressione disponibili sul mercato allo stato dell'arte.
- **SISTEMA PROPOSTO:** nella quale viene spiegata la nostra idea di soluzione.
- **IMPLEMENTAZIONE:** nella quale si è messo in atto quanto definito in fase di progettazione.
- **ESPERIMENTI SVOLTI:** esperimenti condotti al fine di valutare l'efficacia del sistema proposto nella compressione video applicata alle light field images.
- **ANALISI DEI RISULTATI:** esposizione degli esempi di compressione con gli algoritmi implementati e confronto tra i risultati ottenuti.
- **CONCLUSIONE E SVILUPPI FUTURI:** analisi del lavoro svolto nel complesso e i possibili sviluppi futuri.

CAPITOLO 2

LIGHT FIELD IMAGE

Light Field (campo luminoso) si riferisce alla quantità di luce in funzione della posizione e della direzione, ovvero una funzione vettoriale 4D nello spazio libero. Il campo luminoso è stato introdotto nella computer graphics nel 1996 da Levoy e Hanrahan, i quali hanno proposto un sistema di rendering basato su immagini (IBR) e sul campo luminoso registrato di una scena. Tale sistema IBR è potente nel rendering di nuove view, non solo con il cambiamento del punto di vista, ma anche per il punto focale cambiato, noto come ri-messa a fuoco. Tuttavia, la registrazione del campo luminoso 4D non è banale poiché i sensori di immagini esistenti sono progettati per l'imaging 2D. Una fotocamera tradizionale cattura solo una rappresentazione piatta e bidimensionale dei raggi di luce che raggiungono l'obiettivo in una data posizione. Il sensore di immagine 2D registra la somma della luminosità e del colore di tutti i raggi luminosi che arrivano a ogni singolo pixel. Al contrario, una light field camera può registrare non solo i valori di luminosità e colore nel sensore di immagini 2D, ma anche la direzione/l'angolo di tutti i raggi luminosi che arrivano al sensore. Queste informazioni aggiuntive ci permettono di ricostruire da dove proveniva esattamente ogni raggio di luce prima di raggiungere la telecamera, rendendo possibile il calcolo di un modello tridimensionale della scena catturata.

I light field possono essere catturati utilizzando diverse tecniche, tra cui:

- una singola telecamera controllata roboticamente,
- un arco rotante di telecamere,
- una serie di telecamere o moduli telecamere,
- una singola fotocamera dotata di un array di microlenti.

A seconda della tecnica utilizzata, i dati dell'immagine acquisita sono tipicamente costituiti da un'immagine contenente molte immagini secondarie (ottenute da un singolo

2. LIGHT FIELD IMAGE

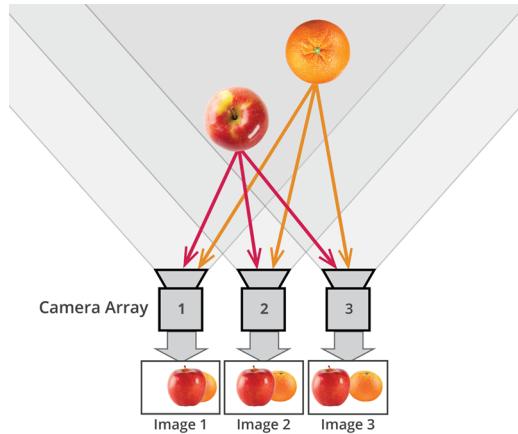


Figura 2.1: Cattura di un ligh field

sensore e una matrice di microlenti) o molte immagini (da più fotocamere o esposizioni). Entrambi i set di dati mostrano immagini con leggere variazioni, poiché catturano raggi di luce da diverse angolazioni nello spazio. Queste variazioni permettono di determinare la posizione degli oggetti e di creare un volume di campo luminoso 3D. Tuttavia, le fotocamere plenottiche come Lytro e Raytrix, sebbene offrano nuove opportunità, richiedono una compressione efficace per gestire le dimensioni considerevoli delle immagini raw e la loro rappresentazione non convenzionale, mantenendo al contempo la qualità dell'immagine.

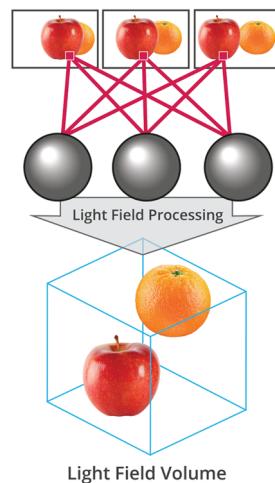


Figura 2.2: Informazioni racchiuse in un ligh field

CAPITOLO 3

LAVORI CORRELATI

Per la compressione dei light field ci siamo rifatti allo studio di un progetto passato svolto da nostri colleghi nell'università di Salerno[2].

I loro risultati si basano sulla compressione dei light field (che sono rappresentati tramite una serie immagini) in file di tipo video, così da sfruttare vari codec che si usano per comprimere le immagini in video.

Precisamente i nostri colleghi hanno utilizzato i seguenti codec:

- HVEC (lossy e lossless)
- VP9 (lossy e lossless)
- AV1 (lossy e lossless)
- HuffYUV (lossless)
- UT Video (lossless)
- FFV1 (lossless)

La nostra idea è quella di proseguire la strategia che i nostri colleghi hanno proposto andando ad espandere il loro studio già presente.

CAPITOLO 4

FASE DI RICERCA

Parte della ricerca si è basata sullo studio degli algoritmi utilizzati, e precisamente:

- perché e quando sono stati creati,
- da chi sono stati prodotti,
- a quali bisogni rispondono,
- da chi sono stati utilizzati nel tempo.

Inoltre, è stato condotto un esperimento per determinare se vi fosse una dipendenza effettiva tra i vari frame dei light field e se tale dipendenza potesse influenzare la qualità complessiva della compressione/decompressione.

Infine, è stata preparata una tabella riassuntiva per evidenziare le peculiarità fondamentali per la comprensione dei risultati del progetto. Ulteriormente, sono stati esaminati il Structural Similarity Index (SSIM) e il Peak Signal-to-Noise Ratio (PSNR), due metodi utilizzati per valutare la qualità percepita delle immagini in scenari di compressione lossy.

4.1 Tabella riassuntiva

Codec	Licenza	Perdita
Cirrus Logic	Commerciale	Lossy
	FLV	GNU
	MJPEG	GNU
	MPEG4	GNU
	ProRes	Commerciale
FFVHUFF	GNU	Lossless
LCL-ZLIB	GNU	Lossless
MAGICYUV	Commerciale	Lossless

4.1.1 Cirrus Logic AccuPak

Cirrus Logic AccuPak è un codec YUV a precisione ridotta.

Il codec AccuPak racchiude 4 campioni Y e 2 campioni C in 32 bit rappresentando ogni campione Y con 5 bit e ogni campione C con 6 bit. È essenzialmente un metodo ridimensionato di codifica YUV 4:1:1, in cui ogni gruppo di 4 pixel su una linea è rappresentato da un campione di luminanza ciascuno ma condivide campioni C[3].

4.1.2 FLV

Flash Video è un formato di file utilizzato per fornire contenuti video digitali (ad esempio, programmi TV, film, ecc.) su Internet utilizzando Adobe Flash Player versione 6 e versioni più recenti. Il contenuto Flash Video può anche essere incorporato nei file SWF. Ci sono due diversi formati di file Flash Video: FLV e F4V. I dati audio e video all'interno dei file FLV sono codificati allo stesso modo dei file SWF. FLV è stato originariamente sviluppato da Macromedia. Nei primi anni 2000, Flash Video era lo standard de facto per lo streaming video basato sul web (su RTMP).

I file Flash Video FLV di solito contengono materiale codificato con codec che seguono i formati di compressione video Sorenson Spark o VP6. A partire dal 2010 le versioni pubbliche di Flash Player (collaborazione tra Adobe Systems e MainConcept) supportano anche il video H.264 e l'audio HE-AAC. Tutti questi formati di compressione sono limitati dai brevetti. Flash Video è visualizzabile sulla maggior parte dei sistemi operativi tramite Adobe Flash Player e il plugin del browser web o uno dei numerosi programmi di terze parti. I dispositivi iOS di Apple, insieme a quasi tutti gli altri dispositivi mobili, non supportano il plugin Flash Player e quindi richiedono altri metodi di consegna come quelli forniti da Adobe Flash Media Server[4].

4.1.3 Motion JPEG

Motion JPEG (M-JPEG) è un codec video nel quale ogni singolo frame del video viene compresso in un’immagine JPEG.

Non offre nessuna compressione interframe, questo fa sì che la qualità della compressione sia indipendente dal movimento presente nell’immagine, a differenza della compressione MPEG dove ci possono essere problemi di qualità quando il video contiene movimenti veloci o cambi scena. Questo codec facilita il montaggio video, in quanto permette tagli su ogni singolo frame, e non solo all’inizio di un gruppo di frame.

Si tratta di un formato che ha avuto una certa diffusione su fotocamere digitali e camere IP in quanto permetteva di usare la tecnologia della compressione JPEG anche per i filmati. Pur richiedendo un bitrate superiore al formato MPEG-1 permette risoluzioni superiori. È stato gradualmente soppiantato dai formati MOV(Apple), MPEG-2 e MPEG-4[5].

4.1.4 MPEG4

In elettronica e telecomunicazioni MPEG-4, nato nel 1996 e finalizzato nel 1998 (fu presentato pubblicamente a settembre di quell’anno), è il nome dato a un insieme di standard per la codifica dell’audio e del video digitale sviluppati dall’ISO/IEC Moving Picture Experts Group (MPEG). L’MPEG-4 è uno standard utilizzato principalmente per applicazioni come la videoteléfono e la televisione digitale, per la trasmissione di filmati via Web, e per la memorizzazione su supporti CD-ROM.

MPEG-4 è suddiviso in vari sotto standard chiamati part (termine inglese che in italiano significa ”parte”) e noi abbiamo usato la parte 2: un codec di compressione per i dati visivi (video, still textures...). Uno dei tanti ”profili” della parte 2 è il Advanced Simple Profile (ASP).

Il concetto alla base del codec (COdificatore-DECodificatore) MPEG-4 è la quantizzazione. Senza scendere nello specifico, si può riassumere come quel processo che permette di trasmettere solamente la variazione dell’immagine mediante un apposito algoritmo di compressione[6].

4.1.5 ProRes

Apple ProRes è un formato di compressione video lossy di alta qualità, ”visually lossless” sviluppato da Apple Inc. per l’uso in post-produzione che supporta una risoluzione video fino a 8K. È il successore dell’Apple Intermediate Codec ed è stato introdotto nel 2007 con Final Cut Studio 2. Proprio come gli standard H.26x e MPEG, la famiglia di codec ProRes utilizza algoritmi di compressione basati sulla trasformata del coseno discreto (DCT). ProRes è ampiamente utilizzato come metodo di consegna del formato finale per i file di trasmissione HD in pubblicità, funzionalità, Blu-ray e streaming.

ProRes è una linea di codec intermedi, il che significa che sono destinati all'uso durante l'editing video e non alla visualizzazione pratica da parte dell'utente finale. Ciò si ottiene utilizzando solo la compressione intra-frame, dove ogni fotogramma viene memorizzato in modo indipendente e può essere decodificato senza dipendenze da altri fotogrammi. Il vantaggio di un codec intermedio è che offre eccellenti prestazioni di accesso casuale nelle applicazioni di post-produzione e mantiene una qualità superiore rispetto ai codec dell'utente finale, pur richiedendo sistemi disco molto meno costosi rispetto ai video non compressi. È paragonabile al codec DNxHD o CineForm di Avid che offrono bitrate simili e sono anche destinati ad essere utilizzati come codec intermedi. ProRes è un codec solo intra-frame basato su scalare DCT ed è quindi più semplice da decodificare rispetto ai formati orientati alla distribuzione come H.264. Nel 2018 Apple ha aggiunto un nuovo "ProRes RAW" (filtro Bayer compresso) a Final Cut Pro X, dopo che Blackmagic Design ha implementato Bayer compresso come "CinemaDNG 3:1" e "CinemaDNG 4:1" nelle loro fotocamere e DaVinci Resolve[7].

4.1.6 FFVHUFF

FFVHUFF è un codec video senza perdita di dati ed è una versione migliorata, e più veloce, del codec huffyuv. Può gestire più formati pixel[8].

4.1.7 LCL-ZLIB

Il codec converte i dati dell'immagine RGB24 originali in uno spazio colore di destinazione e lo comprime con un algoritmo selezionato. Il codec può anche rimuovere i fotogrammi invariati e sostituirli con fotogrammi nulli e può filtrare i dati dell'immagine prima della compressione. L'unica differenza tra avimszh e avizlib è nel compressore di flusso. Il filtraggio PNG è disponibile solo in avizlib. Fatta eccezione per i frame nulli, non c'è compressione temporale, e tutti i frame possono essere decodificati indipendentemente dagli altri. Ogni blocco AVI contiene un fotogramma. In caso di modalità multithread, le due sezioni sono memorizzate nello stesso blocco.

Come suggerisce il nome del codec, tutti i compressori sono lossless.

Questa modalità utilizza il metodo standard di sgonfiamento zlib. Per la descrizione dell'algoritmo fare riferimento ai documenti zlib. Lo stato del compressore viene reimpostato ogni fotogramma (decodifica ogni fotogramma in modo indipendente). I codici di compressione (1, 9, -1) hanno lo stesso significato dei flag di compressione zlib. Zlib non richiede un livello di compressione al decompressore, quindi il valore è lì solo a scopo informativo[9].

4.1.8 MagicYuv

Un codec video ad alte prestazioni, ultraveloce e matematicamente senza perdite per la registrazione, l'archiviazione, la post-produzione e l'editing ad alte risoluzioni.

MagicYUV è stato progettato per la velocità e per supportare pienamente la codifica e la decodifica multi-thread.

Questo codec è utilizzato per:

- Gaming
- Registrazioni e catture
- Editing e post-produzione
- Ricerca

Il Codec MagicYUV consente ai ricercatori di catturare ed elaborare video alle più alte risoluzioni e framerate possibili, in profondità di bit regolari e a colori profondi, mantenendo intatta ogni bit delle informazioni.

Questo rende MagicYUV uno dei codec video lossless più veloci del suo genere[10].

4.2 Metriche selezionate per il confronto

4.2.1 SSIM

L'indice SSIM (Structural Similarity Index Measure) è un modello basato sulla percezione che valuta la degradazione dell'immagine considerando i cambiamenti nella percezione delle informazioni strutturali. Questo metodo tiene conto di importanti fenomeni basati sulla percezione, come il mascheramento della luminanza e il mascheramento del contrasto. Il termine "informazioni strutturali" si riferisce ai pixel fortemente interdipendenti o spazialmente vicini, enfatizzando la correlazione tra di essi. SSIM stima la qualità percepita di immagini e video. Misura la somiglianza tra due immagini: l'originale e la recuperata[11].

In particolare, date due immagini o segnali N -dimensionali (o porzioni corrispondenti), $x = (x_1, \dots, x_N)$ e $y = (y_1, \dots, y_N)$, l'indice SSIM esamina le similarità tra luminanza, contrasto e struttura.

1. Per la luminanza, $l(x, y)$, si utilizzano i valori medi, ad esempio,

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i.$$

2. Per il contrasto, $c(x, y)$, si utilizzano le varianze, ad esempio,

$$s_x^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2.$$

3. Per la struttura, $s(x, y)$, si utilizzano segnali normalizzati (deviazione standard unitaria), ad esempio,

$$x' = \frac{x - \bar{x}}{s_x}.$$

Successivamente, si combinano questi componenti (in qualche modo!) per ottenere una misura complessiva di similarità, cioè,

$$S(x, y) = F(l(x, y), c(x, y), s(x, y)).$$

4.2.2 PSNR

Il PSNR (Peak Signal-to-Noise Ratio) è una metrica utilizzata per calcolare il rapporto tra la potenza massima del segnale possibile e la potenza del rumore di distorsione che influisce sulla qualità della sua rappresentazione. Questo rapporto tra due immagini viene misurato in decibel per tener conto della vasta gamma dinamica dei segnali, che variano dai valori più grandi a quelli più piccoli. Il PSNR viene comunemente calcolato come il logaritmo in scala decibel, poiché i segnali hanno una gamma dinamica estesa. Questa gamma dinamica riflette la variazione tra i valori più alti e più bassi possibili, che sono critici per valutare la qualità dell'immagine in termini di fedeltà alla sua rappresentazione originale. Utilizzato ampiamente come tecnica di valutazione della qualità, il PSNR misura la qualità della ricostruzione nei codec di compressione delle immagini lossy. In questo contesto, il segnale rappresenta i dati originali, mentre il rumore è l'errore introdotto dalla compressione o dalla distorsione.

Nel degrado della qualità della compressione dell'immagine e del video, il valore PSNR varia da 30 a 50 dB per la rappresentazione dei dati a 8 bit e da 60 a 80 dB per i dati a 16 bit.

Il PSNR è espresso come:

$$PSNR = 10\log_{10}(peakval^2)/MSE$$

Qui, *peakval* (Peak Value) è il massimo nei dati dell'immagine. Se si tratta di un tipo di dati intero senza segno a 8 bit, il *peakval* è 255. Dall'equazione, possiamo vedere che è una rappresentazione dell'errore assoluto in dB[11].

Video Quality (MOS)	PSNR Range
Excellent	> 37
Good	31-37
Fair	25-31
Poor	20-25
Bad	< 20

Figura 4.1: Rapporto tra PSNR e qualità del video

Nel paper ”Social-Aware Video Multicast Based on Device-to-Device Communications” [12], viene condotto uno studio che esamina il rapporto tra la qualità del video e il valore dell'indice PSNR. La Tabella 4.1 fornisce una guida per interpretare i valori di PSNR e associarli a diverse categorie di qualità video. In particolare, si osserva che una qualità del video superiore è indicata da valori di PSNR pari o superiori a 37, mentre una qualità inferiore è associata a valori di PSNR inferiori a 25.

Nella fase di analisi dei risultati, la Tabella 4.1 sarà utilizzata come riferimento per valutare la qualità dei video decompressi. In altre parole, i valori ottenuti durante l'esperimento saranno confrontati con i range indicati nella tabella per determinare se

la qualità del video è considerata ottima, buona, accettabile o pessima, in base ai criteri definiti dagli intervalli di PSNR.

4.2.3 Test indice SSIM con riferimento

Per valutare l'accuratezza del calcolo dell'indice di similarità strutturale (SSIM) nel nostro lavoro, è stato condotto un confronto con i valori riportati nel paper di riferimento *"The SSIM Index for Image Quality Assessment"* [13]. A tale scopo, le stesse immagini utilizzate nel paper di riferimento sono state scaricate e utilizzate per il calcolo dell'indice SSIM mediante uno script personalizzato basato sulla libreria *skimage.metrics*. I valori ottenuti da questo script sono stati quindi confrontati con quelli riportati nel paper di riferimento.

Risultati

Il confronto tra i valori SSIM ottenuti dal nostro script e quelli riportati nel paper di riferimento ha mostrato una buona corrispondenza tra i due. Tuttavia, è stato osservato un errore medio relativo, calcolato come la media delle differenze percentuali tra i valori calcolati e quelli di riferimento, pari a **0.030**. Questo risultato suggerisce che il nostro script per il calcolo dell'indice SSIM ha una precisione accettabile e può essere considerato affidabile per valutazioni quantitative della qualità delle immagini.

4.2.4 Test indice PSNR con riferimento

Per valutare l'accuratezza del calcolo del PSNR, sono state selezionate due immagini - l'originale e quella con rumore - estratte dalla pagina web specificata [14], la quale fa riferimento a due paper accademici. Successivamente, utilizzando le informazioni fornite in questi paper, il PSNR è stato calcolato per le due immagini mediante l'utilizzo di uno script apposito che testa la funzione per il calcolo del PSNR fornita dalla libreria *skimage.metrics* e usata nell'esperimento.

Risultati

I valori del PSNR ottenuti sono stati confrontati con quelli riportati sulla pagina web di riferimento, che fornisce i valori di PSNR associati alle immagini specificate. Questo confronto ha permesso di determinare l'errore relativo del PSNR, risultato essere dello 0.026%. Questo risultato indica che vi è una buona corrispondenza tra i valori calcolati utilizzando la funzione di calcolo del PSNR fornita dalla libreria *skimage.metrics* e quelli riportati sulla pagina web di riferimento, confermando l'accuratezza del calcolo del PSNR.

CAPITOLO 5

SISTEMA PROPOSTO

Il nostro sistema ha come scopo principale quello di fornire un confronto tra vari algoritmi di compressione video e analizzare le loro prestazioni nella compressione di light field.

Sostanzialmente, il nostro sistema è progettato per migliorare ed espandere uno studio precedente.

Come detto nel capitolo 3, le light field image hanno una forte componente di similarità tra frame della stessa scena. Si è deciso di sfruttare l'alta inter-dipendenza tra le varie immagini proprio come si fa per i video.

A tal fine abbiamo ri-implementato due script Python diversi, uno per la compressione dei light field in video e l'altro per la decompressione del video nelle sue componenti. Per la compressione usiamo diversi codec video, tutti di tipo lossless e lossy.

Per la decompressione invece usiamo il video creato nelle fasi precedenti e ne estraiamo i singoli frame. I frame estratti vengono poi salvati nello stesso formato delle immagini originali per monitorare così eventuali perdite di dati. Per ogni algoritmo di compressione effettuiamo un'analisi dello spazio risparmiato tramite il confronto della dimensione totale del dataset non compresso e del video compresso. Confrontiamo ogni frame decompresso con il suo corrispettivo originale, usando la structural similarity index measure (SSIM), per verificare che tra le due versioni non ci sia stata una perdita di informazioni nel caso di algoritmi lossless e che la perdita sia "accettabile" per gli algoritmi lossy.

I vari processi verranno spiegati nel dettaglio al capitolo 6.

CAPITOLO 6

IMPLEMENTAZIONE

6.1 Linguaggio, librerie principali e tool usati

Per la realizzazione di questo progetto, è stato impiegato il linguaggio di programmazione Python (versione 3.11.6), facendo uso delle seguenti librerie:

1. **Subprocess (versione 3.8):** La libreria subprocess è stata utilizzata per generare nuovi processi, connettersi alle loro pipe di input/output/errore e ottenere i relativi codici di ritorno;
2. **FFmpeg (versione 6.1):** FFmpeg è una completa suite software nata nel dicembre 2000, specializzata in registrazione, conversione e riproduzione di audio e video. Basata sulla libreria libavcodec per la codifica audio/video, FFmpeg è sviluppata principalmente su Linux, ma può essere compilata ed eseguita su vari sistemi operativi, compreso Microsoft Windows. In particolare, l'uso di FFmpeg in questo contesto è stato orientato alla conversione video da un formato all'altro attraverso uno strumento da riga di comando.
3. **OpenCV (opencv-python==4.8.1.78):** OpenCV è una libreria open-source che fornisce un'ampia varietà di strumenti per la visione artificiale e il machine learning. Nell'implementazione, OpenCV (cv2) è utilizzata per le operazioni di elaborazione delle immagini e manipolazione dei frame video.
4. **CSV (csv):** La libreria csv inclusa in Python che offre funzionalità per la lettura e la scrittura di file CSV (Comma-Separated Values). Nell'implementazione è stata impiegata per la gestione per registrare risultati e statistiche del processo.
5. **Scikit-image (scikit-image==0.22.0):** Scikit-image è una raccolta di algoritmi per il processamento delle immagini basata su Scikit-Learn. Le funzioni `structural_similarity (ssim)` e `peak_signal_noise_ratio (psnr)` fornite da

`skimage.metrics` sono utilizzate per calcolare le metriche di similarità strutturale e rapporto segnale-rumore, rispettivamente.

6.2 Organizzazione del progetto e codice

6.2.1 Moduli python principali

- *test_compression.py*: Contiene funzioni e logica per la compressione video sui dataset presenti nel file *utils.py*.
- *test_decompression.py*: Gestisce la decompressione dei video compressi.
- *utils.py*: Contiene funzioni di utilità condivise, nel nostro caso sono presenti i dataset testati, le estensioni corrette dei codec video e le cartelle di output per la compressione e la decompressione.
- *random_dataset.py*: lo scopo di questo script è randomizzare l'ordine dei file corrispondenti al pattern nella cartella di origine e copiarli in modo ordinato nella cartella di destinazione.

6.2.2 Directory principali

- */datasets*: Contiene i dataset di input, in particolare ogni sottocartella fa riferimento ad uno specifico dataset.
- */compression_test*: Salva i risultati della compressione in una cartella contenente una sottocartella per ogni dataset. Ognuna di queste sottocartelle contiene a seconda dei codec usati il video compresso corrispondente. La seguente cartella in fase di compressione è creata in automatico.
- */decompression_test*: Salva i risultati della decompressione in una cartella contenente una sottocartella per ogni dataset: Ognuna di queste contiene a seconda dei codec usati un'altra sottocartella all'interno della quale sono salvate le immagini decomprese. La cartella in fase di decompressione è creata in automatico.
- */ffmpeg*: Nel caso di un sistema Windows, questa cartella è di fondamentale importanza in quanto contiene l'eseguibile di *ffmpeg*.

6.3 Metodologie esperimento

L'esperimento è stato automatizzato per semplificare il processo, trasformandolo in un processo di *benchmarking automatico* sia durante la fase di compressione che durante quella di decompressione. Di conseguenza, il programma eseguirà automaticamente i test con vari codec e sui vari dataset presenti nel file python *utils.py*, senza richiedere

un’interazione continua da parte dell’utente. Questo approccio è particolarmente utile per condurre test di benchmark o valutare le prestazioni di diversi algoritmi in modo efficiente. Per lanciare il test di compressione è possibile usare il comando *python test_compression.py* all’interno della cartella, mentre per testare la decompressione *python test_decompression.py*.

6.4 Fase di compressione

In fase di compressione sono state utilizzate diverse funzioni le quali firme seguono il pattern `comp_{codec_scelto}` e implementano il processo di compressione video mediante l’utilizzo di un codec specificato tramite FFmpeg. Il procedimento può essere riassunto in termini generici:

1. Parametri di Input:

- `input_path`: Percorso del file video originale;
- `output_path`: Percorso in cui verrà salvato il file video compresso.

2. Calcolo delle dimensioni:

Viene calcolata la dimensione totale del file originale attraverso la somma delle dimensioni di tutti i file presenti nella stessa cartella del file originale.

3. Utilizzo di FFmpeg per la compressione:

Utilizzando la libreria subprocess, viene eseguito FFmpeg dalla riga di comando. Il comando FFmpeg include specifiche come il tasso di frame di input, il codec video da utilizzare, insieme ai percorsi del file di input e di output;

4. Calcolo del tempo di compressione:

Il tempo di inizio viene registrato prima della chiamata FFmpeg, e il tempo di fine viene registrato dopo il completamento della compressione. La durata totale del processo di compressione viene calcolata sottraendo il tempo di inizio da quello di fine.

5. Calcolo del rapporto di compressione:

Utilizzando una funzione ausiliaria, vengono ottenute la dimensione del file compresso e il rapporto di compressione $\frac{\text{dimensione originale}}{\text{dimensione compressa}}$;

6. Output:

Ogni funzione di questo tipo restituisce la dimensione iniziale, la dimensione finale, il rapporto di compressione e il tempo impiegato per la compressione. Inoltre nella cartella di output sarà possibile visualizzare il video ottenuto dalla compressione.

Questo approccio è flessibile e può essere adattato per eseguire test con diversi codec, contribuendo così alla valutazione delle prestazioni dei codec nel contesto dell’esperimento.

6.5 Fase di decompressione

In fase di decompressione di un particolare video, il procedimento generale è questo:

1. **Ciclo di decompressione:** Questa è la fase più importante e principale della decompressione. Il codice itera attraverso ogni frame del video compresso usando la libreria av. Durante l'iterazione, i frame vengono decodificati e salvati come immagini in una nuova directory, corrispondente al nome del dataset;
2. **Creazione della struttura di output:** Viene creata una nuova directory per l'output della decompressione per ogni dataset. All'interno di ciascuna directory, vengono create sotto-directory per ogni algoritmo di compressione utilizzato.
3. **Applicazione di metriche di qualità:** Per ogni coppia di immagini decompresse (originale e compressa), vengono calcolate metriche di qualità quali SSIM e PSNR. Le metriche vengono calcolate tra ogni frame dell'immagine originale e del risultato della decompressione;
4. **Raccolta e salvataggio dei risultati:** I risultati delle metriche vengono raccolti e salvati in una struttura dati, che include informazioni come il nome del dataset, l'algoritmo di compressione utilizzato, e le medie delle metriche SSIM e PSNR (poiché vi è un confronto 1:1 tra immagine originale e immagine decompresa). I risultati vengono inoltre salvati in un file CSV per un'ulteriore analisi o documentazione.

CAPITOLO 7

ESPERIMENTI SVOLTI

Una serie di esperimenti è stata condotta al fine di valutare l'efficacia del sistema proposto nella compressione video applicata alle light field images.

7.1 Applicazione del sistema proposto sui dataset utilizzati in precedenza

Inizialmente sono stati eseguiti una serie di test utilizzando il nostro sistema per valutare le prestazioni dei dataset precedenti. Ciò ha consentito di ottenere una comprensione più approfondita delle dinamiche operative e dei risultati ottenuti dai nostri predecessori, permettendoci, inoltre, di identificare fino a quale punto il sistema precedente ha risposto alle esigenze previste. L'obiettivo principali è stato quello di valutare con maggiore precisione l'efficacia delle metodologie utilizzate precedentemente e di identificare eventuali punti deboli o aree di miglioramento.

7.2 Studio sulla correlazione tra light field contigui

Al fine di valutare l'impatto dell'ordine dei frames nei dataset originali, è stato condotto uno studio supplementare. Ogni dataset originale è stato sottoposto a un processo di trasformazione, generando così un nuovo dataset in cui i frames sono disposti in maniera casuale. L'obiettivo di questo studio è comprendere come la disposizione casuale dei frames influenzi le prestazioni e i risultati dei codec di compressione. Tale approccio è fondamentale per valutare la resilienza dei codec di compressione rispetto a variazioni nell'ordine temporale dei frames.

7.3 Aggiunta di nuovi codec e nuovi dataset

La scelta di aggiungere nuovi codec e di sottoporre tutti i dataset, vecchi e nuovi, a un riesame completo è stata fondamentale per comprendere in maniera esaustiva come le nuove implementazioni influiscano sulle prestazioni complessive del sistema. Tale approccio ha consentito di identificare correlazioni significative tra le caratteristiche dei codec, la selezione del dataset e le performance globali, contribuendo a delineare un quadro completo delle dinamiche del sistema in esame.

7.3.1 Dataset utilizzati

Per il miglioramento dello studio già effettuato dai nostri colleghi, abbiamo deciso di utilizzare i 3 dataset implementati nel loro progetto, in modo da poterli testare su nuovi codec, in aggiunta a 7 ulteriori dataset. Lo scopo è di avere un confronto tra più dati, in modo da raggiungere risultati più accurati.

I dadaset aggiunti sono i seguenti:

- Il dataset "Messerschmitt"[15], realizzato dal MIT Media Lab, propone 25 immagini in formato .png rappresentanti scene renderizzate, Figura7.1. Esse mostrano modelli 3D di una vettura del marchio Messerschmitt realizzati dallo Stanford University Computer Graphics Laboratory [16]. Tutti i light fields hanno una risoluzione di 840 x 593 pixel.
- Il dataset "Dice"[17], realizzato dal MIT Media Lab, propone 25 immagini in formato .png rappresentanti scene renderizzate, Figura 7.2. Esse mostrano modelli 3D di dadi da gioco realizzati dallo Stanford University Computer Graphics Laboratory [16]. Tutti i light fields hanno una risoluzione di 840 x 593 pixel.
- Il dataset "Fish"[18], realizzato dal MIT Media Lab, propone 25 immagini in formato .png rappresentanti scene renderizzate, Figura 7.3. Esse mostrano modelli 3D pesci realizzati dallo Stanford University Computer Graphics Laboratory [16]. Tutti i light fields hanno una risoluzione di 840 x 593 pixel.
- Il dataset "Car"[19] realizzato dal Max Planck Institut Informatik. Esso consiste in 101 immagini in formato .png, rappresentanti scene renderizzate, Figura 7.4. Esse presentano scene 3D di una autovettura. Tutti i light fields hanno una risoluzione spaziale di 960 x 720 pixel
- Il dataset "Cobblestone"[20] realizzato dal Max Planck Institut Informatik. Esso consiste in 101 immagini in formato .png, rappresentanti scene renderizzate, Figura 7.5. Esse presentano scene 3D di un sentiero di pietra. Tutti i light fields hanno una risoluzione spaziale di 960 x 720 pixel

7. ESPERIMENTI SVOLTI

- Il dataset "Mannequin"[21] realizzato dal Max Planck Institut Informatik. Esso consiste in 101 immagini in formato .png, rappresentanti scene renderizzate, Figura 7.6. Esse presentano scene 3D di un manichino in una stanza. Tutti i light fields hanno una risoluzione spaziale di 960 x 720 pixel
- Il dataset "Blob"[22] realizzato dal Max Planck Institut Informatik. Esso consiste in 101 immagini in formato .png, rappresentanti scene renderizzate, Figura 7.7. Esse presentano scene 3D di un "blob". Tutti i light fields hanno una risoluzione spaziale di 960 x 720 pixel



Figura 7.1: Immagine estratta dal dataset "Messerschmitt"



Figura 7.2: Immagine estratta dal dataset "Dice"

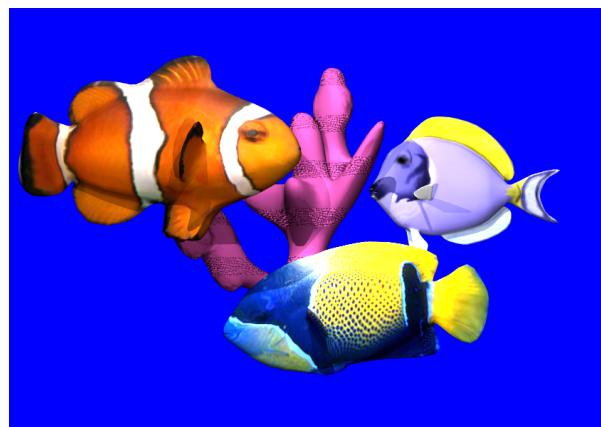


Figura 7.3: Immagine estratta dal dataset "Fish"



Figura 7.4: Immagine estratta dal dataset "Car"

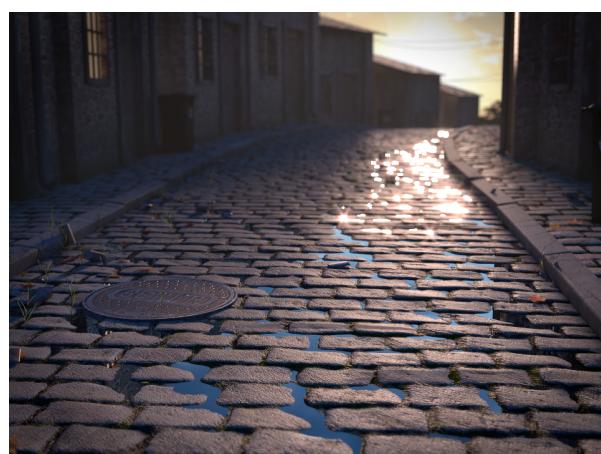


Figura 7.5: Immagine estratta dal dataset "Cobblestone"

7. ESPERIMENTI SVOLTI



Figura 7.6: Immagine estratta dal dataset "Mannequin"



Figura 7.7: Immagine estratta dal dataset "Blob"

CAPITOLO 8

ANALISI DEI RISULTATI

8.1 Configurazione Hardware

Gli esperimento sono stati condotti utilizzando una configurazione hardware di questo tipo:

- processore Ryzen 5 2600;
- scheda grafica GTX 1660;
- 16 GB di RAM

Questa combinazione di componenti è stata scelta per garantire prestazioni affidabili e rappresentative nell'ambito delle attività di compressione.

8.2 Nomenclatura esperimenti svolti

Nel corso delle nostre ricerche e dei vari esperimenti condotti, abbiamo adottato diversi nomi per distinguere e identificare in modo chiaro i diversi aspetti delle nostre indagini. Per riferirci ai codec video e ai dataset forniti dai nostri stimati colleghi, abbiamo deciso di utilizzare l'anno 2022 come riferimento nei nomi degli esperimenti.

Per quanto riguarda i dataset e i codec video scelti e utilizzati nei nostri studi, abbiamo adottato l'anno 2023 come punto di riferimento nei nomi assegnati a tali risorse.

8.3 Risultati ottenuti

Di seguito riportiamo i vari esperimenti da noi proposti. Abbiamo scelto di dividere i risultati in vari esperimenti per avere una visione più completa sia sul lavoro sia dei nostri predecessori che sulle nostre proposte.

Per ogni studio riportiamo una tabella che riassume tutti i dati dell'esperimento che rappresenta visivamente l'andamento dei vari codec sui vari dataset.

8.3.1 Codec 2022 sui dataset 2022

Questo studio è stato svolto per testare nuovamente i risultati dei nostri predecessori per riconfermare i risultati dopo aver apportato modifiche al codice sorgente.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
ArtGallery2	HEVC	1.943	20.48s	45927441
ArtGallery2	VP9	1.645	33.54s	54234262
ArtGallery2	AV1	1.772	1791.34s	50358098
ArtGallery2	FFV1	1.365	1.96s	65351370
ArtGallery2	HUFFYUV	1.057	0.28s	84401008
ArtGallery2	UTVIDEO	1.089	0.36s	81902122

Tabella 8.1: Studio codec 2022 sul dataset "ArtGallery"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dragons	HEVC	1.804	3.55s	6538887
Dragons	VP9	1.450	10.27s	8132950
Dragons	AV1	1.514	251.60s	7789378
Dragons	FFV1	1.329	0.44s	8875630
Dragons	HUFFYUV	0.747	0.12s	15779892
Dragons	UTVIDEO	0.882	0.14s	13369862

Tabella 8.2: Studio codec 2022 sul dataset "Dragons and bunnies"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
OpEx	HEVC	1.130	47.13s	65274281
OpEx	VP9	1.094	56.67s	67410595
OpEx	AV1	1.240	2519.93s	59501300
OpEx	FFV1	1.596	7.92s	46216186
OpEx	HUFFYUV	0.424	5.35s	173654640
OpEx	UTVIDEO	0.676	5.19s	109078914

Tabella 8.3: Studio codec 2022 sul dataset "OpEx Room"

8.3.2 Codec 2022 sui dataset 2022 (random)

Questo studio è stato svolto per testare i codec 2022 sui dataset 2022, però con i frame disposti in maniera casuale (random).

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
ArtGallery2_random	HEVC	1.385	24.83s	64408395
ArtGallery2_random	VP9	1.142	37.77s	78145982
ArtGallery2_random	AV1	1.457	1477.19s	61221006
ArtGallery2_random	FFV1	1.365	1.93s	65370246
ArtGallery2_random	HUFFYUV	1.057	0.27s	84401008
ArtGallery2_random	UTVIDEO	1.089	0.35s	81902122

Tabella 8.4: Studio codec 2022 sul dataset **random** "ArtGallery"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dragons_random	HEVC	1.553	3.51s	7595776
Dragons_random	VP9	1.310	11.40s	9006578
Dragons_random	AV1	1.432	244.98s	8234601
Dragons_random	FFV1	1.329	0.44s	8878232
Dragons_random	HUFFYUV	0.747	0.11s	15779892
Dragons_random	UTVIDEO	0.882	0.13s	13369862

Tabella 8.5: Studio codec 2022 sul dataset **random** "Dragons and bunnies"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
OpEx_random	HEVC	1.001	54.78s	73701921
OpEx_random	VP9	1.037	69.50s	71118193
OpEx_random	AV1	1.161	2797.33s	63524638
OpEx_random	FFV1	1.507	7.95s	48967364
OpEx_random	HUFFYUV	0.393	5.64s	187652124
OpEx_random	UTVIDEO	0.628	4.99s	117380518

Tabella 8.6: Studio codec 2022 sul dataset **random** "OpEx Room"

8. ANALISI DEI RISULTATI

8.3.3 Codec 2022 sui dataset 2023

Questo studio è stato svolto per testare i codec 2022 sui nuovi dataset 2023 implementati da noi. L'obiettivo dietro a questo esperimento è di ottenere una quantità maggiore di dati per avere un'idea più completa sui codec 2022.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Blob	HEVC	2.644	16.23s	20286094
Blob	VP9	1.123	46.23s	47745505
Blob	AV1	2.312	1425.33s	23199679
Blob	FFV1	1.642	1.46s	32667866
Blob	HUFFYUV	0.823	0.28s	65166640
Blob	UTVIDEO	1.002	0.36s	53500422

Tabella 8.7: Studio codec 2022 sul dataset "Blob"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Car	HEVC	2.149	15.36s	29547917
Car	VP9	1.408	32.35s	45082057
Car	AV1	1.753	2647.55s	36207643
Car	FFV1	1.756	2.33s	36146340
Car	HUFFYUV	0.970	0.33s	65438568
Car	UTVIDEO	1.194	0.51s	53141066

Tabella 8.8: Studio codec 2022 sul dataset "Car"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Cobblestone	HEVC	2.044	21.64s	50287099
Cobblestone	VP9	1.232	39.56s	83437410
Cobblestone	AV1	2.006	1597.50s	51230047
Cobblestone	FFV1	1.365	2.54s	75263866
Cobblestone	HUFFYUV	1.028	0.26s	99919324
Cobblestone	UTVIDEO	1.052	0.34s	97681582

Tabella 8.9: Studio codec 2022 sul dataset "Cobblestone"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dice	HEVC	1.007	2.93s	3108647
Dice	VP9	0.927	8.07s	3376922
Dice	AV1	0.990	95.82s	3161975
Dice	FFV1	1.447	0.22s	2163376
Dice	HUFFYUV	0.289	0.09s	10826588
Dice	UTVIDEO	0.469	0.11s	6663838

Tabella 8.10: Studio codec 2022 sul dataset "Dice"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Fish	HEVC	2.109	2.46s	3487710
Fish	VP9	1.726	5.93s	4261923
Fish	AV1	1.946	125.60s	3780963
Fish	FFV1	1.227	0.36s	5992874
Fish	HUFFYUV	0.518	0.12s	14194972
Fish	UTVIDEO	0.658	0.13s	11167262

Tabella 8.11: Studio codec 2022 sul dataset "Fish"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Mannequin	HEVC	1.099	25.17s	72752630
Mannequin	VP9	1.085	40.57s	73663937
Mannequin	AV1	1.263	1962.75s	63306464
Mannequin	FFV1	1.561	1.70s	51220344
Mannequin	HUFFYUV	0.921	0.28s	86783324
Mannequin	UTVIDEO	0.996	0.41s	80218574

Tabella 8.12: Studio codec 2022 sul dataset "Mannequin"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Messerschmitt	HEVC	1.370	4.02s	6462662
Messerschmitt	VP9	0.973	11.24s	9101952
Messerschmitt	AV1	1.089	201.96s	8127707
Messerschmitt	FFV1	1.720	0.37s	5149332
Messerschmitt	HUFFYUV	0.753	0.11s	11756164
Messerschmitt	UTVIDEO	1.027	0.13s	8617930

Tabella 8.13: Studio codec 2022 sul dataset "Messerschmitt"

8. ANALISI DEI RISULTATI

8.3.4 Codec 2022 sui dataset 2023 (random)

Questo studio è stato svolto per testare i codec 2022 sui nuovi dataset 2023 implementati da noi andando a randomizzare i frames dei dataset scelti da noi per testare se c'è correlazione con lo studio precedente.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Blob_random	HEVC	1.681	19.63	31917331
Blob_random	VP9	1.032	42.06s	51991103
Blob_random	AV1	1.963	934.95s	27323634
Blob_random	FFV1	1.641	1.71s	32689958
Blob_random	HUFFYUV	0.823	0.28s	65166640
Blob_random	UTVIDEO	1.002	0.36s	53500422

Tabella 8.14: Studio codec 2022 sul dataset **random** "Blob"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Car_random	HEVC	1.401	22.08s	45307730
Car_random	VP9	1.010	41.80s	62818936
Car_random	AV1	1.319	2314.73s	48134236
Car_random	FFV1	1.755	1.51s	36179160
Car_random	HUFFYUV	0.970	0.28s	65438568
Car_random	UTVIDEO	1.194	0.37s	53141066

Tabella 8.15: Studio codec 2022 sul dataset **random** "Car"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Cobblestone_random	HEVC	1.387	20.03s	74081960
Cobblestone_random	VP9	0.943	40.91s	108910835
Cobblestone_random	AV1	1.601	1118.97s	64174845
Cobblestone_random	FFV1	1.365	2.59s	75277548
Cobblestone_random	HUFFYUV	1.028	0.29s	99919324
Cobblestone_random	UTVIDEO	1.052	0.36s	97681582

Tabella 8.16: Studio codec 2022 sul dataset **random** "Cobblestone"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dice_random	HEVC	0.944	2.51s	3317237
Dice_random	VP9	0.889	8.05s	3520415
Dice_random	AV1	0.976	97.59s	3207578
Dice_random	FFV1	1.446	0.26s	2164950
Dice_random	HUFFYUV	0.289	0.13s	10826588
Dice_random	UTVIDEO	0.469	0.12s	6663838

Tabella 8.17: Studio codec 2022 sul dataset **random** "Dice"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Fish_random	HEVC	1.934	2.54s	3803198
Fish_random	VP9	1.6107	6.52s	4567987
Fish_random	AV1	1.852	127.43s	3972707
Fish_random	FFV1	1.227	0.32s	5994606
Fish_random	HUFFYUV	0.518	0.11s	14194972
Fish_random	UTVIDEO	0.658	0.13s	11167262

Tabella 8.18: Studio codec 2022 sul dataset **random** "Fish"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Mannequin_random	HEVC	0.979	31.77s	81604469
Mannequin_random	VP9	0.918	44.22s	87025029
Mannequin_random	AV1	1.164	1462.18s	68689785
Mannequin_random	FFV1	1.561	1.73s	51228310
Mannequin_random	HUFFYUV	0.921	0.29s	86783324
Mannequin_random	UTVIDEO	0.996	0.34s	80218574

Tabella 8.19: Studio codec 2022 sul dataset **random** "Mannaquin"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Messerschmitt_random	HEVC	1.060	4.16s	8355409
Messerschmitt_random	VP9	0.846	15.90s	10464728
Messerschmitt_random	AV1	1.037	195.90s	8537404
Messerschmitt_random	FFV1	1.719	0.30s	5152306
Messerschmitt_random	HUFFYUV	0.753	0.11s	11756164
Messerschmitt_random	UTVIDEO	1.027	0.13s	8617930

Tabella 8.20: Studio codec 2022 sul dataset **random** "Massershmitt"

8.3.5 Codec 2023 sui dataset 2022

Questo studio è stato svolto per testare i codec scelti da noi sui dataset 2022 per vedere l'andamento dei codec su i dataset scelti dai nostri predecessori.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
ArtGallery2	FLV1	252.343	0.43s	353725
ArtGallery2	CLJR	1.27844	0.38s	69819386
ArtGallery2	MPEG4	328.628	0.37s	271614
ArtGallery2	MJPEG	32.108	0.68s	2779974
ArtGallery2	ProRes	2.160	2.67s	41309465
ArtGallery2	MagicYUV	1.035	0.37s	86179404
ArtGallery2	FFVHUFF	1.057	0.31s	84401008
ArtGallery2	LCL	0.755	1.42s	118074884

Tabella 8.21: Studio codec 2023 sul dataset "ArtGallery"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dragons	FLV1	52.111	0.14s	226438
Dragons	CLJR	0.910	0.12s	12957430
Dragons	MPEG4	55.658	0.12s	212008
Dragons	MJPEG	12.455	0.18s	947362
Dragons	ProRes	1.693	0.81s	6966989
Dragons	MagicYUV	0.871	0.15s	13538438
Dragons	FFVHUFF	0.747	0.11s	15779892
Dragons	LCL	0.861	0.25s	13689098

Tabella 8.22: Studio codec 2023 sul dataset "Dragons and bunnies"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
OpEx	FLV1	45.311	5.03s	1628729
OpEx	MPEG4	61.213	4.76s	1205634
OpEx	MJPEG	15.666	6.12s	4710772
OpEx	ProRes	0.990	8.32s	74538074
OpEx	MagicYUV	0.667	5.00s	110552114
OpEx	FFVHUFF	0.424	5.82s	173654640
OpEx	LCL	0.687	5.89s	107326448

Tabella 8.23: Studio codec 2023 sul dataset "OpEx"

8.3.6 Codec 2023 sui dataset 2022 (random)

Questo studio è stato svolto per testare i codec scelti da noi sui dataset 2022 con i frames disposti in modo random per vedere l'andamento delle prestazioni sul precedente esperimento.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
ArtGallery2_random	FLV1	126.103	0.47s	707833
ArtGallery2_random	CLJR	1.278	0.33s	69819386
ArtGallery2_random	MPEG4	182.970	0.37s	487838
ArtGallery2_random	MJPEG	32.126	0.54s	2778390
ArtGallery2_random	ProRes	2.160	2.75s	41309465
ArtGallery2_random	MagicYUV	1.035	0.36s	86179404
ArtGallery2_random	FFVHUFF	1.057	0.28s	84401008
ArtGallery2_random	LCL	0.755	1.44s	118074884

Tabella 8.24: Studio codec 2023 sul dataset **random** "ArtGallery"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dragons_random	FLV1	43.079	0.16s	273912
Dragons_random	CLJR	0.910	0.12s	12957430
Dragons_random	MPEG4	46.142	0.13s	255730
Dragons_random	MJPEG	12.458	0.19s	947132
Dragons_random	ProRes	1.693	0.82s	6966989
Dragons_random	MagicYUV	0.871	0.15s	13538266
Dragons_random	FFVHUFF	0.747	0.11s	15779892
Dragons_random	LCL	0.861	0.28s	13689098

Tabella 8.25: Studio codec 2023 sul dataset **random** "Dragons and bunnies"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
OpEx_random	FLV1	15.548	5.16s	4746581
OpEx_random	MPEG4	30.284	4.29s	2436928
OpEx_random	MJPEG	14.859	5.66s	4966664
OpEx_random	ProRes	0.918	9.06s	80383530
OpEx_random	MagicYUV	0.620	4.58s	118868546
OpEx_random	FFVHUFF	0.393	5.48s	187652124
OpEx_random	LCL	0.654	5.76s	112813424
OpEx_random	CLJR	Non Funziona	-	-

Tabella 8.26: Studio codec 2023 sul dataset **random** "OpEx"

8.3.7 Codec 2023 sui dataset 2023

Questo studio è stato svolto per testare i codec scelti da noi sui dataset aggiunti da noi (2023)

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Blob	FLV1	151.217	0.44s	354826
Blob	CLJR	0.768	0.34s	69819390
Blob	MPEG4	192.188	0.37s	279184
Blob	MJPEG	22.960	0.56s	2336916
Blob	ProRes	1.336	2.35s	40166200
Blob	MagicYUV	0.965	0.36s	55591710
Blob	FFVHUFF	0.823	0.25s	65166640
Blob	LCL	0.682	1.15s	78693570

Tabella 8.27: Studio codec 2023 sul dataset "Blob"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Car	FLV1	181.836	0.37s	349220
Car	CLJR	0.909	0.32s	69819390
Car	MPEG4	214.747	0.30s	295700
Car	MJPEG	24.035	0.59s	2641968
Car	ProRes	1.582	2.54s	40129240
Car	MagicYUV	1.127	0.38s	56365060
Car	FFVHUFF	0.970	0.33s	65438570
Car	LCL	0.897	1.93s	70786740

Tabella 8.28: Studio codec 2023 sul dataset "Car"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Cobblestone	FLV1	182.216	0.56s	564142
Cobblestone	CLJR	1.472	0.35s	69819390
Cobblestone	MPEG4	211.364	0.42s	486344
Cobblestone	MJPEG	23.859	0.59s	4308462
Cobblestone	ProRes	2.503	3.96s	41075260
Cobblestone	MagicYUV	0.994	0.37s	103389700
Cobblestone	FFVHUFF	1.029	0.27s	99919320
Cobblestone	LCL	0.725	1.06s	141763200

Tabella 8.29: Studio codec 2023 sul dataset "Cobblestone"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dice	FLV1	15.473	0.32s	202386
Dice	CLJR	0.251	0.11s	12459290
Dice	MPEG4	17.873	0.12s	175208
Dice	MJPEG	4.971	0.17s	630020
Dice	ProRes	0.583	0.59s	5370439
Dice	MagicYUV	0.437	0.14s	7159920
Dice	FFVHUFF	0.289	0.10s	10826590
Dice	LCL	0.979	0.15s	3199498

Tabella 8.30: Studio codec 2023 sul dataset "Dice"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Fish	FLV1	38.966	0.17s	188832
Fish	CLJR	0.591	0.12s	12459290
Fish	MPEG4	40.741	0.13s	180606
Fish	MJPEG	8.866	0.18s	829886
Fish	ProRes	1.558	0.80s	4721496
Fish	MagicYUV	0.506	0.18s	14544920
Fish	FFVHUFF	0.518	0.13s	14194970
Fish	LCL	0.722	0.19s	10197960

Tabella 8.31: Studio codec 2023 sul dataset "Fish"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Mannequin	FLV1	188.561	0.44s	424100
Mannequin	CLJR	1.145	0.35s	69819390
Mannequin	MPEG4	226.555	0.36s	352978
Mannequin	MJPEG	24.138	0.60s	3312942
Mannequin	ProRes	1.958	2.67s	40834080
Mannequin	MagicYUV	0.972	0.37s	82240410
Mannequin	FFVHUFF	0.921	0.29s	86783320
Mannequin	LCL	0.717	1.02s	111512600

Tabella 8.32: Studio codec 2023 sul dataset "Mannequin"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Messerschmitt	FLV1	29.958	0.21s	295716
Messerschmitt	CLJR	0.711	0.12s	12459290
Messerschmitt	MPEG4	38.265	0.13s	231522
Messerschmitt	MJPEG	12.793	0.17s	692506
Messerschmitt	ProRes	1.175	0.57s	7539665
Messerschmitt	MagicYUV	0.974	0.19s	9096656
Messerschmitt	FFVHUFF	0.754	0.14s	11756160
Messerschmitt	LCL	1.004	0.26s	8822124

Tabella 8.33: Studio codec 2023 sul dataset "Messerschmitt"

8.3.8 Codec 2023 sui dataset 2023 (random)

Questo studio è stato svolto per testare i codec scelti da noi sui aggiunti da noi (2023) con i frames disposti in modo random per vedere l'andamento delle prestazioni sul precedente esperimento.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Blob_random	FLV1	38.392	0.46s	1397587
Blob_random	CLJR	0.768	0.33s	69819390
Blob_random	MPEG4	75.080	0.35s	714648
Blob_random	MJPEG	22.968s	0.52s	2336130
Blob_random	ProRes	1.336	2.20s	40166200
Blob_random	MagicYUV	0.965	0.38s	55591710
Blob_random	FFVHUFF	0.823	0.27s	65166640
Blob_random	LCL	0.682	1.15s	78693570

Tabella 8.34: Studio codec 2023 sul dataset **random** "Blob"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Car_random	FLV1	63.584	16.58s	998682
Car_random	CLJR	0.909	0.32s	69819390
Car_random	MPEG4	94.757	0.31s	670142
Car_random	MJPEG	24.046	0.55s	2640814
Car_random	ProRes	1.582	2.57s	40129240
Car_random	MagicYUV	1.127	0.38s	56365060
Car_random	FFVHUFF	0.970	0.29s	65438570
Car_random	LCL	0.897	1.83s	70786740

Tabella 8.35: Studio codec 2023 sul dataset **random** "Car"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Cobblestone_random	FLV1	64.334	0.49s	1597836
Cobblestone_random	CLJR	1.472	0.36s	69819390
Cobblestone_random	MPEG4	108.435	0.35s	947992
Cobblestone_random	MJPEG	23.859	0.57s	4308444
Cobblestone_random	ProRes	2.503	3.96s	41075260
Cobblestone_random	MagicYUV	0.994	0.36s	103389700
Cobblestone_random	FFVHUFF	1.029	0.28s	99919320
Cobblestone_random	LCL	0.725	1.09s	141763200

Tabella 8.36: Studio codec 2023 sul dataset **random** "Cobblestone"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Dice_random	FLV1	13.044	0.18s	240076
Dice_random	CLJR	0.251	0.13s	12459290
Dice_random	MPEG4	14.843	0.16s	210984
Dice_random	MJPEG	5.024	0.17s	623306
Dice_random	ProRes	0.583	0.60s	5370439
Dice_random	MagicYUV	0.438	0.13s	7147860
Dice_random	FFVHUFF	0.289	0.10s	10826590
Dice_random	LCL	0.979	0.15s	3199498

Tabella 8.37: Studio codec 2023 sul dataset **random** "Dice"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Fish_random	FLV1	34.613	0.15s	212580
Fish_random	CLJR	0.591	0.12s	12459290
Fish_random	MPEG4	35.554	0.14s	206952
Fish_random	MJPEG	8.867	0.19s	829778
Fish_random	ProRes	1.558	0.77s	4721496
Fish_random	MagicYUV	0.506	0.15s	14549420
Fish_random	FFVHUFF	0.518	0.13s	14194970
Fish_random	LCL	0.722	0.21s	10197960

Tabella 8.38: Studio codec 2023 sul dataset **random** "Fish"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Mannequin_random	FLV1	79.886	0.49s	1001032
Mannequin_random	CLJR	1.145	0.31s	69819390
Mannequin_random	MPEG4	107.542	0.35s	743604
Mannequin_random	MJPEG	24.138	0.59s	3312976
Mannequin_random	ProRes	1.958	2.69s	40834080
Mannequin_random	MagicYUV	0.972	0.38s	82240410
Mannequin_random	FFVHUFF	0.921	0.30s	86783320
Mannequin_random	LCL	0.717	0.99s	111512600

Tabella 8.39: Studio codec 2023 sul dataset **random** "Mannequin"

Dataset	Algoritmo	Rapporto compressione	Tempo compressione	Dimensione finale
Messerschmitt_random	FLV1	24.308	0.22s	364455
Messerschmitt_random	CLJR	0.711	0.13s	12459290
Messerschmitt_random	MPEG4	29.915	0.16s	296142
Messerschmitt_random	MJPEG	12.910	0.20s	686216
Messerschmitt_random	ProRes	1.175	0.58s	7539665
Messerschmitt_random	MagicYUV	0.974	0.15s	9096140
Messerschmitt_random	FFVHUFF	0.754	0.13s	11756160
Messerschmitt_random	LCL	1.004	0.26s	8822124

Tabella 8.40: Studio codec 2023 sul dataset **random** "Messerschmitt"

8.4 Decompressione codec 2023 su Dataset 2022

Questo studio è stato svolto per testare la decompressione per i codec scelti da noi sui dataset 2022.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Average SSIM	Average PSNR
ArtGallery2	FLV1	0.917	33.606
ArtGallery2	CLJR	0.846	34.976
ArtGallery2	MPEG4	0.911	33.352
ArtGallery2	MJPEG	0.926	34.946
ArtGallery2	ProRes	0.992	48.100
ArtGallery2	MagicYUV	1.000	inf
ArtGallery2	FFVHUFF	1.000	inf
ArtGallery2	LCL	1.000	inf

Tabella 8.41: Studio decompressione codec 2023 sul dataset "Art Gallery"

Dataset	Algoritmo	Average SSIM	Average PSNR
Dragons	FLV1	0.904	32.887
Dragons	CLJR	0.883	35.456
Dragons	MPEG4	0.901	32.779
Dragons	MJPEG	0.902	32.880
Dragons	ProRes	0.991	44.418
Dragons	MagicYUV	1.000	inf
Dragons	FFVHUFF	1.000	inf
Dragons	LCL	1.000	inf

Tabella 8.42: Studio decompressione codec 2023 sul dataset "Dragons and bunnies"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Average SSIM	Average PSNR
OpEx	FLV1	0.975	41.343
OpEx	MPEG4	0.968	40.463
OpEx	MJPEG	0.985	47.427
OpEx	ProRes	0.999	58.536
OpEx	MagicYUV	0.999	inf
OpEx	FFVHUFF	0.999	inf
OpEx	LCL	0.999	inf

Tabella 8.43: Studio decompressione codec 2023 sul dataset "OpEx"

8.5 Decompressione codec 2023 su Dataset 2022 (random)

Questo studio è stato svolto per testare la decompressione per i codec scelti da noi sui dataset 2022, dispondendo i frames in modo randomico.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Average SSIM	Average PSNR
ArtGallery2_random	FLV1	0.909	32.747
ArtGallery2_random	CLJR	0.846	34.975
ArtGallery2_random	MPEG4	0.900	32.377
ArtGallery2_random	MJPEG	0.926	34.942
ArtGallery2_random	ProRes	0.992	48.100
ArtGallery2_random	MagicYUV	1.000	inf
ArtGallery2_random	FFVHUFF	1.000	inf
ArtGallery2_random	LCL	1.000	inf

Tabella 8.44: Studio decompressione codec 2023 sul dataset **random** "Art Gallery"

Dataset	Algoritmo	Average SSIM	Average PSNR
Dragons_random	FLV1	0.900	32.601
Dragons_random	CLJR	0.883	35.456
Dragons_random	MPEG4	0.896	32.492
Dragons_random	MJPEG	0.902	32.876
Dragons_random	ProRes	0.991	44.418
Dragons_random	MagicYUV	1.000	inf
Dragons_random	FFVHUFF	1.000	inf
Dragons_random	LCL	1.000	inf

Tabella 8.45: Studio decompressione codec 2023 sul dataset **random** "Dragons and bunnies"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Average SSIM	Average PSNR
OpEx_random	FLV1	0.976	41.997
OpEx_random	MPEG4	0.964	40.285
OpEx_random	MJPEG	0.985	47.379
OpEx_random	ProRes	0.999	58.789
OpEx_random	MagicYUV	0.999	inf
OpEx_random	FFVHUFF	0.999	inf
OpEx_random	LCL	0.999	inf

Tabella 8.46: Studio decompressione codec 2023 sul dataset **random** "OpEx"

8.6 Decompressione codec 2023 su Dataset 2023

Questo studio è stato svolto per testare la decompressione per i codec scelti da noi sui dataset aggiunti da noi.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Average SSIM	Average PSNR
Blob	FLV1	0.902	35.263
Blob	CLJR	0.856	35.100
Blob	MPEG4	0.893	34.818
Blob	MJPEG	0.930	37.438
Blob	ProRes	0.999	56.568
Blob	MagicYUV	1.000	inf
Blob	FFVHUFF	1.000	inf
Blob	LCL	1.000	inf

Tabella 8.47: Studio decompressione codec 2023 sul dataset "Blob"

Dataset	Algoritmo	Average SSIM	Average PSNR
Car	FLV1	0.930	33.856
Car	CLJR	0.837	34.938
Car	MPEG4	0.924	33.562
Car	MJPEG	0.944	35.055
Car	ProRes	0.998	52.979
Car	MagicYUV	1.000	inf
Car	FFVHUFF	1.000	inf
Car	LCL	1.000	inf

Tabella 8.48: Studio decompressione codec 2023 sul dataset "Car"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Average SSIM	Average PSNR
Cobblestone	FLV1	0.813	28.567
Cobblestone	CLJR	0.911	34.906
Cobblestone	MPEG4	0.809	28.486
Cobblestone	MJPEG	0.842	29.006
Cobblestone	ProRes	0.992	41.253
Cobblestone	MagicYUV	1.000	inf
Cobblestone	FFVHUFF	1.000	inf
Cobblestone	LCL	1.000	inf

Tabella 8.49: Studio decompressione codec 2023 sul dataset "Cobblestone"

Dataset	Algoritmo	Average SSIM	Average PSNR
Dice	FLV1	0.949	34.733
Dice	CLJR	0.815	35.248
Dice	MPEG4	0.948	34.679
Dice	MJPEG	0.957	36.754
Dice	ProRes	0.998	52.176
Dice	MagicYUV	1.000	inf
Dice	FFVHUFF	1.000	inf
Dice	LCL	1.000	inf

Tabella 8.50: Studio decompressione codec 2023 sul dataset "Dice"

Dataset	Algoritmo	Average SSIM	Average PSNR
Fish	FLV1	0.960	35.214
Fish	CLJR	0.972	39.840
Fish	MPEG4	0.959	35.133
Fish	MJPEG	0.958	35.093
Fish	ProRes	0.996	45.356
Fish	MagicYUV	1.000	inf
Fish	FFVHUFF	1.000	inf
Fish	LCL	1.000	inf

Tabella 8.51: Studio decompressione codec 2023 sul dataset "Fish"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Average SSIM	Average PSNR
Mannequin	FLV1	0.854	30.767
Mannequin	CLJR	0.881	35.292
Mannequin	MPEG4	0.848	30.647
Mannequin	MJPEG	0.884	31.992
Mannequin	ProRes	0.996	47.907
Mannequin	MagicYUV	1.000	inf
Mannequin	FFVHUFF	1.000	inf
Mannequin	LCL	1.000	inf

Tabella 8.52: Studio decompressione codec 2023 sul dataset "Mannequin"

Dataset	Algoritmo	Average SSIM	Average PSNR
Messerschmitt	FLV1	0.853	32.642
Messerschmitt	CLJR	0.883	34.993
Messerschmitt	MPEG4	0.852	32.597
Messerschmitt	MJPEG	0.875	33.758
Messerschmitt	ProRes	0.996	50.284
Messerschmitt	MagicYUV	1.000	inf
Messerschmitt	FFVHUFF	1.000	inf
Messerschmitt	LCL	1.000	inf

Tabella 8.53: Studio decompressione codec 2023 sul dataset "Messerschmitt"

8.7 Decompressione codec 2023 su Dataset 2023 (random)

Questo studio è stato svolto per testare la decompressione per i codec scelti da noi sui dataset aggiunti da noi, disponendo i frame in modo randomico.

Di seguito riportiamo le tabelle (una per dataset) su questo studio:

Dataset	Algoritmo	Average SSIM	Average PSNR
Blob_random	FLV1	0.881	33.923
Blob_random	CLJR	0.856	35.101
Blob_random	MPEG4	0.864	33.137
Blob_random	MJPEG	0.930	37.432
Blob_random	ProRes	0.999	56.568
Blob_random	MagicYUV	1.000	inf
Blob_random	FFVHUFF	1.000	inf
Blob_random	LCL	1.000	inf

Tabella 8.54: Studio decompressione codec 2023 sul dataset **random** "Blob"

Dataset	Algoritmo	Average SSIM	Average PSNR
Car_random	FLV1	0.913	32.542
Car_random	CLJR	0.837	34.938
Car_random	MPEG4	0.902	32.054
Car_random	MJPEG	0.944	35.053
Car_random	ProRes	0.998	52.979
Car_random	MagicYUV	1.000	inf
Car_random	FFVHUFF	1.000	inf
Car_random	LCL	1.000	inf

Tabella 8.55: Studio decompressione codec 2023 sul dataset **random** "Car"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Average SSIM	Average PSNR
Cobblestone_random	FLV1	0.783	27.515
Cobblestone_random	CLJR	0.911	34.906
Cobblestone_random	MPEG4	0.775	27.295
Cobblestone_random	MJPEG	0.842	29.006
Cobblestone_random	ProRes	0.992	41.253
Cobblestone_random	MagicYUV	1.000	inf
Cobblestone_random	FFVHUFF	1.000	inf
Cobblestone_random	LCL	1.000	inf

Tabella 8.56: Studio decompressione codec 2023 sul dataset **random** "Cobblestone"

Dataset	Algoritmo	Average SSIM	Average PSNR
Dice_random	FLV1	0.947	34.549
Dice_random	CLJR	0.815	35.248
Dice_random	MPEG4	0.945	34.556
Dice_random	MJPEG	0.957	36.673
Dice_random	ProRes	0.998	52.176
Dice_random	MagicYUV	1.000	inf
Dice_random	FFVHUFF	1.000	inf
Dice_random	LCL	1.000	inf

Tabella 8.57: Studio decompressione codec 2023 sul dataset **random** "Dice"

Dataset	Algoritmo	Average SSIM	Average PSNR
Fish_random	FLV1	0.957	34.747
Fish_random	CLJR	0.972	39.837
Fish_random	MPEG4	0.956	34.619
Fish_random	MJPEG	0.958	35.084
Fish_random	ProRes	0.996	45.356
Fish_random	MagicYUV	1.000	inf
Fish_random	FFVHUFF	1.000	inf
Fish_random	LCL	1.000	inf

Tabella 8.58: Studio decompressione codec 2023 sul dataset **random** "Fish"

8. ANALISI DEI RISULTATI

Dataset	Algoritmo	Average SSIM	Average PSNR
Mannequin_random	FLV1	0.827	29.560
Mannequin_random	CLJR	0.881	35.292
Mannequin_random	MPEG4	0.819	29.417
Mannequin_random	MJPEG	0.884	31.991
Mannequin_random	ProRes	0.996	47.907
Mannequin_random	MagicYUV	1.000	inf
Mannequin_random	FFVHUFF	1.000	inf
Mannequin_random	LCL	1.000	inf

Tabella 8.59: Studio decompressione codec 2023 sul dataset **random** "Mannequin"

Dataset	Algoritmo	Average SSIM	Average PSNR
Messerschmitt_random	FLV1	0.837	32.097
Messerschmitt_random	CLJR	0.883	34.993
Messerschmitt_random	MPEG4	0.831	31.931
Messerschmitt_random	MJPEG	0.873	33.755
Messerschmitt_random	ProRes	0.996	50.284
Messerschmitt_random	MagicYUV	1.000	inf
Messerschmitt_random	FFVHUFF	1.000	inf
Messerschmitt_random	LCL	1.000	inf

Tabella 8.60: Studio decompressione codec 2023 sul dataset **random** "Messerschmitt"

8.8 Analisi riassuntiva dei risultati

Durante l'ampia fase di testing condotta, abbiamo osservato risultati distinti in termini di rapporto di compressione e decompressione tra vari codec. Inoltre l'introduzione di un indice di similarità come il PSNR ha contribuito significativamente a migliorare la nostra comprensione dei risultati ottenuti durante la fase di decompressione.

In particolare, è emerso che i codec **FLV1** e **MPEG4** hanno dimostrato di ottenere ottimi risultati nella compressione, presentando un rapporto di compressione superiore rispetto agli altri codec considerati (compressione rate oltre i 250). Nonostante si tratti di codec lossy, mostrano ottime performance tra i codec testati e godono di buoni valori di similarità secondo i due incidi usati (PSNR E SSIM), il che li rende molto affidabili in fase di decompressione.

Per quanto riguarda i codec lossless utilizzati non abbiamo riscontrato miglioramenti rispetto ai risultati dei nostri colleghi nel precedente studio. Infatti in tutti gli esperimenti effettuati abbiamo riscontrato bassi valori di compressione.

Un interessante scoperta è stato il codec **ProRes** emerso dopo la grande quantità di esperimenti condotti mostrando prestazioni tendenti a un codec lossless (ovvero qualità eccellente e discreto rapporto di compressione).

Infatti nel contesto della decompressione, è rilevante notare che il codec ProRes, nonostante sia lossy, ha manifestato non solo un notevole valore di SSIM pari a 0,99, indicante una fedeltà estremamente alta nella ricostruzione dell'immagine dopo il processo di decompressione, ma ha anche presentato un indice PSNR costantemente ottimo (sempre superiore a 40 punti). L'alto valore di PSNR conferma ulteriormente la capacità del codec ProRes di mantenere una qualità visiva elevata, anche considerando il processo di compressione e decompressione. Pertanto, il codec ProRes si rivela un'opzione robusta quando la conservazione della qualità dell'immagine è di primaria importanza nelle applicazioni di compressione video.

8.8.1 Analisi sui dataset random

Di seguito analizziamo le prestazioni dei vari codec di compressione sui rispettivi dataset randomizzati. In generale abbiamo notato che per ogni esperimento c'è un peggioramento importante su alcuni codec mentre altri sembrano soffrire di meno la disposizione dei frames in ordine randomico. Nello specifico i codec che peggiorano la loro performance, sia in termini di velocità di esecuzione e sia in termini di compression rate:

- HEVC
- VP9
- FLV1
- MPEG4

Per i codec non citati sopra, le prestazioni risultano essere molto vicine a quelle sui dataset originali, anche se, in alcuni casi, di poco peggiori.

Si può quindi affermare che **c'è effettivamente dipendenza** tra i frame dei vari Light Field. Quindi l'approccio di compressione/decompressione degli stessi, deve tener conto di questa dipendenza tra i frame per ottenere una compressione efficiente senza compromettere la qualità dell'immagine riprodotta. Gli esperimenti condotti sui dataset con frame disposti casualmente hanno evidenziato dei peggioramenti sia in fase di compressione, in termini di tempi e rapporto di compressione, sia in fase di decompressione, in termini di similarità e rapporto segnale/rumore.

CAPITOLO 9

CONCLUSIONE E SVILUPPI FUTURI

La conclusione dello studio evidenzia l'importanza cruciale della scelta del dataset e del codec nella compressione dei light field images, con rilevanti impatti sulle prestazioni complessive. L'analisi ha mostrato variazioni significative nell'efficacia della compressione in base al dataset utilizzato, con l'ordine dei frame che può influenzare la qualità e le performance, infine, la decisione tra codec lossy e lossless è fondamentale.

Il codec Prores ha dimostrato di avere un interessante equilibrio tra il rapporto di compressione, la perdita di informazioni e la qualità visiva. Nonostante il suo rapporto di compressione potrebbe non essere ottimo rispetto ad alcuni codec altamente efficienti in termini di riduzione delle dimensioni, il ProRes compensa questo aspetto con eccellenti indici SSIM e PSNR, nonostante la sua natura lossy.

D'altra parte, codec come MPEG4 e FLV1 dimostrano un equilibrio ben calibrato in tutte le fasi, presentando ottimi rapporti di compressione insieme a qualità visiva e fedeltà strutturale notevoli. Questi codec si distinguono per essere efficienti in termini di spazio di archiviazione senza compromettere eccessivamente la qualità dell'immagine, nonostante la loro natura lossy.

Questi esempi evidenziano la complessità delle scelte nel campo della compressione video, poiché diversi codec possono eccellere in modi differenti.

Per lo sviluppo futuro, potrebbe essere interessante esplorare ottimizzazioni specifiche per determinati dataset e valutare l'evoluzione di nuovi approcci o codec che possano offrire prestazioni superiori. Un'altra direzione potenziale potrebbe essere quella di esaminare nuove metodologie nell'ambito dei Light Field, poiché l'adattamento continuo delle tecniche di compressione alle immagini light field rimane un ambito di ricerca promettente.

RIFERIMENTI

- [1] PMF RESEARCH. *PMF RESEARCH COSA SONO GLI OLOGRAMMI? DEFINIZIONE E PROSPETTIVE*. 2022. URL: [https://pmf-research.eu/realta-aumentata-nuova-frontiera%20ologrammi/#:~:text=L%27ologrammi%3A%20storia%20ed%20evoluzione,-Gli%20ologrammi%20nascono&text=Dennis%20Gabor%2C%20famoso%20scienziato%20ungherese,\)%20e%20gramma%20\(messaggio\)..](https://pmf-research.eu/realta-aumentata-nuova-frontiera%20ologrammi/#:~:text=L%27ologrammi%3A%20storia%20ed%20evoluzione,-Gli%20ologrammi%20nascono&text=Dennis%20Gabor%2C%20famoso%20scienziato%20ungherese,)%20e%20gramma%20(messaggio)..)
- [2] Antonio Giammetta Raffaele Squillante. «Ligh Field Hologram Compression». In: (2023), p. 34.
- [3] wikipedia. *CirrusLogicAccuPak*. URL: https://wiki.multimedia.cx/index.php/Cirrus_Logic_AccuPak.
- [4] wikipedia. *FLV*. URL: https://en.wikipedia.org/wiki/Flash_Video.
- [5] wikipedia. *M-JPEG*. URL: https://it.wikipedia.org/wiki/Motion_JPEG.
- [6] wikipedia. *MPEG-4*. URL: <https://it.wikipedia.org/wiki/MPEG-4>.
- [7] wikipedia. *ProRes*. URL: https://en.wikipedia.org/wiki/Apple_ProRes.
- [8] Christoph Gerstbauer. *FFVHUFF*. URL: <https://ffmpeg-user.ffmpeg.narkive.com/kNFlDpI2/ffvhuff-codec-improvement>.
- [9] wikipedia. *zlib*. URL: https://wiki.multimedia.cx/index.php/Lossless_Codec_Libraries.
- [10] MagicYuv. *magicyuv*. URL: <https://www.magicyuv.com>.
- [11] Umme Sara, Morium Akter e Mohammad Shorif Uddin. «Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study». In: *Journal of Computer and Communications* 7.3 (2019), pp. 8–18.
- [12] Yang Cao et al. «Social-Aware Video Multicast Based on Device-to-Device Communications». In: *IEEE Transactions on Mobile Computing* 15 (giu. 2016), pp. 1528–1539. DOI: 10.1109/TMC.2015.2461214.

- [13] Hamid R. Sheikh Zhou Wang Alan C. Bovik e Eero P. Simoncelli. *The SSIM Index for Image Quality Assessment*. 2011. URL: <https://www.cns.nyu.edu/~lcv/ssim/#test>.
- [14] PMF RESEARCH. *PSNR-HVS-M*. 2009. URL: <https://www.ponomarenko.info/psnrhvs.html>.
- [15] MIT Media Lab Light Fields. *Messerschmitt*. URL: <https://web.media.mit.edu/~gordonw/SyntheticLightFields/>.
- [16] The Stanford 3D Scanning Repository. *3Dscanrep*. URL: <http://graphics.stanford.edu/data/3Dscanrep/>.
- [17] MIT Media Lab Light Fields. *Dice*. URL: <https://web.media.mit.edu/~gordonw/SyntheticLightFields/>.
- [18] MIT Media Lab Light Fields. *Fish*. URL: <https://web.media.mit.edu/~gordonw/SyntheticLightFields/>.
- [19] Max Planck Institut Informatik. *Car*. URL: <https://lightfields.mpi-inf.mpg.de/Dataset.html>.
- [20] Max Planck Institut Informatik. *Cobblestone*. URL: <https://lightfields.mpi-inf.mpg.de/Dataset.html>.
- [21] Max Planck Institut Informatik. *Mannequin*. URL: <https://lightfields.mpi-inf.mpg.de/Dataset.html>.
- [22] Max Planck Institut Informatik. *Blob*. URL: <https://lightfields.mpi-inf.mpg.de/Dataset.html>.

ELENCO DELLE FIGURE

2.1	Cattura di un ligh field	7
2.2	Informazioni racchiuse in un ligh field	7
4.1	Rapporto tra PSNR e qualità del video	14
7.1	Immagine estratta dal dataset "Messerschmitt"	23
7.2	Immagine estratta dal dataset "Dice"	23
7.3	Immagine estratta dal dataset "Fish"	24
7.4	Immagine estratta dal dataset "Car"	24
7.5	Immagine estratta dal dataset "Cobblestone"	24
7.6	Immagine estratta dal dataset "Mannequin"	25
7.7	Immagine estratta dal dataset "Blob"	25

ELENCO DELLE TABELLE

8.1	Studio codec 2022 sul dataset "ArtGallery"	27
8.2	Studio codec 2022 sul dataset "Dragons and bunnies"	27
8.3	Studio codec 2022 sul dataset "OpEx Room"	28
8.4	Studio codec 2022 sul dataset random "ArtGallery"	29
8.5	Studio codec 2022 sul dataset random "Dragons and bunnies"	29
8.6	Studio codec 2022 sul dataset random "OpEx Room"	29
8.7	Studio codec 2022 sul dataset "Blob"	30
8.8	Studio codec 2022 sul dataset "Car"	30
8.9	Studio codec 2022 sul dataset "Cobblestone"	30
8.10	Studio codec 2022 sul dataset "Dice"	31
8.11	Studio codec 2022 sul dataset "Fish"	31
8.12	Studio codec 2022 sul dataset "Mannequin"	31
8.13	Studio codec 2022 sul dataset "Messerschmitt"	31
8.14	Studio codec 2022 sul dataset random "Blob"	32
8.15	Studio codec 2022 sul dataset random "Car"	32
8.16	Studio codec 2022 sul dataset random "Cobblestone"	32
8.17	Studio codec 2022 sul dataset random "Dice"	33
8.18	Studio codec 2022 sul dataset random "Fish"	33
8.19	Studio codec 2022 sul dataset random "Mannaquin"	33
8.20	Studio codec 2022 sul dataset random "Massershmitt"	33
8.21	Studio codec 2023 sul dataset "ArtGallery"	34
8.22	Studio codec 2023 sul dataset "Dragons and bunnies"	34
8.23	Studio codec 2023 sul dataset "OpEx"	35
8.24	Studio codec 2023 sul dataset random "ArtGallery"	36
8.25	Studio codec 2023 sul dataset random "Dragons and bunnies"	36
8.26	Studio codec 2023 sul dataset random "OpEx"	37
8.27	Studio codec 2023 sul dataset "Blob"	38
8.28	Studio codec 2023 sul dataset "Car"	38

ELENCO DELLE TABELLE

8.29 Studio codec 2023 sul dataset "Cobblestone"	39
8.30 Studio codec 2023 sul dataset "Dice"	39
8.31 Studio codec 2023 sul dataset "Fish"	39
8.32 Studio codec 2023 sul dataset "Mannequin"	40
8.33 Studio codec 2023 sul dataset "Messerschmitt"	40
8.34 Studio codec 2023 sul dataset random "Blob"	41
8.35 Studio codec 2023 sul dataset random "Car"	41
8.36 Studio codec 2023 sul dataset random "Cobblestone"	42
8.37 Studio codec 2023 sul dataset random "Dice"	42
8.38 Studio codec 2023 sul dataset random "Fish"	42
8.39 Studio codec 2023 sul dataset random "Mannequin"	43
8.40 Studio codec 2023 sul dataset random "Messerschmitt"	43
8.41 Studio decompressione codec 2023 sul dataset "Art Gallery"	44
8.42 Studio decompressione codec 2023 sul dataset "Dragons and bunnies" . .	44
8.43 Studio decompressione codec 2023 sul dataset "OpEx"	45
8.44 Studio decompressione codec 2023 sul dataset random "Art Gallery" . .	46
8.45 Studio decompressione codec 2023 sul dataset random "Dragons and bunnies"	46
8.46 Studio decompressione codec 2023 sul dataset random "OpEx"	47
8.47 Studio decompressione codec 2023 sul dataset "Blob"	48
8.48 Studio decompressione codec 2023 sul dataset "Car"	48
8.49 Studio decompressione codec 2023 sul dataset "Cobblestone"	49
8.50 Studio decompressione codec 2023 sul dataset "Dice"	49
8.51 Studio decompressione codec 2023 sul dataset "Fish"	49
8.52 Studio decompressione codec 2023 sul dataset "Mannequin"	50
8.53 Studio decompressione codec 2023 sul dataset "Messerschmitt"	50
8.54 Studio decompressione codec 2023 sul dataset random "Blob"	51
8.55 Studio decompressione codec 2023 sul dataset random "Car"	51
8.56 Studio decompressione codec 2023 sul dataset random "Cobblestone" .	52
8.57 Studio decompressione codec 2023 sul dataset random "Dice"	52
8.58 Studio decompressione codec 2023 sul dataset random "Fish"	52
8.59 Studio decompressione codec 2023 sul dataset random "Mannequin" . .	53
8.60 Studio decompressione codec 2023 sul dataset random "Messerschmitt"	53