



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Light-Field Image Compression

PROFESSORE

Prof. Bruno Carpentieri

Università degli studi di Salerno

AUTORI

Adriano Emanuele Califano

Matricola: 0522501790

Dario De Maio

Matricola: 0522501764

Nicola Mauro

Matricola: 0522501789

Anno Accademico 2024-2025

Indice	i
Elenco delle figure	iv
Elenco delle tabelle	vi
1 Introduzione	1
1.1 Contesto applicativo	1
1.2 Motivazioni e Obiettivi	2
1.3 Organizzazione del documento	2
2 Stato dell'arte	3
2.1 Light Field Images	3
2.1.1 Definizione e caratteristiche	3
2.1.2 Tecniche di acquisizione	4
3 Lavori correlati	5
3.1 Lossy e lossless	5
3.2 Deep learning	7
4 Analisi dei codec di compressione	8
4.1 Introduzione ai sistemi lossy	9
4.2 Codec di compressione lossy analizzati	9
4.2.1 HEVC (High Efficiency Video Coding)	9

4.2.2	VP9	10
4.2.3	AV1	10
4.3	Introduzione ai sistemi lossless	10
4.4	Codec di compressione lossless analizzati	11
4.4.1	Modalità lossless in HEVC, VP9 e AV	11
4.4.2	HUFFYUV	11
4.4.3	FFV1	11
4.4.4	UTVIDEO	11
4.4.5	Dirac	12
4.4.6	MS-RLE	13
4.5	Introduzione a PCA e SVD	14
4.5.1	PCA (Principal Component Analysis)	14
4.5.2	SVD (Singular Value Decomposition)	14
4.6	Applicazione di PCA e SVD alla compressione dei dati	15
4.7	AutoEncoder	16
4.7.1	Funzionamento	16
4.7.2	Vantaggi	16
4.7.3	Applicazioni nelle Light Field Images	17
4.8	Metriche selezionate per il confronto dei risultati	17
4.8.1	PSNR (Peak Signal-to-Noise Ratio)	18
4.8.2	SSIM (Structural Similarity Index)	18
4.8.3	PSNR vs SSIM	20
4.9	Applicazione delle metriche ai risultati	20
5	Soluzione proposta	22
5.1	Tecnologie utilizzate	22
5.2	Dataset usati	23
6	Implementazione	25
6.1	Organizzazione e struttura del progetto	25
6.1.1	Cartelle principali	25
6.1.2	Moduli python principali	26
6.2	Fase di compressione	26
6.3	Fase di decompressione	28
6.4	Compressione/Decompressione tramite PCA e SVD	29

6.5	Compressione/Decompressione con autoencoder	31
6.6	Valutazione dei risultati	32
7	Risultati	33
7.1	Configurazione Hardware/Software	33
7.2	Risultati codec tradizionali	33
7.2.1	Risultati codifica lossless	34
7.2.2	Risultati codifica lossy	37
7.3	Risultati PCA/SVD	39
7.3.1	Dataset ArtGallery	40
7.3.2	Dataset Blob	41
7.3.3	Dataset Car	42
7.3.4	Dataset Cobblestone	43
7.3.5	Dataset Dice	44
7.3.6	Dataset Dragons	45
7.3.7	Dataset Fish	46
7.3.8	Dataset Mannequin	47
7.3.9	Dataset Messerschmitt	48
7.4	Risultati autoencoder	49
7.4.1	Train & Validation Loss	49
7.4.2	Dimensione dei dati & Rapporto di compressione	50
7.4.3	Tempi di compressione	52
7.4.4	Metriche SSIM & PSNR	53
8	Analisi dei Risultati e Conclusioni	54
8.1	Analisi	54
8.1.1	Analisi codec lossless	54
8.1.2	Analisi codec lossy	54
8.1.3	Analisi PCA e SVD	55
8.1.4	Confronto tra approcci lossy e riduzione di dimensionalità	55
8.1.5	Analisi autoencoder	59
8.2	Conclusioni	60
Bibliografia		62

Elenco delle figure

2.1	Esempio di immagini catturate da tre fotocamere con prospettive diverse.	4
4.1	Dirac EC schema	12
4.2	Struttura tipica di un autoencoder	17
6.1	Struttura della pipeline implementata	29
7.1	Andamento PCA e SVD su ArtGallery	40
7.2	Andamento PCA e SVD su Blob	41
7.3	Andamento PCA e SVD su Car	42
7.4	Andamento PCA e SVD su Cobblestone	43
7.5	Andamento PCA e SVD su Dice	44
7.6	Andamento PCA e SVD su Dragons	45
7.7	Andamento PCA e SVD su Fish	46
7.8	Andamento PCA e SVD su Mannequin	47
7.9	Andamento PCA e SVD su Messerschmitt	48
7.10	Train e Validation loss sul dataset di riferimento.	50
7.11	Rapporto di compressione per ogni Epoch.	51
7.12	Tempo di compressione e totale per ogni epoch.	52
7.13	SSIM e PSNR sul dataset di riferimento.	53
8.1	Frame originale	58
8.2	Frame decompresso con VP9	58
8.3	Frame decompresso con PCA	58

8.4 Frame decompresso con SVD	58
8.5 Confronto visivo su dataset ArtGallery	58
8.6 Frame originale	59
8.7 Frame decompresso con VP9	59
8.8 Frame decompresso con PCA	59
8.9 Frame decompresso con SVD	59
8.10 Confronto visivo su dataset Dice	59
8.11 Frame originale	60
8.12 Frame decompresso	60
8.13 Confronto visivo con autoencoder	60

Elenco delle tabelle

7.1	Risultati di testing con codec lossless su dataset "Room"	34
7.2	Risultati di testing con codec lossless su dataset "Blob"	34
7.3	Risultati di testing con codec lossless su dataset "Car"	35
7.4	Risultati di testing con codec lossless su dataset "Cobblestone"	35
7.5	Risultati di testing con codec lossless su dataset "Dice"	35
7.6	Risultati di testing con codec lossless su dataset "Dragons & Bunnies"	36
7.7	Risultati di testing con codec lossless su dataset "Fish"	36
7.8	Risultati di testing con codec lossless su dataset "Mannequin"	36
7.9	Risultati di testing con codec lossless su dataset "Messerschmitt"	37
7.10	Risultati di testing con i codec lossy su dataset "Room"	37
7.11	Risultati di testing con i codec lossy su dataset "Blob"	37
7.12	Risultati di testing con i codec lossy su dataset "Car"	37
7.13	Risultati di testing con i codec lossy su dataset "Cobblestone"	38
7.14	Risultati di testing con i codec lossy su dataset "Dice"	38
7.15	Risultati di testing con i codec lossy su dataset "Dragons & Bunnies"	38
7.16	Risultati di testing con i codec lossy su dataset "Fish"	38
7.17	Risultati di testing con i codec lossy su dataset "Mannequin"	38
7.18	Risultati di testing con i codec lossy su dataset "Messerschmitt"	38
7.19	Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su ArtGallery.	40
7.20	Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Blob.	41

7.21 Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Car.	42
7.22 Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Cobblestone.	43
7.23 Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Dice.	44
7.24 Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Dragons.	45
7.25 Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Fish.	46
7.26 Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Mannequin.	47
7.27 Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Messerschmitt.	48
7.28 Valori di Train Loss e Validation Loss per ciascun Epoch.	49
7.29 Tabella delle dimensioni per Epoch con il rapporto di compressione.	51
7.30 Tempi di compressione e codifica entropica per epoca.	52
7.31 Risultati delle metriche SSIM e PSNR per ogni Epoch.	53
8.1 Confronto tempi di compressione	56
8.2 Confronto rapporti di compressione	57
8.3 Confronto indici di similarità	57

CAPITOLO 1

Introduzione

1.1 Contesto applicativo

Con l'evoluzione delle tecnologie per l'acquisizione di immagini si sono evoluti anche i tool e gli strumenti per ottenere fotografie con risoluzioni sempre migliori in modo tale da catturare quanti più dettagli possibili. Tuttavia l'acquisizione di immagini "normali" non permette molti lavori di post-produzione in quanto catturano il mondo tri-dimensionale per poi portarlo su uno spazio bi-dimensionale con un focus fisso. Tutto ciò che si trova quindi di fronte oppure dietro la zona di focus appare sfocata e fuori fuoco.

Per questi motivi sono state introdotte nuove tipologie di fotocamere capaci di catturare l'intero spazio tri-dimensionale con informazioni aggiuntive riguardanti i raggi luminosi. Questa particolarità permette la modifica delle immagini in una fase successiva alla loro acquisizione, potendo modificare quindi il focus dei soggetti all'interno delle immagini a seconda dei casi di utilizzo.

Gli strumenti adatti all'acquisizione di tali immagini sono chiamati **Light-Field Cameras** o **Plenoptic Cameras**, e quindi le immagini prendono il nome di **Light Field Images**. Queste sono capaci non solo di catturare la luce proveniente da una scena, ma anche la sua direzione e intensità in ogni punto della scena stessa.

Le applicazioni e gli utilizzi di questo nuovo tipo di immagini ad alto tasso di informazioni sono quindi numerose e spaziano dalle "semplici" attività di modifica in post-produzione ad attività più complesse in ambito medico e nella realtà virtuale [1].

1.2 Motivazioni e Obiettivi

Questo studio si pone come estensione di precedenti ricerche dedicate all'implementazione, all'analisi e al confronto di algoritmi di compressione applicati alle Light Field Images. Il lavoro svolto è stato diviso in tre fasi:

- Nella prima fase si è seguito un approccio metodologico analogo a quello delle ricerche precedenti, raccogliendo dati utili per condurre confronti accurati.
- Nella seconda fase sono stati applicati metodi di analisi statistica volti alla riduzione della dimensionalità dei dati, con un focus particolare su **Principal Component Analysis** e **Singular Value Decomposition**.
- Nella terza fase si è approfondito un approccio di tipo deep learning che prevede l'utilizzo di un autoencoder al fine di riuscire a sintetizzare quanto più possibile le informazioni presenti nelle Light Field Images mediante tecniche di autoapprendimento.

Infine, i risultati ottenuti sono stati analizzati e confrontati, per valutare le prestazioni delle tecniche di compressione studiate.

1.3 Organizzazione del documento

La struttura del documento mostra in ordine i seguenti capitoli:

1. Introduzione delle Light Field Images con definizioni, tecniche di acquisizione e utilizzi.
2. Studio della metodologia dei lavori correlati al progetto.
3. Analisi dei codec di compressione tradizionali utilizzati, delle tecniche di compressione nuove e delle metriche utilizzate per la valutazione.
4. Approfondimento autoencoder.
5. Soluzione proposta, in cui vengono specificate le tecnologie utilizzate e i dataset usati per lo studio.
6. Analisi dei risultati e confronti.

CAPITOLO 2

Stato dell'arte

2.1 Light Field Images

2.1.1 Definizione e caratteristiche

Le Light Field Images rappresentano una tecnica avanzata per la cattura di informazioni luminose, poiché registrano non solo l'intensità e il colore della luce, ma anche la direzione dei raggi nello spazio tridimensionale. Questo approccio si distingue significativamente dalle immagini tradizionali, che si limitano a registrare l'intensità luminosa su un piano bidimensionale.

La registrazione del campo luminoso in 4D pone sfide significative, poiché i sensori di immagine tradizionali sono progettati esclusivamente per la rappresentazione bidimensionale. Una fotocamera convenzionale cattura un'immagine piatta che rappresenta la somma della luminosità e del colore dei raggi luminosi che raggiungono ogni singolo pixel del sensore[2].

Al contrario, una light field camera è in grado di registrare, oltre ai valori di intensità e colore, anche la direzione e l'angolo dei raggi luminosi che arrivano al sensore. Questa informazione aggiuntiva consente di ricostruire la provenienza di ogni raggio di luce prima che raggiunga la fotocamera, rendendo possibile la generazione di un modello tridimensionale della scena catturata.

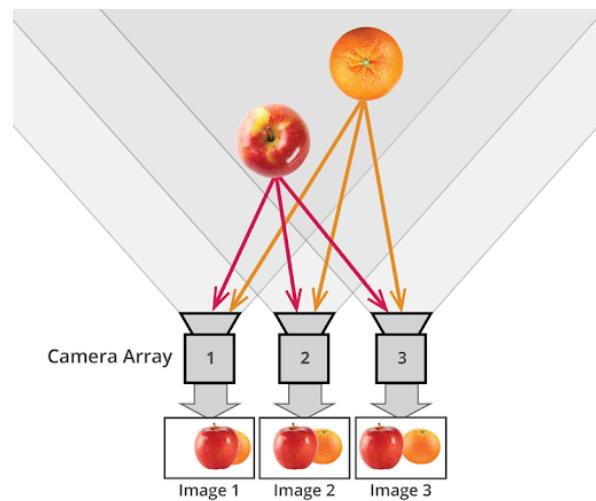


Figura 2.1: Esempio di immagini catturate da tre fotocamere con prospettive diverse.

2.1.2 Tecniche di acquisizione

La cattura dei Light Field può essere effettuata attraverso diverse tecniche:

- Tramite l'utilizzo di una singola telecamera controllata roboticamente.
- Tramite l'uso di un arco rotante composto da più telecamere che ruota intorno alla scena.
- Tramite un'array di telecamere che catturano la stessa scena ma da angolazioni e con punti di luce differenti.
- Tramite l'impiego di una singola fotocamera dotata di un array di microlenti (Plenoptic Camera).

A seconda della tecnica adottata, i dati acquisiti possono consistere in un'unica immagine contenente numerose immagini secondarie (ottenute tramite un singolo sensore associato a una matrice di microlenti) oppure in una collezione di immagini catturate da più fotocamere o esposizioni.

Un elemento comune a tutte queste tecniche è rappresentato dal fatto che le immagini secondarie o i singoli scatti differiscono leggermente l'uno dall'altro. Questa discrepanza è dovuta alla registrazione di raggi luminosi provenienti da posizioni spazialmente diverse, un aspetto che consente di ottenere una ricca rappresentazione del campo luminoso.

CAPITOLO 3

Lavori correlati

Per la compressione dei Light Field ci siamo ispirati a dei progetti precedenti condotti da colleghi presso l'Università di Salerno.

I loro risultati si basano sulla rappresentazione dei Light Field (costituiti da una serie di immagini) sotto forma di file video, sfruttando quindi codec che comunemente vengono utilizzati appunto per la compressione video. Questa scelta è motivata dal fatto che le Light Field Images sono composte da una sequenza di immagini fortemente correlate tra loro, analogamente a un flusso video.

Codec video, quali ad esempio HEVC, VP9 e AV1, sono ottimizzati per sfruttare questa correlazione, impiegando tecniche di compressione temporale e spaziale che risultano più efficienti rispetto ai codec pensati per immagini singole (o tradizionali).

3.1 Lossy e lossless

I lavori precedenti hanno preso in considerazioni sia approcci che utilizzano compressione lossless sia compressione lossy. Nel contesto di questo progetto sono state riprodotte le analisi per verificare la congruenza dei risultati con i vecchi test ma, al fine di effettuare comparazioni con le tecniche di riduzione di dimensionalità, sono stati presi in considerazione solamente i codec video precedentemente utilizzati che effettuassero una codifica di tipo lossy. Tra i vari codec utilizzati in precedenza, quelli che dispongono sia di una compressione lossless che lossy sono i seguenti:

- HEVC
- VP9
- AV1

Va notato che i diversi progetti passati hanno utilizzato dei dataset talvolta identici, talvolta completamente diversi, o in alternativa con sovrapposizioni parziali. Tuttavia, nessuno prima di noi ha esaminato l'applicazione degli specifici tre codec sopraelencati sugli stessi dataset che abbiamo preso in considerazione per il nostro studio.

Laddove parte dei calcoli fossero già stati effettuati dai nostri colleghi, si è convenuto sul replicarli ugualmente sulla nostra macchina. Questo approccio ci ha consentito di ottenere risultati uniformi, evitando possibili discrepanze e/o inconsistenze dei dati dovute a differenze nelle prestazioni hardware o software tra le macchine utilizzate.

Successivamente, gli stessi dati sono stati sottoposti a metodi statistici per la riduzione della dimensionalità, con un particolare focus su PCA (Principal Component Analysis) e SVD (Singular Value Decomposition).

Questo processo ha garantito che i risultati delle analisi statistiche fossero pienamente compatibili e consistenti con quelli derivati dalla compressione tramite HEVC, VP9 e AV1, al fine di poter svolgere un'accurata comparazione tra le prestazioni dei tre codec e quelle statistiche in termini di efficienza e qualità della compressione.

Questo ha reso possibile valutare non solo l'efficienza dei codec nel ridurre le dimensioni dei dati, ma anche l'impatto delle tecniche di compressione sulla qualità delle Light Field Images, contribuendo così ad un avanzamento nello studio di tecniche di compressione applicate a questa particolare tipologia di dati complessi.

Analogamente agli esperimenti condotti con i codec lossy e per le medesime motivazioni, anche per quanto riguarda gli approcci di compressione lossless, si è scelto, in primo luogo, di replicare gli esperimenti con una serie di codec che erano già stati utilizzati in studi precedenti. Questi codec saranno elencati e descritti nel dettaglio nelle sezioni successive del documento. Successivamente alla riproduzione di tali prove condotte coi codec già studiati, sono stati introdotti due nuovi codec lossless: DIRAC e MS-RLE. L'inclusione di questi nuovi metodi ha permesso di effettuare un confronto diretto tra le soluzioni precedentemente analizzate e approcci non ancora presi in considerazione, ampliando così il panorama delle tecniche di compressione studiate. Naturalmente, anche questi codec saranno descritti più in dettaglio nelle sezioni successive.

3.2 Deep learning

Nel campo delle Light Field Images, le tecniche basate sul deep learning, come l'uso di autoencoder, offrono un approccio innovativo e adattivo rispetto ai metodi tradizionali. Il lavoro di base analizzato preliminarmente, svolto negli anni passati, si concentrava sull'utilizzo degli autoencoder per comprimere ologrammi, evidenziandone le potenzialità e le limitazioni. Questa breve sezione illustra le differenze tra il lavoro originale e le modifiche introdotte nell'implementazione di questo progetto.

Nel lavoro analizzato, gli autoencoder sono stati utilizzati per comprimere ologrammi mediante un'architettura standard composta da un encoder e un decoder. L'encoder estrae una rappresentazione latente compatta del dato, mentre il decoder ricostruisce il dato originale a partire dalla rappresentazione latente. Gli aspetti principali includono:

- utilizzo di una rete convoluzionale per l'estrazione delle caratteristiche;
- funzione di perdita basata sull'errore quadratico medio (MSE);
- addestramento su dataset contenenti immagine rappresentate da matrici complesse reinterpretate come matrici di floating-point.

Nonostante i risultati promettenti, il lavoro originale evidenzia alcune limitazioni, come la difficoltà di generalizzare a causa di dataset limitati e l'onerosità computazionale.

Nel file **trainer.ipynb**, sono state apportate modifiche significative per migliorare la compressione di Light Field Images con autoencoder, tra cui:

- è stata introdotta un'architettura personalizzata tramite l'aggiunta di strati convoluzionali con kernel di dimensione diversa per migliorare la cattura delle caratteristiche spaziali e l'aggiunta di un approccio di normalizzazione batch per stabilizzare l'addestramento;
- è stata migliorata la codifica latente andando a ottimizzarne la dimensione per bilanciare compressione e fedeltà e aggiungendo dropout layer per ridurre il rischio di overfitting;
- oltre alla semplice reinterpretazione delle matrici complesse, sono state sperimentate tecniche di preprocessing come la normalizzazione dei dati;

Le differenze tra l'approccio originale e quello implementato sottolineano il potenziale degli autoencoder personalizzati per applicazioni specifiche come la compressione degli ologrammi.

CAPITOLO 4

Analisi dei codec di compressione

Questo capitolo si pone come obiettivo quello di andare a descrivere e ad analizzare gli algoritmi di compressione applicabili nell'ambito delle Light Field Images che si è scelto di utilizzare per il progetto, sottolineando il ruolo fondamentale che la compressione ricopre, quando si ha a che fare con un numero così grande di immagini.

Come discusso in precedenza, l'obiettivo principale di questo progetto è approfondire gli studi esistenti sulle tecniche di compressione applicate alle Light Field Images, con particolare attenzione al confronto tra approcci consolidati e strategie innovative.

Il percorso di ricerca si suddivide in due fasi principali. Nella prima fase, viene effettuata un'analisi critica delle tecniche di compressione già validate da studi precedenti. Questo lavoro si concentra sull'esame dettagliato delle metodologie adottate, dei risultati raggiunti e delle eventuali limitazioni emerse, con l'obiettivo di individuare spunti per un confronto approfondito con soluzioni più recenti.

La seconda fase della ricerca è invece dedicata all'implementazione e all'adattamento di due tecniche ampiamente utilizzate nella data science: la Principal Component Analysis (PCA) e la Singular Value Decomposition (SVD). Questi metodi, riconosciuti per la loro capacità di ridurre la dimensionalità e comprimere dati complessi, vengono applicati e studiati nel contesto specifico delle Light Field Images.

L'obiettivo è sfruttare le caratteristiche intrinseche di questi dataset, come la loro struttura 4D e le correlazioni tra dimensioni spaziali e angolari, per migliorare l'efficienza della compressione e ridurre al minimo la perdita di qualità.

L'intero lavoro si propone di confrontare le prestazioni di PCA e SVD rispetto alle tecniche tradizionali, valutandone l'efficacia in termini di rapporto compressione-qualità e complessità temporale.

4.1 Introduzione ai sistemi lossy

I sistemi di compressione lossy riducono significativamente le dimensioni dei dati sacrificando una parte dell'informazione originale. Questa perdita, spesso non percepibile dall'occhio umano, rende le tecniche lossy particolarmente utili in applicazioni che richiedono un bilanciamento tra qualità e dimensione dei file, come nel caso di immagini, video e audio.

Nel contesto della compressione delle Light Field Images, i sistemi lossy sono vantaggiosi poiché consentono di trattare i grandi volumi di dati generati da questa tecnologia in modo efficiente, garantendo una qualità accettabile per molte applicazioni pratiche. Tra i codec video più rilevanti in questo ambito troviamo HEVC, VP9 e AV1, ottimizzati per sfruttare le correlazioni spaziali e temporali nei dati.

4.2 Codec di compressione lossy analizzati

4.2.1 HEVC (High Efficiency Video Coding)

HEVC, noto anche come H.265, è uno standard di compressione video approvato il 25 gennaio 2013 e sviluppato dal Moving Pictures Experts Group (MPEG) e dal Video Coding Experts Group (VCEG). Esso è stato implementato per offrire una maggiore efficienza rispetto al suo predecessore H.264/AVC. Le principali caratteristiche di HEVC includono:

- blocchi di codifica più grandi per gestire meglio le immagini in alta definizione;
- compressione temporale ottimizzata attraverso l'analisi dei frame precedenti e successivi;
- supporto per risoluzioni elevate, fino a 8K.

HEVC è particolarmente efficace nel ridurre le dimensioni dei dati mantenendo una qualità visiva elevata, rendendolo una scelta diffusa per applicazioni come lo streaming e l'archiviazione video.

4.2.2 VP9

VP9 è un codec open-source sviluppato da Google come alternativa a HEVC. È progettato per fornire una compressione efficiente senza richiedere royalty, il che lo rende particolarmente attraente per piattaforme come YouTube. Le sue caratteristiche principali includono:

- algoritmi ottimizzati per la compressione di video ad alta risoluzione;
- buona efficienza di codifica anche a bitrate più bassi rispetto a molti codec tradizionali;
- prestazioni eccellenti in scenari di streaming, con particolare attenzione alla qualità in relazione alla banda disponibile.

4.2.3 AV1

AV1, sviluppato dall'Alliance for Open Media, è uno dei codec più avanzati e promettenti attualmente disponibili. Pensato per sostituire VP9, offre miglioramenti significativi in termini di compressione ed efficienza. Tra le sue caratteristiche principali:

- un'efficienza di compressione superiore a HEVC e VP9, soprattutto a bitrate bassi;
- progettato per supportare applicazioni future, come video a 360° e realtà virtuale;
- compatibilità con un ecosistema open-source, senza royalty.

4.3 Introduzione ai sistemi lossless

I sistemi di compressione lossless permettono di ridurre le dimensioni dei dati senza alcuna perdita di informazione, garantendo che i dati originali possano essere completamente ricostruiti dal file compresso. Questa caratteristica li rende ideali per applicazioni in cui è essenziale preservare ogni dettaglio, come archiviazione di documenti, analisi scientifica, o trattamento di immagini mediche.

Nel contesto delle Light Field Images, i metodi lossless sono utilizzati in applicazioni che richiedono la massima fedeltà ai dati originali, ad esempio per scopi di analisi o archiviazione di dati di alta qualità. Tuttavia, i sistemi lossless tendono ad offrire un tasso di compressione più basso rispetto ai metodi lossy.

4.4 Codec di compressione lossless analizzati

4.4.1 Modalità lossless in HEVC, VP9 e AV

Sebbene HEVC, VP9 e AV1 siano ottimizzati principalmente per la compressione lossy, tutti e tre i codec supportano anche la modalità di compressione lossless. Queste modalità consentono di preservare integralmente i dati originali, rendendoli utili in scenari che richiedono la qualità assoluta, come l'archiviazione di contenuti o applicazioni professionali. La compressione lossless in questi codec sfrutta tecniche come la codifica entropica avanzata e l'eliminazione delle ridondanze, garantendo una rappresentazione fedele dei dati senza compromessi di qualità.

4.4.2 HUFFYUV

HuffYUV è un codec di compressione video lossless progettato per offrire una compressione rapida con un basso utilizzo della CPU. Il suo funzionamento si basa su una combinazione di predizione differenziale e codifica Huffman per ridurre la ridondanza nei dati video. Sebbene non sia più sviluppato attivamente, HuffYUV rimane una scelta popolare per la cattura e l'editing di video non compressi grazie alla sua semplicità e velocità.

4.4.3 FFV1

FFV1 è un codec lossless sviluppato nell'ambito del progetto FFmpeg. È noto per la sua elevata efficienza di compressione e l'ottima capacità di preservare i dettagli, rendendolo ideale per l'archiviazione a lungo termine di contenuti video. Utilizza tecniche avanzate di predizione e codifica entropica come il range coding o l'arithmetic coding, per ridurre al minimo la dimensione dei file senza sacrificarne la qualità.

4.4.4 UVIDEO

UTVIDEO è un codec lossless multipiattaforma che offre un'eccellente combinazione di velocità di compressione e qualità. È particolarmente popolare in applicazioni di editing video grazie alla sua capacità di comprimere i dati video senza perdite, mantenendo una decodifica rapida e un supporto esteso per vari formati di colore, come YUV e RGB.

4.4.5 Dirac

Il codec **Dirac**[3] è un codec basato su trasformate wavelet con compensazione del moto. La sua architettura si compone di quattro moduli principali:

Trasformata, Scaling e Quantizzazione

La trasformata e lo scaling prevedono l'applicazione di una trasformata wavelet ai dati del frame e la successiva scalatura dei coefficienti per consentire la quantizzazione. Per la quantizzazione, si utilizza un algoritmo di ottimizzazione rate-distortion, che rimuove l'informazione riducendo al minimo la distorsione visiva.

Codifica Entropica (Entropy Coding, EC)

La codifica entropica (EC) si compone di tre fasi principali, illustrate in Figura 4.1:

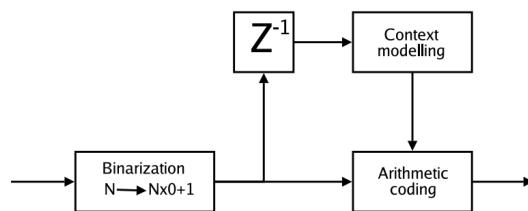


Figura 4.1: Dirac EC schema

1. **Binarizzazione:** converte i dati in una rappresentazione binaria che facilita l'analisi statistica e l'adattamento al codificatore aritmetico.
2. **Modellazione del contesto:** sfrutta le correlazioni tra coefficienti adiacenti e i loro genitori nella decomposizione wavelet per predire con maggiore precisione la distribuzione dei simboli.
3. **Codifica aritmetica (AC):** esegue la compressione senza perdita dei dati binarizzati, adattandosi dinamicamente alla distribuzione statistica dei simboli.

Lo stesso schema di codifica entropica viene applicato sia ai coefficienti della trasformata quantizzati sia ai dati dei vettori di movimento.

Stima del Moto (Motion Estimation, ME)

La stima del moto sfrutta la ridondanza temporale nei flussi video cercando somiglianze tra frame adiacenti. Dirac implementa una stima del moto gerarchica e definisce tre tipi di frame:

- **Frame Intra (I)**: codificati senza riferimento ad altri frame.
- **Frame di livello 1 (L1)**: codificati con riferimento a frame precedenti e usati come riferimento per altri frame.
- **Frame di livello 2 (L2)**: codificati con riferimento a frame precedenti ma non usati come riferimento per altri frame.

Dirac utilizza la **Compensazione del Moto a Blocchi Sovrapposti** (Overlapped Block-based Motion Compensation, OBMC) per ridurre gli artefatti ai bordi dei blocchi e migliorare la qualità della compressione. Inoltre, il codec fornisce una **compensazione del moto sub-pixel**, con vettori di movimento precisi fino a 1/8 di pixel.

Compensazione del Moto (Motion Compensation, MC)

La compensazione del moto è l'operazione inversa della stima del moto. Utilizza i vettori di movimento per predire il frame corrente in modo da minimizzare il costo di codifica dei dati residui, ossia la differenza tra il frame originale e quello stimato. Esiste un compromesso tra accuratezza della predizione e bit rate necessario per la trasmissione dei vettori di movimento.

4.4.6 MS-RLE

Il codec **MS-RLE (Microsoft Run-Length Encoding)** è un algoritmo di compressione che utilizza una tecnica di **Run-Length Encoding (RLE)**[4][5] per ridurre le dimensioni dei dati, rappresentando sequenze ripetitive di valori con un solo valore e la sua lunghezza. MS-RLE è principalmente utilizzato per la compressione di immagini, in particolare nei formati BMP (Bitmap) e PCX, ma può essere applicato anche a flussi di dati generici.

Una delle sue caratteristiche principali è la sua complessità ridotta; infatti è relativamente semplice e veloce, adatto per applicazioni che richiedono una compressione/decompressione rapida e che possono tollerare una riduzione limitata del fattore di compressione rispetto ad algoritmi più complessi.

Metodo di compressione

Il funzionamento dell'algoritmo si compone dei seguenti passi:

- Lettura dei dati di input.
- Conteggio dei caratteri consecutivi, chiamati "run". Un run è rappresentato dal valore del pixel e dal numero di volte in cui si ripete consecutivamente.
- I run individuati vengono salvati come una coppia di valori (v, l) , dove v è il valore del pixel ripetuto e l è il numero di ripetizione nella sequenza. Ad esempio se in una sequenza di dati si trova il valore "255" ripetuto cinque volte, questa sequenza sarà rappresentata come $(255, 5)$.

4.5 Introduzione a PCA e SVD

4.5.1 PCA (Principal Component Analysis)

PCA è una tecnica di analisi statistica utilizzata per ridurre la dimensionalità di un dataset, mantenendo il maggior numero possibile di informazioni significative. Si basa sulla trasformazione delle variabili originali in nuove variabili, chiamate componenti principali, che sono ortogonali tra loro e ordinate per varianza decrescente.

Nel contesto della compressione dei dati, PCA può essere applicata per:

- identificare i pattern principali nei dati compressi;
- ridurre la dimensionalità dei dati, rendendo più efficienti i processi di archiviazione e trasmissione;
- analizzare l'impatto della compressione sulla struttura delle Light Field Images.

4.5.2 SVD (Singular Value Decomposition)

SVD è un'altra tecnica fondamentale per la riduzione della dimensionalità. Essa decomponete una matrice in tre matrici separate:

$$A = U \cdot \Sigma \cdot V^T$$

dove U e V^T rappresentano le matrici ortogonali e Σ contiene i valori singolari che rappresentano l'importanza delle componenti.

In ambito di compressione:

- SVD può essere utilizzato per ridurre il rumore e mantenere solo le componenti più rilevanti di un'immagine o di un video;
- consente una rappresentazione compatta e ottimizzata dei dati, particolarmente utile per analizzare gli effetti della compressione lossy.

4.6 Applicazione di PCA e SVD alla compressione dei dati

Le tecniche di PCA e SVD trovano applicazione diretta nell'analisi delle Light Field Images compresse con codec video. Ecco come possono essere utilizzate:

- valutazione dell'impatto della compressione: PCA e SVD possono evidenziare come la compressione lossy influisce sulla struttura interna dei dati, permettendo di identificare eventuali perdite critiche di informazioni;
- riduzione della dimensionalità: questi metodi consentono di estrarre informazioni essenziali dai dati compressi, riducendo il rumore o le ridondanze introdotte dalla compressione;
- confronto tra codec: analizzando i dataset compressi con HEVC, VP9 e AV1, PCA e SVD possono aiutare a quantificare e visualizzare le differenze di efficienza e qualità tra i codec, fornendo una base statistica solida per il confronto.

4.7 AutoEncoder

Gli autoencoder sono un tipo specifico di architettura di rete neurale addestrata tramite apprendimento non supervisionato per ricostruire i dati di input.

Un autoencoder è progettato per comprimere (codificare) in modo efficiente i dati di input fino a una rappresentazione essenziale, e quindi ricostruire (decodificare) l'input originale partendo da questa rappresentazione compressa [6].

4.7.1 Funzionamento

Gli autoencoder scoprono le variabili latenti passando i dati di input attraverso un “collo di bottiglia” prima che raggiungano il decoder. Questo processo forza l'encoder a imparare a estrarre e trasmettere solo le informazioni più rilevanti per ricostruire accuratamente l'input originale.

L' **encoder** è composto da strati che codificano una rappresentazione compressa dei dati di input riducendo la loro dimensionalità. In un tipico autoencoder, gli strati nascosti contengono un numero progressivamente minore di nodi rispetto allo strato di input. Mentre i dati attraversano gli strati dell'encoder, vengono compressi in meno dimensioni tramite questo processo di riduzione.

Il **collo di bottiglia** (o “codice”) rappresenta la versione più compressa dell'input: è sia lo strato di output dell'encoder sia lo strato di input del decoder. Un obiettivo cruciale nella progettazione e nell'addestramento di un autoencoder è la scoperta del numero minimo di caratteristiche essenziali (o dimensioni) necessarie per ricostruire efficacemente i dati di input. La rappresentazione nello spazio latente, ossia il codice, generata in questo livello viene quindi inserita nel decoder.

Il **decoder** è composto da strati nascosti con un numero progressivamente maggiore di nodi che decomprimono (o decodificano) la rappresentazione compressa, ricostruendo infine i dati nella loro forma originale. L'output ricostruito viene quindi confrontato con i dati reali, che corrispondono all'input originale, per valutare l'efficacia dell'autoencoder. La differenza tra l'output e l'input è chiamata **errore di ricostruzione**.

4.7.2 Vantaggi

Uno dei principali vantaggi degli autoencoder rispetto ad altre tecniche di riduzione della dimensionalità, come la **PCA**, è la capacità di catturare complesse correlazioni non lineari.

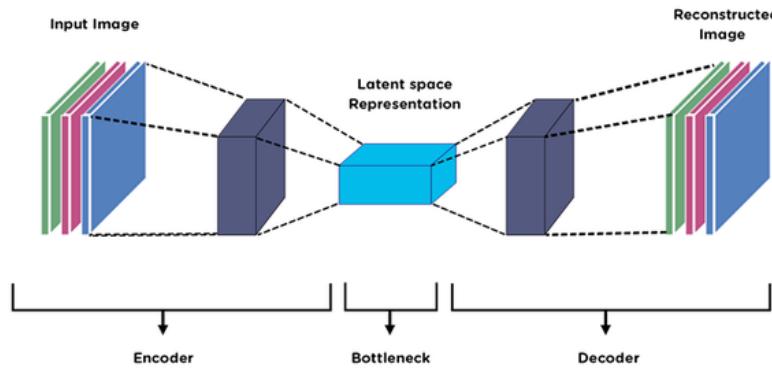


Figura 4.2: Struttura tipica di un autoencoder

Per questo motivo, le funzioni di attivazione utilizzate negli autoencoder sono tipicamente funzioni non lineari come la funzione sigmoidea.

4.7.3 Applicazioni nelle Light Field Images

Gli autoencoder possono essere estremamente utili per comprimere e decomprimere le immagini light field grazie alla loro capacità di catturare e rappresentare le complesse strutture spaziali e angolari di questi dati. Le light field images contengono una grande quantità di informazioni, che includono non solo l'intensità della luce ma anche la direzione dei raggi. Utilizzando un autoencoder, è possibile ridurre significativamente la dimensionalità di queste immagini, mantenendo le caratteristiche essenziali necessarie per una ricostruzione accurata. Questo processo consente una memorizzazione più efficiente e una trasmissione più rapida dei dati, facilitando l'uso delle light field images in applicazioni pratiche.

4.8 Metriche selezionate per il confronto dei risultati

Per valutare l'efficacia dei diversi codec lossy (HEVC, VP9 e AV1) nella compressione delle Light Field Images, sono state utilizzate due metriche principali: SSIM (Structural Similarity Index) e PSNR (Peak Signal-to-Noise Ratio). Queste metriche forniscono una misura quantitativa del degrado della qualità delle immagini o dei video compressi rispetto ai dati originali.

4.8.1 PSNR (Peak Signal-to-Noise Ratio)

PSNR è una misura classica utilizzata per quantificare il livello di distorsione introdotto dalla compressione. Si basa sul calcolo dell'errore quadratico medio (MSE, Mean Squared Error) tra i valori dei pixel dell'immagine originale e quelli dell'immagine compressa. La formula per il calcolo è:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

dove:

- MAX rappresenta il valore massimo possibile per i pixel (ad esempio 255 per immagini a 8 bit);
- MSE è il sopracitato errore quadratico medio calcolato sui pixel mediante la seguente formula:

$$\text{MSE} = \frac{1}{M \cdot N} \sum_{i=1}^M \sum_{j=1}^N [I(i, j) - K(i, j)]^2$$

con $I(i, j)$ e $K(i, j)$ che indicano rispettivamente i valori dei pixel dell'immagine originale e compressa.

Un valore PSNR più alto indica una qualità migliore della compressione, poiché implica che l'errore tra le immagini è minore. Tuttavia, PSNR è sensibile solo alle differenze globali e non tiene conto della percezione umana.

4.8.2 SSIM (Structural Similarity Index)

SSIM è una metrica più avanzata rispetto a PSNR, progettata per valutare la degradazione delle immagini comprimendo la qualità percepita delle immagini in termini di informazioni strutturali. Le informazioni strutturali rappresentano i pattern di correlazione tra pixel vicini che risultano essenziali per la percezione umana.

A differenza di altre metriche come il PSNR, che si basano su differenze assolute tra i valori dei pixel, infatti, l'SSIM si concentra sui cambiamenti percepiti nella struttura delle immagini, tenendo conto di fenomeni visivi umani come il mascheramento della luminanza e il mascheramento del contrasto.

L'SSIM misura la qualità comparando due immagini, tipicamente l'originale e quella compressa, analizzando nello specifico tre aspetti principali che contribuiscono alla percezione visiva: luminanza, contrasto e struttura. Ecco come vengono calcolati le tre componenti:

- luminanza ($l(x, y)$) : misura la somiglianza nelle intensità medie dei pixel delle due immagini (quella originale e quella compressa). Per due segnali x e y , rappresentati da N pixel:

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i \quad \text{e} \quad \mu_y = \frac{1}{N} \sum_{i=1}^N y_i$$

dove μ_x e μ_y rappresentano le medie dei pixel per le immagini x e y ;

- contrasto ($c(x, y)$): valuta le differenze di intensità tra i pixel nelle due immagini e viene calcolato utilizzando la varianza o la deviazione standard dei pixel:

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

in maniera del tutto analoga σ_y^2 rappresenta la varianza per i pixel dell'immagine y ;

- struttura ($s(x, y)$): misura la somiglianza delle strutture spaziali tra le immagini analizzando la correlazione tra i pixel delle due immagini, valutando la loro dipendenza spaziale. La struttura normalizzata può essere espressa come:

$$x' = \frac{x - \mu_x}{\sigma_x}, \quad y' = \frac{y - \mu_y}{\sigma_y}$$

La correlazione tra x e y è rappresentata dalla covarianza σ_{xy} :

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y)$$

Infine, SSIM combina queste tre componenti attraverso una funzione aggregata per ottenere una misura complessiva di similarità, ottenendo la seguente formula generale:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

dove C_1 e C_2 sono costanti che evitano instabilità numeriche in caso di denominatori piccoli. Tali valori non influiscono significativamente sui risultati finali di SSIM, a meno che non ci siano immagini con caratteristiche particolari. Vengono solitamente scelti in base alla dinamica dei pixel delle immagini. Ad esempio, nelle immagini a 8 bit, i valori possibili per ogni pixel variano da 0 a 255. La scelta tipica delle costanti è la seguente:

- $C_1 = (K_1 L)^2$

- $C_2 = (K_2 L)^2$

dove:

- L è il valore massimo che un pixel può assumere (ad esempio, 255 per immagini a 8 bit);
- K_1 e K_2 sono costanti che solitamente assumono i seguenti valori tipici: $K_1 = 0,01$ e $K_2 = 0,03$.

Questi valori sono una convenzione standard e sono stati determinati empiricamente per ottenere una buona stabilità e performance nella maggior parte delle immagini, ma possono essere adattati a seconda delle necessità.

SSIM - Interpretazione

L'SSIM restituisce un valore compreso tra 0 e 1, dove 1 indica una somiglianza perfetta tra le immagini. Grazie alla sua capacità di catturare le caratteristiche visive, questa metrica è ampiamente adottata per valutare la qualità percepita di immagini e video compressi, risultando più affidabile di metriche puramente matematiche come PSNR.

4.8.3 PSNR vs SSIM

Mentre PSNR è semplice da calcolare ed è ampiamente utilizzato, SSIM è considerato più affidabile per catturare la qualità percepita, poiché tiene conto delle caratteristiche visive umane.

In questo studio, entrambe le metriche sono state impiegate per offrire una valutazione completa:

- PSNR per confronti tecnici e misurazioni standardizzate;
- SSIM per analisi più vicine alla percezione reale della qualità visiva.

4.9 Applicazione delle metriche ai risultati

I valori di PSNR e SSIM sono stati calcolati per ciascun dataset compresso in tutte le varie fasi della sperimentazione al fine di effettuare dei confronti tra i vari approcci. I risultati ottenuti hanno permesso di:

- quantificare la qualità delle Light Field Images dopo la compressione;
- confrontare l'efficienza dei codec, evidenziando le differenze di prestazioni in termini di qualità percepita e fedeltà rispetto ai dati originali;

- correlare i valori di PSNR e SSIM con i risultati delle analisi statistiche (PCA e SVD) per valutare l'impatto complessivo delle tecniche di compressione.

CAPITOLO 5

Soluzione proposta

5.1 Tecnologie utilizzate

Per lo sviluppo di questo studio è stato utilizzato **Python** come linguaggio di programmazione, grazie alla sua versatilità e all'ampia gamma di librerie disponibili. Python si è dimostrato particolarmente efficace nell'analisi e manipolazione dei dati, permettendo l'implementazione di soluzioni in modo semplice ed efficiente. L'ampio ecosistema di librerie disponibili ha inoltre contribuito a ridurre significativamente il rischio di errori durante le operazioni di elaborazione, migliorando la qualità e l'affidabilità complessiva del lavoro.

Tra le tecnologie adottate, un ruolo fondamentale è stato svolto da **FFmpeg**, una suite software estremamente versatile per la registrazione, conversione e riproduzione di file multimediali. In particolare, il tool principale `ffmpeg`, accessibile da riga di comando, è stato utilizzato per convertire file video e audio tra diversi formati. Questo ha consentito di riprodurre con successo gli esperimenti precedentemente condotti dai colleghi, garantendo un'elevata interoperabilità tra i dati multimediali utilizzati.

Un contributo significativo è stato fornito anche dai seguenti due moduli: **tensorflow.keras** e **sklearn.decomposition**.

- **tensorflow.keras**: utilizzato per la costruzione, addestramento ed utilizzo della rete neurale convoluzionale.
- **sklearn.decomposition**: utilizzato per l'applicazione di modelli di decomposizio-

ne come *Singular Value Decomposition (SVD)* e *Principal Component Analysis (PCA)*, fondamentali per l'analisi e la riduzione dimensionale dei dati.

5.2 Dataset usati

Per la realizzazione di questo lavoro sono stati utilizzati i seguenti dataset:

1. "Dragons and Bunnies" [7]: realizzato dal MIT Media Lab, consiste in 26 immagini in formato .png rappresentanti scene renderizzate. Esse mostrano modelli 3D di conigli e draghi realizzati dallo Stanford University Computer Graphics Laboratory. Tutti i Light Fields hanno una risoluzione di 840 x 593 pixel.
2. "Messerschmitt" [8]: realizzato dal MIT Media Lab, consiste in 25 immagini in formato .png rappresentanti scene renderizzate. Esse mostrano modelli 3D di una vettura del marchio Messerschmitt realizzati dallo Stanford University Computer Graphics Laboratory. Tutti i Light Fields hanno una risoluzione di 840 x 593 pixel.
3. "Dice" [9]: realizzato dal MIT Media Lab, consiste in 25 immagini in formato .png rappresentanti scene renderizzate. Esse mostrano modelli 3D di dadi da gioco realizzati dallo Stanford University Computer Graphics Laboratory. Tutti i Light Fields hanno una risoluzione di 840 x 593 pixel.
4. "Fish" [10]: realizzato dal MIT Media Lab, consiste in 25 immagini in formato .png rappresentanti scene renderizzate. Esse mostrano modelli 3D pesci realizzati dallo Stanford University Computer Graphics Laboratory. Tutti i Light Fields hanno una risoluzione di 840 x 593 pixel.
5. "Car" [11]: realizzato dal Max Planck Institut Informatik, consiste in 101 immagini in formato .png, rappresentanti scene renderizzate. Esse presentano scene 3D di una autovettura. Tutti i Light Fields hanno una risoluzione spaziale di 960 x 720 pixel.
6. "Room" [12]: realizzato dal Max Planck Institut Informatik, consiste in 101 immagini in formato .png, rappresentanti scene renderizzate. Tutti i Light Fields hanno una risoluzione spaziale di 960 x 720 pixel.
7. "Cobblestone" [13]: realizzato dal Max Planck Institut Informatik, consiste in 101 immagini in formato .png, rappresentanti scene renderizzate. Esse presentano scene 3D di un sentiero di pietra. Tutti i Light Fields hanno una risoluzione spaziale di 960 x 720 pixel.

8. "Mannequin" [14]: realizzato dal Max Planck Institut Informatik, consiste in 101 immagini in formato .png, rappresentanti scene renderizzate. Esse presentano scene 3D di un manichino in una stanza. Tutti i Light Fields hanno una risoluzione spaziale di 960 x 720 pixel.
9. "Blob" [15]: realizzato dal Max Planck Institut Informatik, consiste in 101 immagini in formato .png, rappresentanti scene renderizzate. Esse presentano scene 3D di un "blob". Tutti i Light Fields hanno una risoluzione spaziale di 960 x 720 pixel.

CAPITOLO 6

Implementazione

Per l'implementazione del progetto è stata fatta la scelta di mantenere una struttura simile ai progetti precedentemente svolti, almeno nello studio e nella fase di esecuzione dei codec precedentemente testati. Sono stati utilizzati quindi gli stessi parametri utilizzati precedentemente dai nostri colleghi per garantire continuità e congruenza con quello che è stato fatto in precedenza.

6.1 Organizzazione e struttura del progetto

6.1.1 Cartelle principali

La struttura del progetto, disponibile al seguente link¹, è organizzata come segue:

- **./datasets**, contenente i dataset di input divisi in sottocartelle specifiche per ogni dataset.
- **./src**, contenente i moduli python creati per effettuare la compressione e decompressione delle immagini tramite i codec video e le tecniche di riduzione della dimensionalità e autoencoder.
- **./**, contenente i file principali quali il file di requirements per l'installazione delle dipendenze python e file docker per la creazione del container di sviluppo.

¹<https://github.com/adriano22jr/light-field-compression.git>

6.1.2 Moduli python principali

I moduli python sono stati suddivisi per funzionalità implementate, in particolare:

- **video_compression.py**, contenente le chiamate per invocare le compressioni con i codec utilizzati.
- **video_decompression.py**, contenente le chiamate per decomprimere i video e riottenere i frame dai dataset compressi.
- **compressors.py**, modulo utilizzato per la definizione delle funzioni di compressione specifiche per ciascun codec video.
- **results_evaluation.py**, contenente il codice per quantificare la bontà della compressione tramite le metriche precedentemente specificate.

6.2 Fase di compressione

La fase di compressione con i codec video tradizionali è stata effettuata per mezzo dei due moduli python discussi in precedenza **video_compression.py** e **compressors.py**.

La compressione viene effettuata utilizzando il seguente comando

```
1     python video_compression.py [ALGORITHM]
2                           [INPUT PATH]
3                           [OUTPUT PATH]
```

dove:

- **ALGORITHM** identifica l'algoritmo da utilizzare per la compressione.
- **INPUT PATH** identifica il percorso relativo al primo frame del dataset da voler comprimere (es. /datasets/ArtGallery2/Frame_%3d.png).
- **OUTPUT PATH** identifica il percorso dove il file di output verrà salvato (es. /results/output.mp4).

Ogni funzione di compressione specifica per ogni codec effettua una chiamata al tool `ffmpeg` tramite il comando `subprocess.run()` passando in input i parametri per l'invocazione del codec specifico, in particolare viene scelto il **framerate** del video in output, il **codec** da voler utilizzare, il **percorso di input**, un **constant rate factor** che regola la compressione, e

il percorso di output.

Le invocazioni per i codec in modalità lossy di HEVC, AV1 e VP9 sono le seguenti:

```
1 # codec HEVC
2 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
3                 "-crf", "3", "-c:v", "libx265", output_file])
4
5 # codec AV1
6 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
7                 "-crf", "3", "-c:v", "libaom-av1", output_file])
8
9 # codec VP9
10 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
11                  "-crf", "3", "-c:v", "libvpx-vp9", output_file])
```

Le invocazioni per i codec in modalità lossless sono le seguenti:

```
1 # codec HEVC
2 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
3                 "-c:v", "libx265", "-x265-params", "lossless=1", output_file])
4
5 # codec AV1
6 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
7                 "-c:v", "libaom-av1", "-aom-params", "lossless=1", output_file])
8
9 # codec VP9
10 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
11                  "-c:v", "libvpx-vp9", "-lossless", "1", output_file])
12
13 # codec HuffYUV
14 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
15                  "-c:v", "huffyuv", output_file])
16
17 # codec UTVideo
18 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
19                  "-c:v", "utvideo", output_file])
20
21 # codec FFV1
22 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
23                  "-level", "3", "-coder", "2", "-c:v", "ffv1", output_file])
24
25 # codec DIRAC
26 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
27                  "-c:v", "dirac", output_file])
28
29 # codec MS-RLE
30 subprocess.run(["../../../bin/ffmpeg", "-framerate", "120", "-i", input_file,
31                  "-c:v", "msrle", output_file])
```

Gli output della fase di compressione sono dei file video nei seguenti formati:

- **.mp4**, per il codec HEVC.
- **.mkv**, per i codec AV1 e Dirac.
- **.webm**, per il codec VP9.
- **.avi**, per i codec lossless HuffYUV, UTVideo, FFV1 e MS-RLE.

L'intera fase di compressione viene monitorata effettuando due chiamate della libreria `time.time()` prima e dopo l'invocazione di `ffmpeg` tramite subprocess per quantificare il tempo necessario per la compressione.

6.3 Fase di decompressione

La fase di decompressione viene effettuata invocando il seguente comando

```
1 python video_decompression.py [INPUT PATH]
2                               [OUTPUT PATH]
```

dove:

- **INPUT PATH**, identifica il percorso relativo al file di output della fase di compressione.
- **OUTPUT PATH**, identifica il percorso relativo alla cartella dove salvare i frame da estrarre.

In questa fase viene preso in input il file video specifico per ogni dataset ottenuto nella fase di compressione e viene suddiviso nei suoi frame tramite l'utilizzo della libreria `pyav`. I frame vengono poi salvati nel percorso specificato all'invocazione del comando di avvio.

6.4 Compressione/Decompressione tramite PCA e SVD

In questa sezione vengono descritti i metodi implementati per la compressione e la decompressione delle immagini Light Field utilizzando tecniche di riduzione della dimensionalità, quali **Principal Component Analysis** (PCA) e **Singular Value Decomposition** (SVD).

L'implementazione e lo scenario di utilizzo sono equivalenti per entrambe le metodologie, poiché differiscono esclusivamente nel modello applicato. Lo scenario prevede l'uso di due modelli pre-addestrati sulla fonte di dati che si desidera condividere, modelli che sono comuni sia all'encoder che al decoder. Poiché l'obiettivo è facilitare la comunicazione online, è fondamentale che entrambe le parti della comunicazione dispongano dello stesso modello, utilizzato per la fase di codifica dei dati. La pipeline di addestramento e scambio dei dati è strutturata come segue:

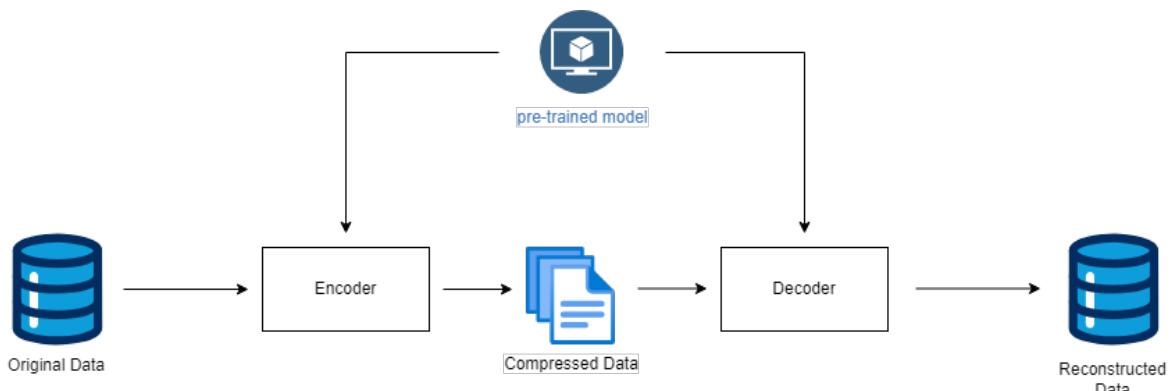


Figura 6.1: Struttura della pipeline implementata

Per ottimizzare ulteriormente lo scambio di dati tra le due parti, si è deciso di applicare un ulteriore livello di compressione lossless ai dati da trasmettere. Questa strategia consente di ridurre il volume delle informazioni senza perdita di qualità, evitando di lavorare direttamente su dati grezzi e migliorando così l'efficienza della trasmissione.

Per ottimizzare ulteriormente lo scambio di dati tra le due parti, è stato deciso di applicare un ulteriore livello di compressione lossless, riducendo il volume delle informazioni trasmesse senza perdita di qualità. Questa strategia consente di migliorare l'efficienza della trasmissione evitando di lavorare direttamente su dati grezzi.

A tal fine, è stato scelto **Zstandard (zstd)** [16], un algoritmo di compressione senza perdita noto per il suo elevato rapporto di compressione e l'alta velocità sia in fase di compressione che di decompressione. La sua efficienza è garantita dall'uso di tecniche avanzate che combinano flessibilità e prestazioni ottimali.

La compressione in Zstandard si basa sulla suddivisione dei dati in frame, unità indipendenti che contengono tutte le informazioni necessarie per la decompressione, inclusi parametri di compressione e, opzionalmente, un checksum per la verifica dell'integrità. Ogni frame è ulteriormente suddiviso in blocchi, che possono essere compressi o lasciati inalterati in base alla loro comprimibilità, garantendo un'elaborazione più efficiente.

Per ottimizzare la compressione, Zstandard impiega due algoritmi di codifica entropica:

- **Finite State Entropy (FSE):** una tecnica di compressione entropica basata su stati finiti, che consente di ridurre la ridondanza mantenendo uno stato tra la codifica di un simbolo e il successivo. Questo approccio permette di rappresentare i dati in modo più compatto rispetto ad altre tecniche tradizionali.
- **Codifica di Huffman:** un metodo che assegna codici di lunghezza variabile ai simboli in base alla loro frequenza, con i simboli più comuni rappresentati da codici più brevi. Questo consente di migliorare ulteriormente il rapporto di compressione.

Nel processo di compressione di Zstandard, i dati vengono inizialmente codificati con la codifica di Huffman. Successivamente, le informazioni strutturali, come le lunghezze dei literal, le lunghezze delle corrispondenze e gli offset, vengono compresse utilizzando FSE. Inoltre, FSE è impiegato per comprimere le intestazioni degli alberi di Huffman, contribuendo a una maggiore efficienza complessiva.

6.5 Compressione/Decompressione con autoencoder

Per la costruzione e l’addestramento della rete neurale è stato necessario effettuare una fase preliminare di ridimensionamento delle immagini alla risoluzione di **960x720 pixel**. Questa scelta è risultata obbligatoria sia per limitazioni hardware, sia per esigenze progettuali, poiché la rete richiede immagini di dimensioni uniformi per garantire un’elaborazione coerente.

La rete neurale progettata per questa sperimentazione è strutturata in due blocchi simmetrici:

- **Encoder:** costituito da 4 layer convoluzionali alternati a 4 layer di pooling, con l’utilizzo della funzione di attivazione **ReLU**. La scelta di ReLU è motivata dalla sua capacità di individuare pattern nascosti nei dati di training grazie alla sua non linearità e dai vantaggi in termini di efficienza computazionale.
- **Decoder:** speculare all’encoder, con la differenza che l’ultimo layer utilizza una funzione di attivazione **sigmoide**. Questa funzione normalizza i dati nell’intervallo $[0, 1]$, garantendo una ricostruzione accurata dell’immagine.

Per la fase di addestramento è stato scelto l’ottimizzatore **AdamW**, che introduce una regolarizzazione efficace sui pesi del modello separando il decadimento dei pesi dagli aggiornamenti basati sul gradiente. Ciò consente una migliore capacità di generalizzazione e una convergenza più rapida.

Come funzione di perdita è stato utilizzato il **Mean Squared Error (MSE)**, in quanto permette di focalizzarsi sulla ricostruzione fedele dei pixel dell’immagine originale, minimizzando l’errore complessivo.

Allo stesso modo delle tecniche di riduzione della dimensionalità, per la fase di compressione e invio dei dati abbiamo inserito un’ulteriore fase di compressione entropica utilizzando l’algoritmo **Zstandard** per ridurre ulteriormente la rappresentazione latente dei dati compressi forniti dall’encoder.

6.6 Valutazione dei risultati

I frame decompressi vengono confrontati con i frame originali nel modulo **results_evaluation.py**. Grazie all’invocazione del comando

```
1 python results_evaluation.py [DECOMPRESSED PATH]
2                               [ORIGINAL PATH]
3                               [COMPRESSED VIDEO PATH]
4                               [MODE]
```

vengono confrontati pixel per pixel tutti i frame con la stessa numerazione del dataset passato come input e ad ogni ciclo di esecuzione vengono forniti in output i valori relativi al SSIM e al PSNR per il singolo frame. Questi valori verranno poi salvati e dati in output sotto forma di valore medio per il singolo dataset. Inoltre viene fornito anche il rapporto di compressione ottenuto mettendo in relazione il dataset originale e il video compresso.

CAPITOLO 7

Risultati

7.1 Configurazione Hardware/Software

Gli esperimenti condotti in questo studio sono stati effettuati con diversi tipi di macchine ottimizzate per le varie fasi del progetto. Per la fase di compressione/decompressione utilizzando i codec tradizionali e di applicazione di PCA/SVD, è stata utilizzata una macchina dotata di processore Ryzen 5 5600x, 16GB di ram e scheda grafica RTX 3060 12GB OC. La scelta di questa configurazione è stata presa per dare continuità anche con l'hardware utilizzato in precedenza, mantenendo quindi la stessa famiglia di processori (aggiornati però a quelli più recenti) e mantenendo quindi la stessa architettura lato cpu.

Per l'addestramento della rete neurale è stato utilizzata la piattaforma Google Colab utilizzando macchine virtuali su hardware con schede video apposite.

Per quanto riguarda il lato software è stato preferito l'utilizzo di un container docker con immagine Ubuntu:24.04 per garantire portabilità ed un ambiente di sviluppo controllato.

7.2 Risultati codec tradizionali

Di seguito vengono mostrati i risultati ottenuti per le fasi di compressione/decompressione su ogni dataset utilizzato. Verranno messi a confronto i comportamenti dei codec sullo stesso dataset mettendo in evidenza:

- **Dimensione Iniziale** del dataset utilizzato e **Dimensione Finale** del file video risultante.

- **Rapporto di compressione e Tempo di compressione** ottenuti e misurati durante la fase di compressione.
- **Structural Similarity Index (SSIM)** e **Peak Signal-to-Noise Ratio (PSNR)** ottenuti dalla media su ogni frame del dataset laddove i codec utilizzati sono in modalità lossy.

7.2.1 Risultati codifica lossless

Dataset Room				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	89.26 MB	45.922042 MB	1.9437310518552289	18.83 s
AV1	89.26 MB	50.358712 MB	1.7724857418910953	973.59 s
VP9	89.26 MB	54.234268 MB	1.6458247210048083	21.82 s
HUFFYUV	89.26 MB	84.401008 MB	1.0575714806628849	1.23 s
FFV1	89.26 MB	65.35137 MB	1.3658489332358297	2.19 s
UTVIDEO	89.26 MB	81.902122 MB	1.089838661323085	1.32 s
DIRAC	89.26 MB	62.674506 MB	1.4241851224164415	3.86 s
MS-RLE	89.26 MB	67.66914 MB	1.3190665493901652	2.60 s

Tabella 7.1: Risultati di testing con codec lossless su dataset "Room"

Dataset Blob				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	53.65 MB	20.277525 MB	2.646068590718049	14.07 s
AV1	53.65 MB	23.201484 MB	2.312598711358291	684.78 s
VP9	53.65 MB	47.779522 MB	1.1229857427204901	26.07 s
HUFFYUV	53.65 MB	65.16664 MB	0.8233617998411458	0.97 s
FFV1	53.65 MB	32.667866 MB	1.642461800229008	1.60 s
UTVIDEO	53.65 MB	53.500422 MB	1.0029027808416164	1.01 s
DIRAC	53.65 MB	58.472368 MB	0.9176252619014849	3.01 s
MS-RLE	53.65 MB	65.901612 MB	0.8141792039927642	2.04 s

Tabella 7.2: Risultati di testing con codec lossless su dataset "Blob"

Dataset Car				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	63.50 MB	29.539703 MB	2.1496721547945152	15.55 s
AV1	63.50 MB	36.225283 MB	1.7529380515812671	1387.18 s
VP9	63.50 MB	45.082057 MB	1.4085576663016952	22.11 s
HUFFYUV	63.50 MB	65.438568 MB	0.9703861031922336	1.02 s
FFV1	63.50 MB	36.14634 MB	1.756766438870436	1.67 s
UTVIDEO	63.50 MB	53.141066 MB	1.1949454871680594	1.04 s
DIRAC	63.50 MB	56.220836 MB	1.1294865305809398	2.25 s
MS-RLE	63.50 MB	70.215302 MB	0.9043709161857625	1.93 s

Tabella 7.3: Risultati di testing con codec lossless su dataset "Car"

Dataset Cobblestone				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	102.79 MB	50.282646 MB	2.044356536050231	18.49 s
AV1	102.79 MB	51.228589 MB	2.006607209111303	839.03 s
VP9	102.79 MB	83.43727 MB	1.232011258278225	25.22 s
HUFFYUV	102.79 MB	99.919324 MB	1.028786543832102	1.42 s
FFV1	102.79 MB	75.263866 MB	1.365803558376871	2.49 s
UTVIDEO	102.79 MB	97.681582 MB	1.052354537009853	1.31 s
DIRAC	102.79 MB	59.568074 MB	1.7256837278304482	2.99 s
MS-RLE	102.79 MB	70.205926 MB	1.4642019820378125	1.85 s

Tabella 7.4: Risultati di testing con codec lossless su dataset "Cobblestone"

Dataset Dice				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	3.13 MB	3.112495 MB	1.0061198491885128	1.76 s
AV1	3.13 MB	3.144974 MB	0.9957293764590741	53.22 s
VP9	3.13 MB	3.394978 MB	0.9224045045358173	3.22 s
HUFFYUV	3.13 MB	10.826588 MB	0.2892456053559995	0.53 s
FFV1	3.13 MB	2.163444 MB	1.4474804986863539	0.59 s
UTVIDEO	3.13 MB	6.663838 MB	0.4699308416561147	0.50 s
DIRAC	3.13 MB	9.5773 MB	0.32697555678531526	0.52 s
MS-RLE	3.13 MB	12.35909 MB	0.2533797391231879	0.44 s

Tabella 7.5: Risultati di testing con codec lossless su dataset "Dice"

Dataset Dragons & Bunnies				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	11.79 MB	6.87641 MB	1.7160067535240044	2.98 s
AV1	11.79 MB	7.805644 MB	1.5117222871040494	132.65 s
VP9	11.79 MB	8.535186 MB	1.3825083600990067	4.34 s
HUFFYUV	11.79 MB	15.779892 MB	0.7477849658286635	0.65 s
FFV1	11.79 MB	8.876274 MB	1.3293828018377982	0.72 s
UTVIDEO	11.79 MB	13.369862 MB	0.8825794910972156	0.56
DIRAC	11.79 MB	13.143299 MB	0.8977933165790415	0.78 s
MS-RLE	11.79 MB	13.002676 MB	0.9075028863289372	0.56 s

Tabella 7.6: Risultati di testing con codec lossless su dataset "Dragons & Bunnies"

Dataset Fish				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	7.35 MB	3.486439 MB	2.1104591246254416	1.82 s
AV1	7.35 MB	3.780963 MB	1.9460616250410279	66.73 s
VP9	7.35 MB	4.26184 MB	1.7264812850787454	3.11 s
HUFFYUV	7.35 MB	14.194972 MB	0.5183516388760753	0.55 s
FFV1	7.35 MB	5.992874 MB	1.2277893711765007	0.65 s
UTVIDEO	7.35 MB	11.167262 MB	0.6588890813164409	0.54 s
DIRAC	7.35 MB	10.18342 MB	0.7225457655679526	0.64 s
MS-RLE	7.35 MB	4.352048 MB	1.6906952772579715	0.47 s

Tabella 7.7: Risultati di testing con codec lossless su dataset "Fish"

Dataset Mannequin				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	79.96 MB	72.74602 MB	1.0992893356914921	24.39 s
AV1	79.96 MB	63.306464 MB	1.2632031383082776	1045.08 s
VP9	79.96 MB	73.664008 MB	1.0855901840149669	26.34 s
HUFFYUV	79.96 MB	86.783324 MB	0.9214780019258078	1.19 s
FFV1	79.96 MB	51.220344 MB	1.5612726849315968	1.90 s
UTVIDEO	79.96 MB	80.218574 MB	0.9968878778623016	1.20 s
DIRAC	79.96 MB	58.807385 MB	1.3598449242386819	2.20 s
MS-RLE	79.96 MB	70.625238 MB	1.1322995329233438	2.16 s

Tabella 7.8: Risultati di testing con codec lossless su dataset "Mannequin"

Dataset Messerschmitt				
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione
HEVC	8.85 MB	6.46197 MB	1.3709690698037904	3.63 s
AV1	8.85 MB	8.113527 MB	1.0919001070681098	102.45 s
VP9	8.85 MB	9.120755 MB	0.971318821742279	4.53 s
HUFFYUV	8.85 MB	11.756164 MB	0.7535758262644175	0.53 s
FFV1	8.85 MB	5.149332 MB	1.7204485941089058	0.64 s
UTVIDEO	8.85 MB	8.61793 MB	1.0279917567211616	0.53 s
DIRAC	8.85 MB	14.229117 MB	0.6226079243005732	0.67 s
MS-RLE	8.85 MB	12.287512 MB	0.7209890008652687	0.59 s

Tabella 7.9: Risultati di testing con codec lossless su dataset "Messerschmitt"

7.2.2 Risultati codifica lossy

Dataset Room						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	89.26 MB	6.844156 MB	13.041797849143123	9.84 s	0.985503	46.729196 db
AV1	89.26 MB	10.974918 MB	8.133099399922624	4707 s	0.994106	51.183961 db
VP9	89.26 MB	21.956528 MB	4.065310280386772	78.01 s	0.994100	51.376430 db

Tabella 7.10: Risultati di testing con i codec lossy su dataset "Room"

Dataset Blob						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	53.65 MB	1.859889 MB	28.84888399253934	4.69 s	0.996165	51.919650 db
AV1	53.65 MB	2.764076 MB	19.411811397371128	1463.4 s	0.998507	56.109551 db
VP9	53.65 MB	5.338641 MB	10.050445796973424	28.83 s	0.997751	54.723596 db

Tabella 7.11: Risultati di testing con i codec lossy su dataset "Blob"

Dataset Car						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	63.5 MB	3.193158 MB	19.88648134542669	5.92 s	0.993617	49.253664 db
AV1	63.5 MB	6.766472 MB	9.384606483260406	5455.6 s	0.997647	54.059200 db
VP9	63.5 MB	11.886147 MB	5.3424105389240095	48.87 s	0.996156	52.707646 db

Tabella 7.12: Risultati di testing con i codec lossy su dataset "Car"

Dataset Cobblestone						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	102.79 MB	9.820082 MB	10.467901999189008	11.42 s	0.995688	45.617457 db
AV1	102.79 MB	17.421661 MB	5.9004509386332336	3046.1 s	0.998400	51.740324 db
VP9	102.79 MB	30.659907 MB	3.352771291837252	52.67 s	0.998221	52.262581 db

Tabella 7.13: Risultati di testing con i codec lossy su dataset "Cobblestone"

Dataset Dice						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	3.13 MB	1.392293 MB	2.249198265020366	1.75 s	0.997201	49.334695 db
AV1	3.13 MB	1.24892 MB	2.5074007942862635	3046.17 s	0.947228	25.943003 db
VP9	3.13 MB	2.165676 MB	1.4459886889821008	5.84 s	0.999591	59.284309 db

Tabella 7.14: Risultati di testing con i codec lossy su dataset "Dice"

Dataset Dragons & Bunnies						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	11.79 MB	2.156162 MB	5.472671348442279	2.36 s	0.992033	44.914928 db
AV1	11.79 MB	2.968382 MB	3.975218149146572	533.99 s	0.985377	34.892404 db
VP9	11.79 MB	5.242096 MB	2.2510015077938292	11.16 s	0.998658	54.390578 db

Tabella 7.15: Risultati di testing con i codec lossy su dataset "Dragons & Bunnies"

Dataset Fish						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	7.35 MB	1.126768 MB	6.530170363375602	1.57 s	0.996240	48.069559 db
AV1	7.35 MB	1.464882 MB	5.022921300145677	396.59 s	0.999302	55.438439 db
VP9	7.35 MB	2.443049 MB	3.011804920818207	5.97 s	0.999464	57.308073 db

Tabella 7.16: Risultati di testing con i codec lossy su dataset "Fish"

Dataset Mannequin						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	76.96 MB	12.173003 MB	6.569366983644052	13.5 s	0.989400	45.428984 db
AV1	76.96 MB	26.046136 MB	3.0702797528201495	6942.1 s	0.995820	51.073931 db
VP9	76.96 MB	38.614363 MB	2.07096318020318	37.25 s	0.996839	52.320795 db

Tabella 7.17: Risultati di testing con i codec lossy su dataset "Mannequin"

Dataset Messerschmitt						
Algoritmo	Dimensione iniziale	Dimesione Finale	Rapporto di compressione	Tempo di compressione	SSIM	PSNR
HEVC	8.85 MB	2.615783 MB	3.386810373796297	2.89 s	0.988935	45.225550 db
AV1	8.85 MB	2.296094 MB	3.858361635020169	516.52 s	0.996613	51.647442 db
VP9	8.85 MB	5.200209 MB	1.7036163354203648	15.64 s	0.997490	53.259706 db

Tabella 7.18: Risultati di testing con i codec lossy su dataset "Messerschmitt"

7.3 Risultati PCA/SVD

Per l'analisi dei risultati delle tecniche di riduzione della dimensionalità sono stati considerati i seguenti parametri:

- **Dimensione Iniziale** del sottoinsieme di dati iniziali.
- **Dimensione Finale** dei dati iniziali compressi e inviati al decodificatore, con valori relativi alla presenza e assenza di compressione entropica.
(ES: dimensione senza compressione / dimensione con compressione)
- **Tempo di compressione** ottenuto e misurato durante la fase di compressione, con valori relativi alla presenza e assenza di compressione entropica.
(ES: tempo senza compressione / tempo con compressione)
- **Numero di componenti** utilizzate per la valutazione dei modelli.
- **Structural Similarity Index (SSIM)** e **Peak Signal-to-Noise Ratio (PSNR)** ottenuti dalla media su ogni frame dei dati trasmessi.

È importante evidenziare che, a differenza degli esperimenti condotti con i codec tradizionali, il numero di dati utilizzati è limitato, poiché una parte di essi è stata impiegata per l'addestramento del modello su ciascun dataset. In generale, la suddivisione tra il training set e il test set è stata effettuata in una proporzione 70% - 30%. Di conseguenza, il numero massimo di componenti su cui lavorare corrisponde al numero di elementi presenti nel training set. I dati relativi alle dimensioni iniziali e finali dei dati compressi e inviati si riferiscono a un sottoinsieme del dataset complessivo, nello specifico il test set ottenuto dallo split precedentemente descritto.

7.3.1 Dataset ArtGallery

Su 101 frames disponibili, 70 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	27.4 MB	0.94 KB / 0.53 KB	7.355 / 7.369	0.839593	16.454604	7.948 / 7.964	0.817626	18.673158
11	27.4 MB	3.36 KB / 2.88KB	8.109 / 8.128	0.922033	19.177110	10.904 / 10.916	0.930408	20.565870
21	27.4 MB	5.79 KB / 5.18 KB	12.170 / 12.194	0.961933	24.087929	14.718 / 14.729	0.960036	23.923026
31	27.4 MB	8.21 KB / 7.49 KB	15.846 / 15.866	0.967657	26.020578	18.943 / 18.955	0.969555	26.398225
40	27.4 MB	10.39 KB / 9.58 KB	21.677 / 21.699	0.956441	24.122979	23.600 / 23.615	0.953545	23.144019
50	27.4 MB	12.81 KB / 11.86 KB	26.230 / 26.244	0.974743	30.310498	28.129 / 28.140	0.974968	30.374689
60	27.4 MB	15.23 KB / 14.2 KB	25.942 / 25.953	0.975919	31.108428	33.769 / 33.783	0.976207	31.134145

Tabella 7.19: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su ArtGallery.

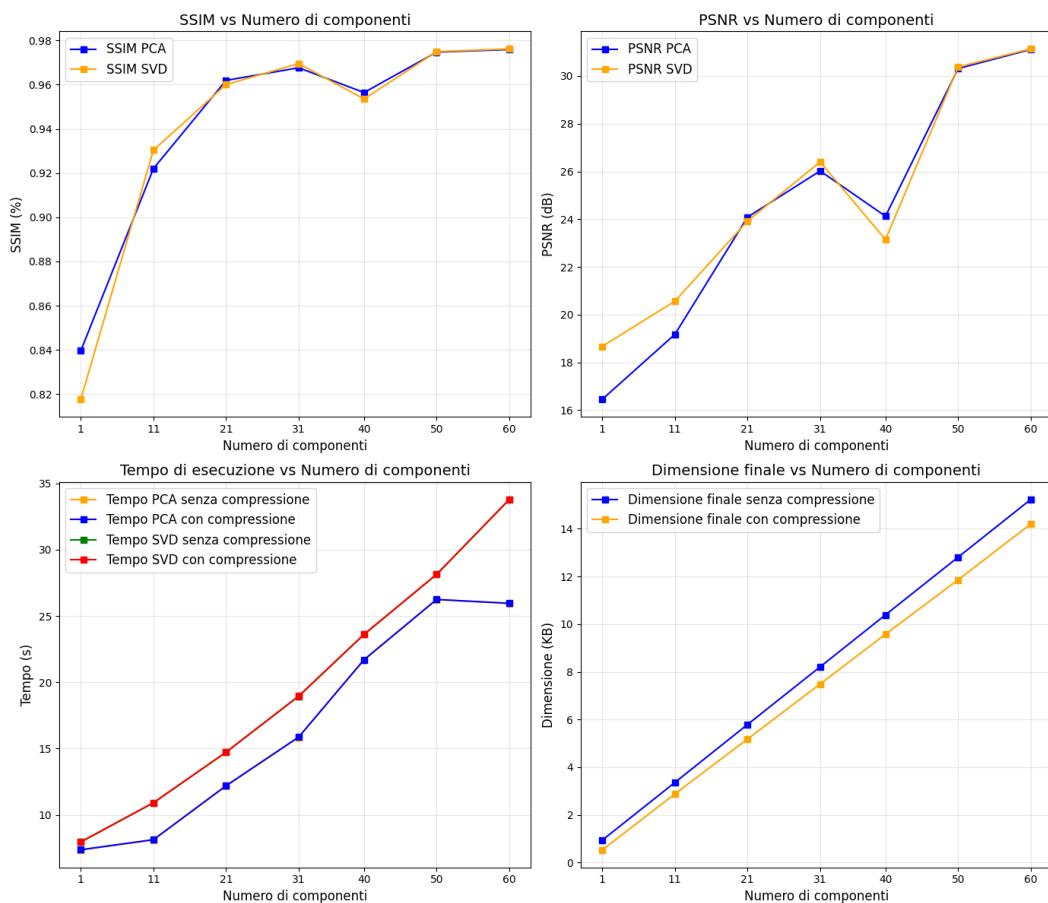


Figura 7.1: Andamento PCA e SVD su ArtGallery

7.3.2 Dataset Blob

Su 101 frames disponibili, 70 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	16.5 MB	0.94 KB / 0.53 KB	7.059 / 7.078	0.717196	17.630212	8.008 / 8.022	0.675957	18.821064
11	16.5 MB	3.36 KB / 2.88KB	7.677 / 7.695	0.84753	19.046673	12.530 / 12.543	0.851769	19.403665
21	16.5 MB	5.79 KB / 5.18 KB	11.327 / 11.343	0.897602	21.547339	15.550 / 15.562	0.898408	21.194338
31	16.5 MB	8.21 KB / 7.49 KB	14.899 / 14.915	0.901246	22.173558	20.069 / 20.082	0.903839	22.65663
40	16.5 MB	10.39 KB / 9.58 KB	20.207 / 20.227	0.888926	19.956837	27.689 / 27.701	0.887172	19.752524
50	16.5 MB	12.81 KB / 11.86 KB	23.654 / 23.678	0.926535	24.087861	28.396 / 28.408	0.927054	24.14764
60	16.5 MB	15.23 KB / 14.2 KB	25.029 / 25.069	0.926367	23.951124	33.183 / 33.198	0.926864	24.021371

Tabella 7.20: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Blob.

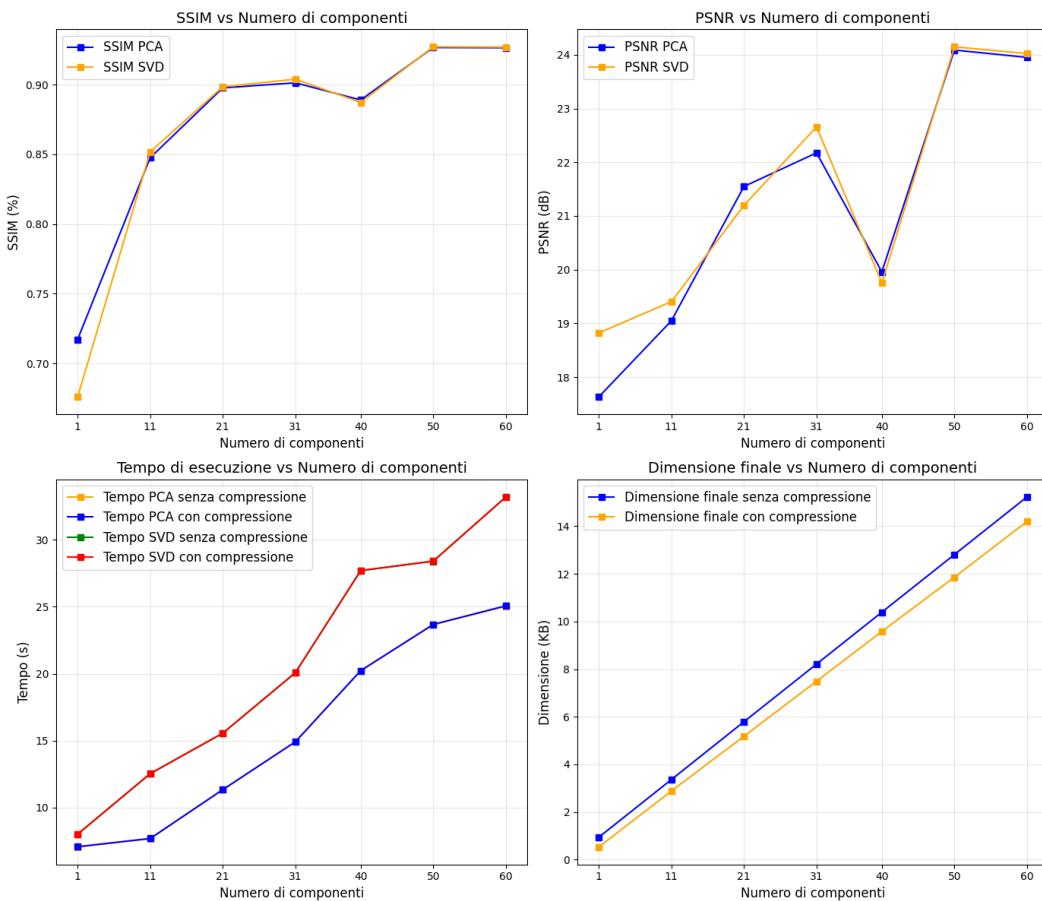


Figura 7.2: Andamento PCA e SVD su Blob

7.3.3 Dataset Car

Su 101 frames disponibili, 70 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	19.5 MB	0.94 KB / 0.53 KB	7.317 / 7.330	0.830241	23.356569	7.667 / 7.681	0.800187	21.611909
11	19.5 MB	3.36 KB / 2.88KB	7.369 / 7.384	0.927283	26.879416	11.293 / 11.305	0.914261	25.903732
21	19.5 MB	5.79 KB / 5.18 KB	10.805 / 10.825	0.954707	28.177387	16.686 / 16.701	0.954874	28.164272
31	19.5 MB	8.21 KB / 7.49 KB	14.429 / 14.444	0.952251	27.290834	19.899 / 19.911	0.953022	27.473111
40	19.5 MB	10.39 KB / 9.58 KB	19.287 / 19.309	0.960457	27.52977	26.291 / 26.302	0.95437	27.012108
50	19.5 MB	12.81 KB / 11.86 KB	23.820 / 23.839	0.968186	28.362289	37.563 / 37.577	0.968148	28.34708
60	19.5 MB	15.23 KB / 14.2 KB	24.079 / 24.097	0.968808	28.438704	40.137 / 40.152	0.968762	28.403629

Tabella 7.21: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Car.

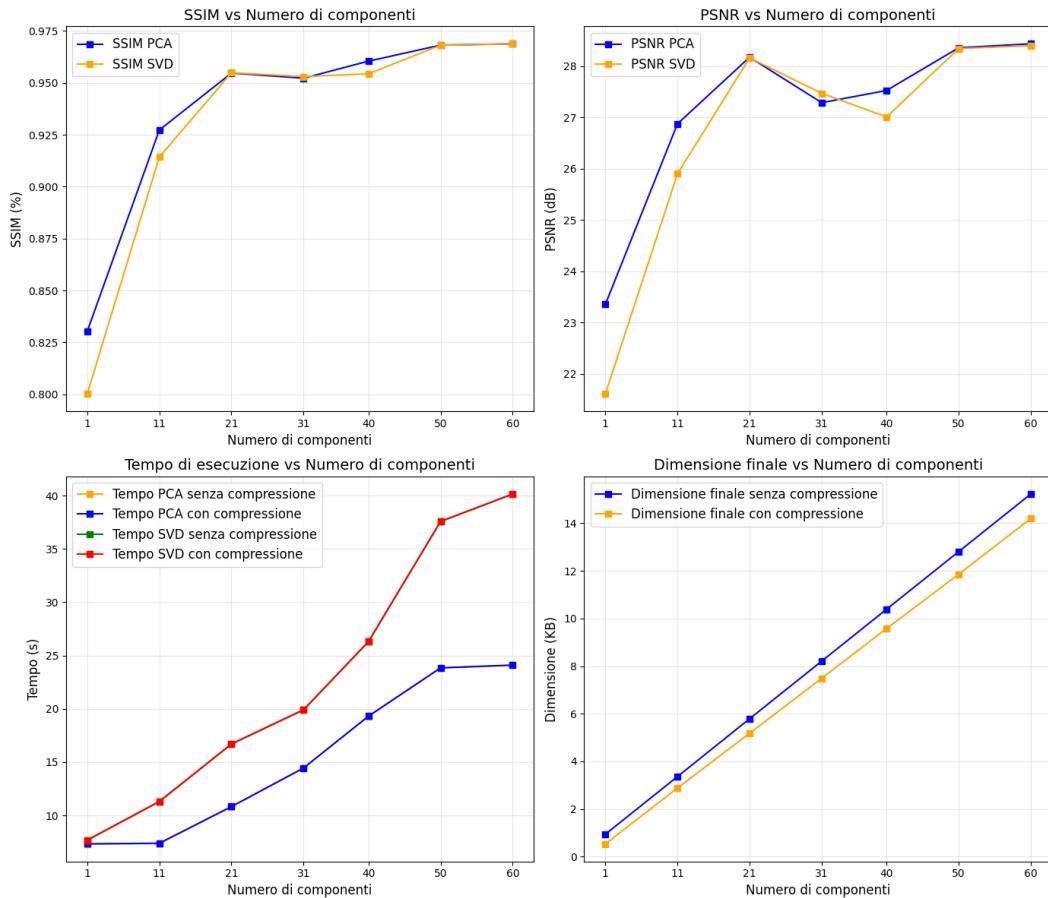


Figura 7.3: Andamento PCA e SVD su Car

7.3.4 Dataset Cobblestone

Su 101 frames disponibili, 70 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	31.5 MB	0.94 KB / 0.53 KB	7.620 / 7.635	0.685182	17.825953	8.208 / 8.222	0.635297	18.264537
11	31.5 MB	3.36 KB / 2.88 KB	7.714 / 7.730	0.855818	22.263653	11.792 / 11.803	0.857003	22.973967
21	31.5 MB	5.79 KB / 5.18 KB	11.271 / 11.289	0.930593	26.024262	15.283 / 15.294	0.931184	26.417875
31	31.5 MB	8.21 KB / 7.49 KB	15.286 / 15.304	0.935486	26.815666	19.739 / 19.758	0.93602	26.974395
40	31.5 MB	10.39 KB / 9.58 KB	19.866 / 19.882	0.936283	26.896366	28.341 / 28.354	0.938462	27.253004
50	31.5 MB	12.81 KB / 11.86 KB	23.916 / 23.934	0.966484	29.334732	35.523 / 35.540	0.966854	29.506809
60	31.5 MB	15.23 KB / 14.2 KB	25.214 / 25.246	0.967534	29.948481	34.709 / 34.721	0.967863	30.104938

Tabella 7.22: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Cobblestone.

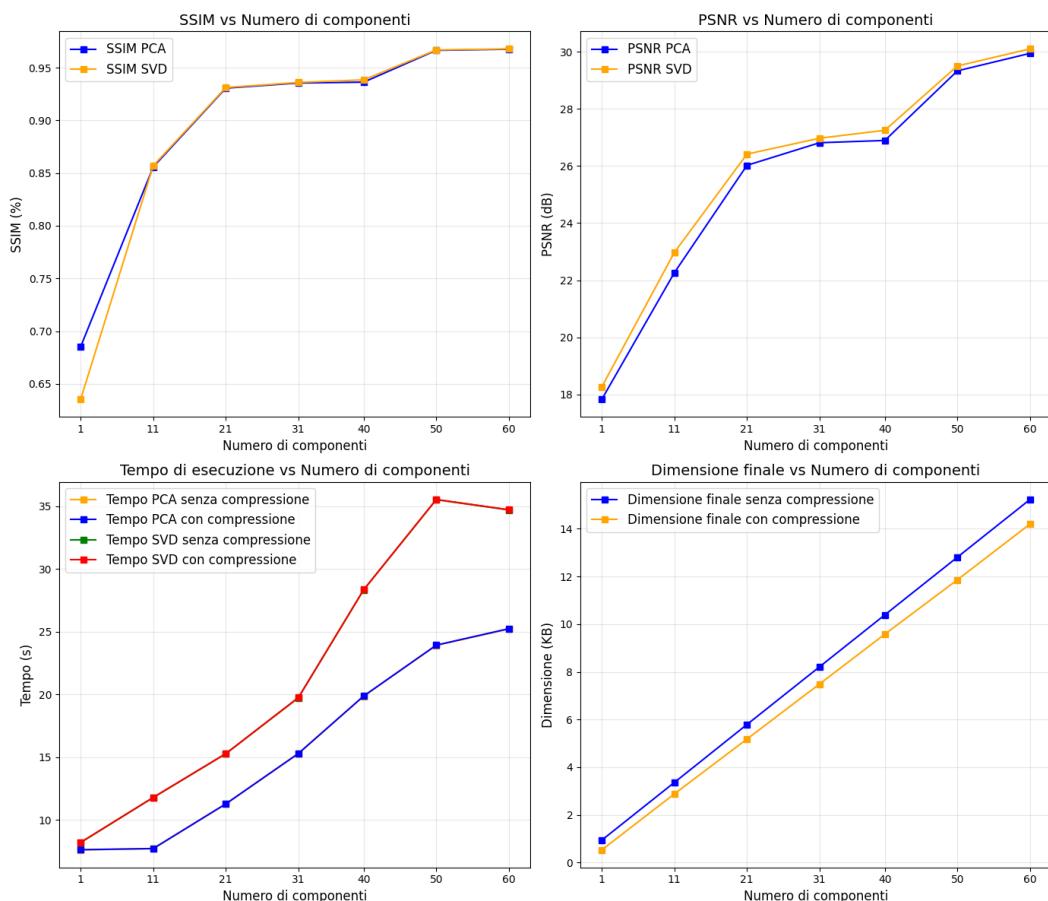


Figura 7.4: Andamento PCA e SVD su Cobblestone

7.3.5 Dataset Dice

Su 25 frames disponibili, 17 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	1 MB	0.40 KB / 0.31 KB	2.731 / 2.747	0.856841	24.042892	2.490 / 2.500	0.84097	22.911778
	3	0.52 KB / 0.43 KB	2.106 / 2.130	0.864213	25.360088	3.073 / 3.084	0.857619	25.234029
	6	0.71 KB / 0.62 KB	2.742 / 2.762	0.86328	25.987345	3.443 / 3.453	0.86425	25.989251
	8	0.84 KB / 0.75 KB	2.959 / 2.980	0.870984	26.393965	3.619 / 3.629	0.869092	26.263866
	10	0.96 KB / 0.87 KB	2.944 / 2.961	0.870879	26.424994	3.855 / 3.864	0.87105	26.403629
	12	1.09 KB / 0.98 KB	3.108 / 3.124	0.870282	26.438422	3.938 / 3.948	0.87085	26.425787
	15	1.28 KB / 1.17 KB	2.021 / 2.040	0.875017	26.551664	3.873 / 3.884	0.869879	26.458677

Tabella 7.23: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Dice.

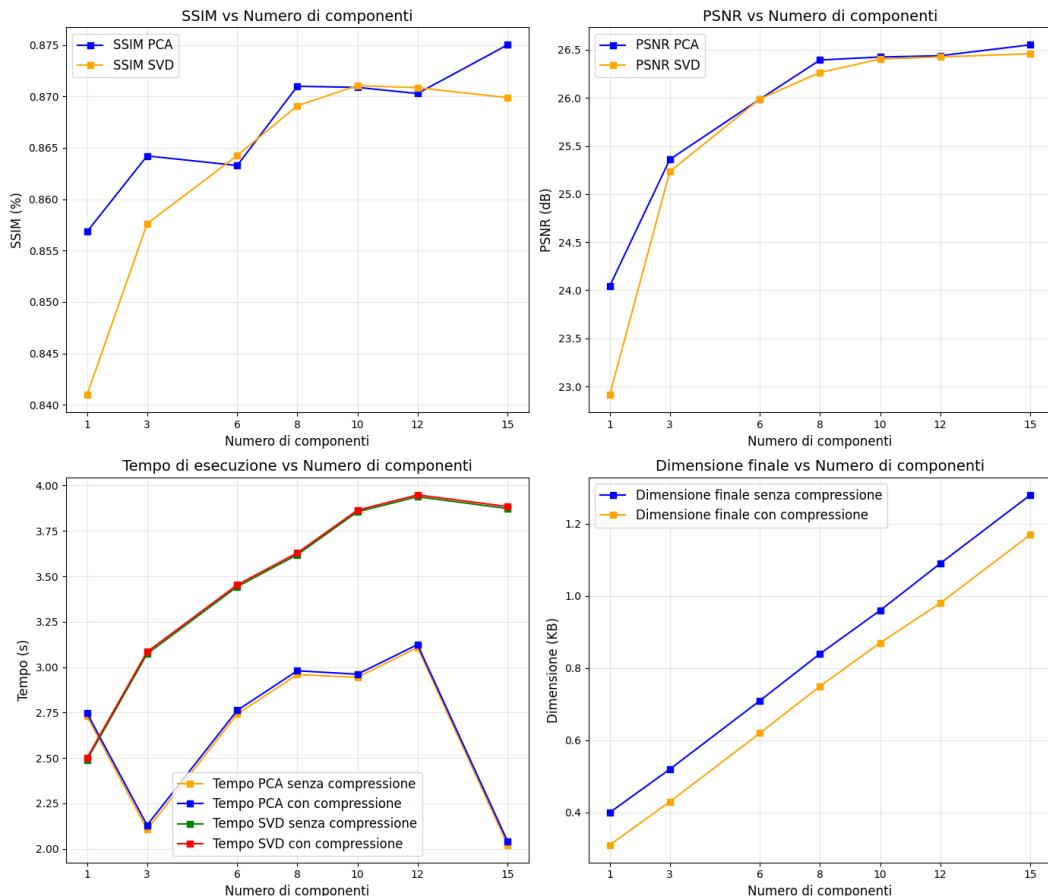


Figura 7.5: Andamento PCA e SVD su Dice

7.3.6 Dataset Dragons

Su 26 frames disponibili, 18 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	3.6 MB	0.40 KB / 0.31 KB	2.707 / 2.721	0.899128	28.278944	2.887 / 2.898	0.897983	28.232910
	3.6 MB	0.52 KB / 0.43 KB	2.278 / 2.293	0.900277	24.076280	3.394 / 3.406	0.913902	29.309934
	3.6 MB	0.71 KB / 0.62 KB	2.830 / 2.852	0.908039	23.631535	3.679 / 3.688	0.901116	23.183665
	3.6 MB	0.84 KB / 0.75 KB	3.330 / 3.342	0.915229	24.291877	4.294 / 4.306	0.908391	23.626631
	3.6 MB	1.03 KB / 0.92 KB	3.424 / 3.440	0.914832	23.945705	4.675 / 4.697	0.915783	24.106700
	3.6 MB	1.15 KB / 1.05 KB	3.521 / 3.537	0.913754	23.812443	4.886 / 4.898	0.914400	23.894681
	3.6 MB	1.34 KB / 1.24 KB	2.129 / 2.145	0.916405	24.141552	5.629 / 5.641	0.913879	23.830420

Tabella 7.24: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Dragons.

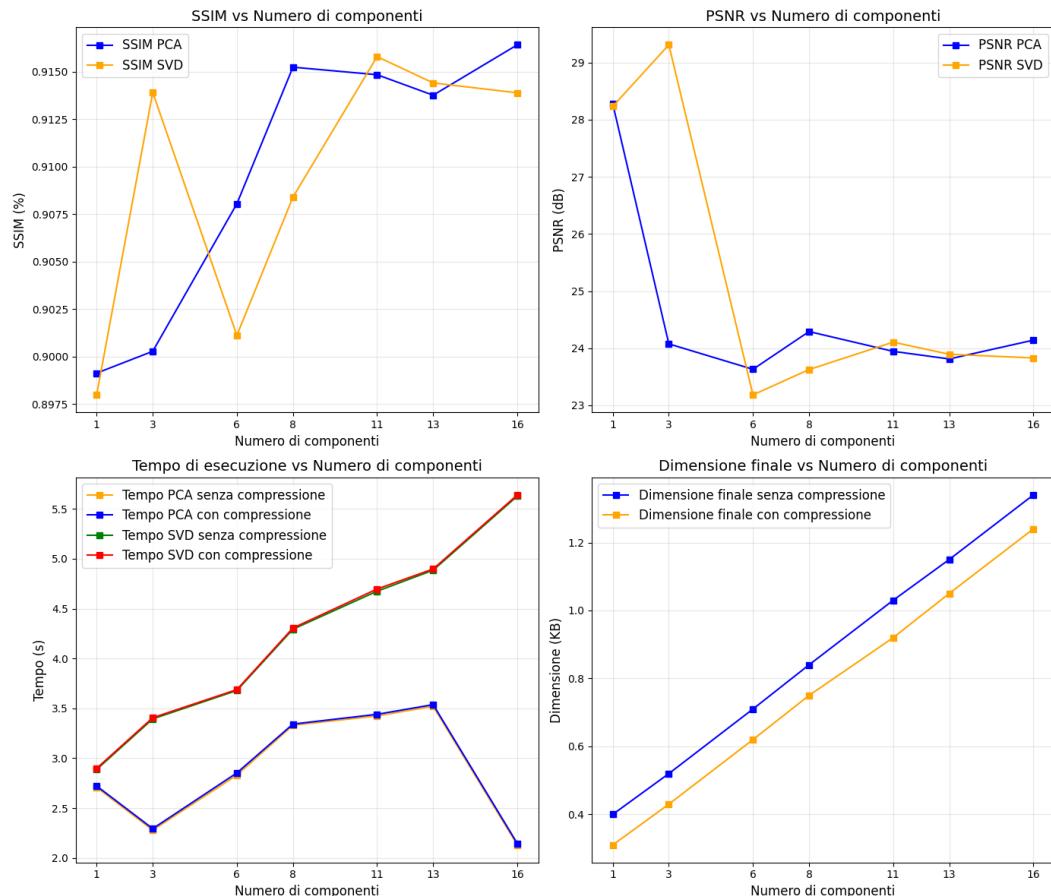


Figura 7.6: Andamento PCA e SVD su Dragons

7.3.7 Dataset Fish

Su 25 frames disponibili, 17 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	2.4 MB	0.40 KB / 0.31 KB	2.545 / 2.572	0.87429	17.974339	2.623 / 2.634	0.88774	22.292288
	2.4 MB	0.52 KB / 0.43 KB	2.425 / 2.443	0.889595	18.014187	3.504 / 3.516	0.891268	18.163817
	2.4 MB	0.71 KB / 0.62 KB	2.889 / 2.911	0.912876	20.187454	3.875 / 3.888	0.909776	20.080925
	2.4 MB	0.84 KB / 0.75 KB	3.014 / 3.029	0.90482	19.116036	4.692 / 4.703	0.908629	19.456255
	2.4 MB	0.96 KB / 0.87 KB	3.052 / 3.069	0.901418	18.492804	4.959 / 4.969	0.910058	19.086421
	2.4 MB	1.09 KB / 0.99 KB	3.099 / 3.115	0.914998	19.634306	4.852 / 4.862	0.904355	18.754379
	2.4 MB	1.28 KB / 1.18 KB	2.104 / 2.119	0.920399	20.054652	5.297 / 5.308	0.917612	20.064176

Tabella 7.25: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Fish.

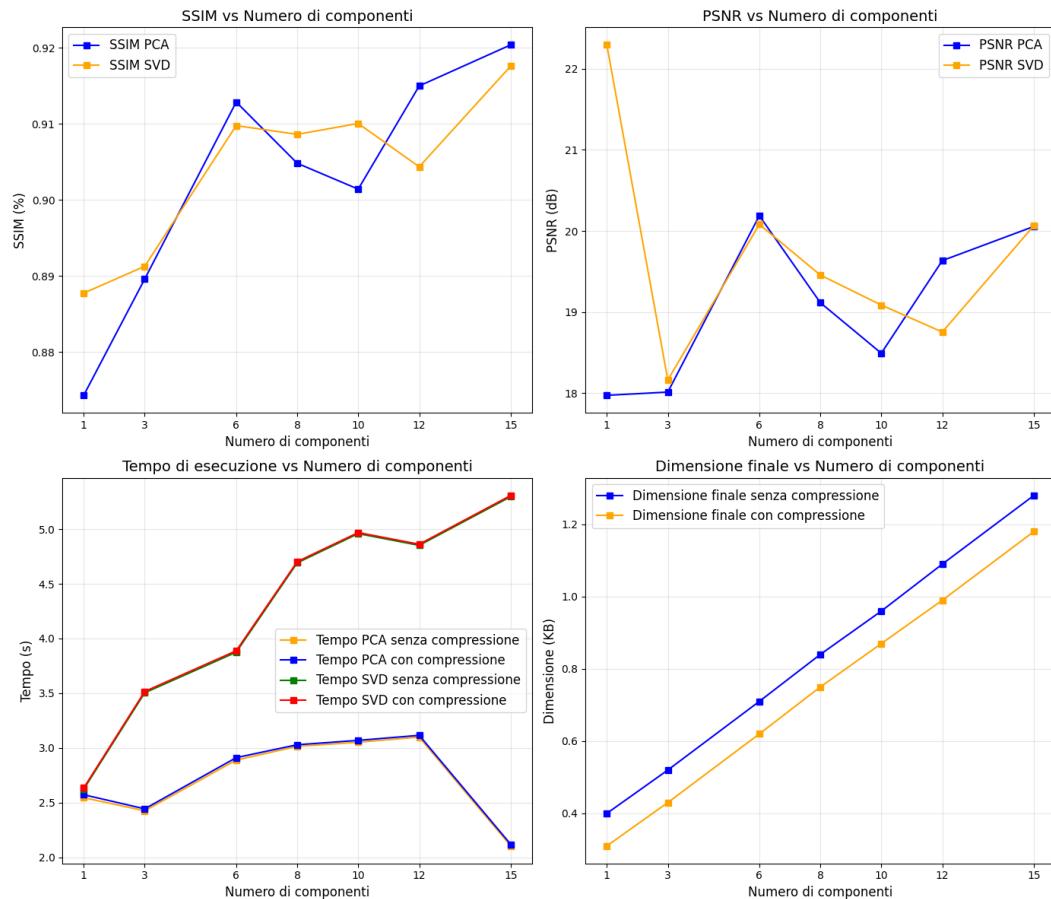


Figura 7.7: Andamento PCA e SVD su Fish

7.3.8 Dataset Mannequin

Su 101 frames disponibili, 70 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	24.5 MB	0.94 KB / 0.53 KB	7.268 / 7.280	0.766559	22.168989	8.467 / 8.483	0.704878	22.586695
11	24.5 MB	3.36 KB / 2.87 KB	7.680 / 7.694	0.921934	27.381849	12.453 / 12.465	0.918281	27.605645
21	24.5 MB	5.79 KB / 5.19 KB	11.225 / 11.244	0.960444	32.191975	16.561 / 16.573	0.959970	32.140186
31	24.5 MB	8.21 KB / 7.48 KB	14.634 / 14.650	0.960841	32.577691	21.571 / 21.584	0.961273	32.758053
40	24.5 MB	10.39 KB / 9.53 KB	18.985 / 19.002	0.961424	33.039452	29.105 / 29.122	0.959908	32.484246
50	24.5 MB	12.81 KB / 11.81 KB	23.275 / 23.293	0.963027	33.518551	33.130 / 33.144	0.963482	33.601573
60	24.5 MB	15.23 KB / 14.08 KB	24.277 / 24.304	0.963047	33.602202	34.026 / 34.039	0.964100	33.830004

Tabella 7.26: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Mannequin.

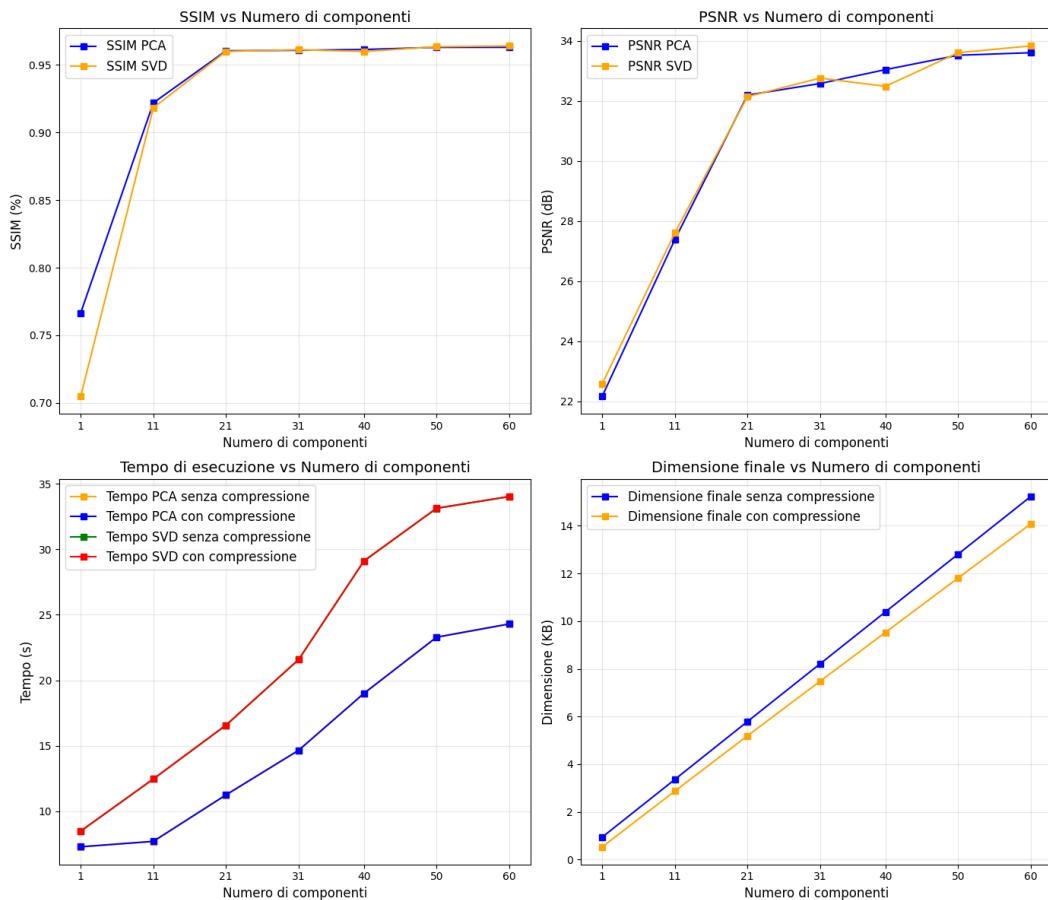


Figura 7.8: Andamento PCA e SVD su Mannequin

7.3.9 Dataset Messerschmitt

Su 25 frames disponibili, 17 sono stati utilizzati per l'addestramento del modello e i restanti per il test delle metriche.

Numero di componenti	Dim. iniziale	Dim. finale	PCA			SVD		
			Tempo (s)	SSIM	PSNR (db)	Tempo (s)	SSIM	PSNR (db)
1	2.8 MB	0.40 KB / 0.31 KB	2.719 / 2.734	0.666508	19.616771	2.416 / 2.426	0.635477	21.512253
	2.8 MB	0.52 KB / 0.43 KB	2.383 / 2.402	0.693333	21.320661	2.758 / 2.770	0.678003	20.245738
	2.8 MB	0.71 KB / 0.62 KB	2.827 / 2.842	0.698348	22.040359	3.295 / 3.306	0.696006	21.789397
	2.8 MB	0.84 KB / 0.75 KB	3.053 / 3.065	0.699954	22.557444	3.627 / 3.636	0.700149	22.671477
	2.8 MB	0.96 KB / 0.87 KB	3.649 / 3.662	0.709949	23.388046	3.809 / 3.818	0.700891	22.586415
	2.8 MB	1.09 KB / 0.99 KB	3.701 / 3.715	0.713383	23.976121	3.653 / 3.663	0.714857	24.329000
	2.8 MB	1.28 KB / 1.17 KB	2.094 / 2.079	0.716018	24.429946	3.871 / 3.881	0.717525	24.881448

Tavella 7.27: Confronto delle prestazioni di PCA e SVD per diversi numeri di componenti su Messerschmitt.

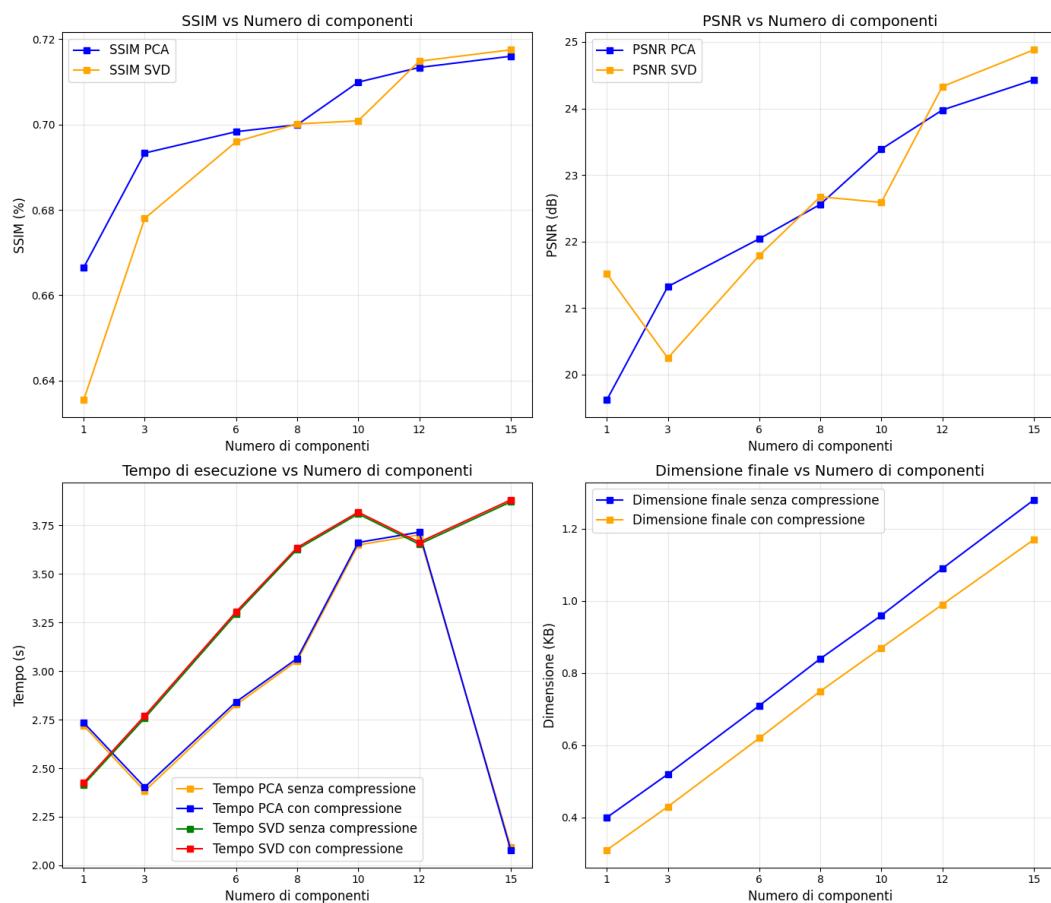


Figura 7.9: Andamento PCA e SVD su Messerschmitt

7.4 Risultati autoencoder

7.4.1 Train & Validation Loss

Il dataset utilizzato per le fasi di addestramento, validazione e testing è composto da tutte le immagini ridimensionate provenienti dai diversi dataset considerati nello studio. I dati sono stati suddivisi seguendo una ripartizione pari a **80%** per l’addestramento, **10%** per la validazione e **10%** per il testing.

Di seguito sono riportati i risultati dell’addestramento, con l’andamento della funzione di perdita relativo ai dati di training e testing.

Epochs	1	2	3	4	5	6	7	8	9	10
Train Loss	0.0449	0.0074	0.0046	0.0036	0.0032	0.0049	0.0228	0.0072	0.0112	0.0083
Validation Loss	0.0075	0.0056	0.0037	0.0032	0.0027	0.0366	0.0069	0.0088	0.0047	0.0054
<hr/>										
Epochs	11	12	13	14	15	16	17	18	19	20
Train Loss	0.0096	0.0057	0.0074	0.0038	0.0031	0.0030	0.0025	0.0024	0.0022	0.0025
Validation Loss	0.0061	0.0054	0.0037	0.0030	0.0034	0.0026	0.0022	0.0021	0.0021	0.0020
<hr/>										
Epochs	21	22	23	24	25	26	27	28	29	30
Train Loss	0.0022	0.0019	0.0020	0.0018	0.0018	0.0053	0.0063	0.0081	0.0039	0.0031
Validation Loss	0.0020	0.0017	0.0018	0.0019	0.0074	0.0067	0.0115	0.0041	0.0031	0.0028

Tabella 7.28: Valori di Train Loss e Validation Loss per ciascun Epoch.

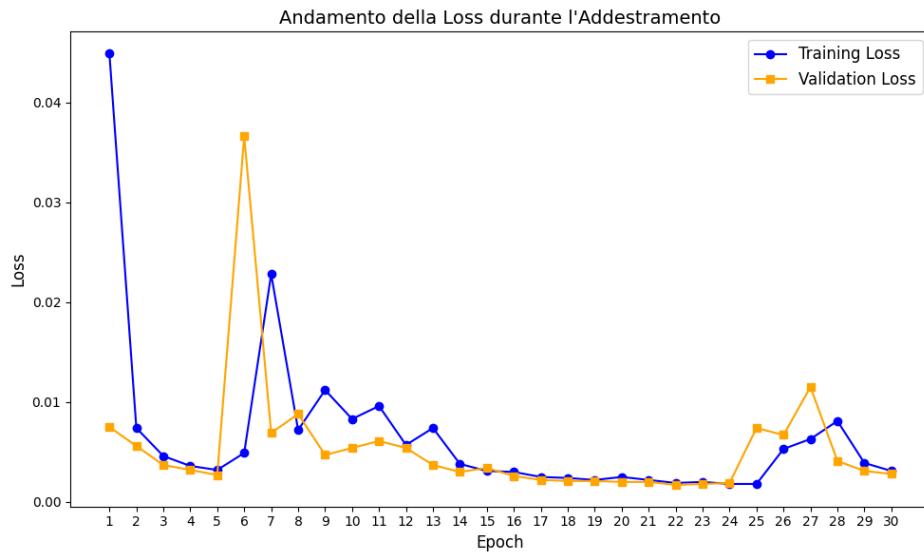


Figura 7.10: Train e Validation loss sul dataset di riferimento.

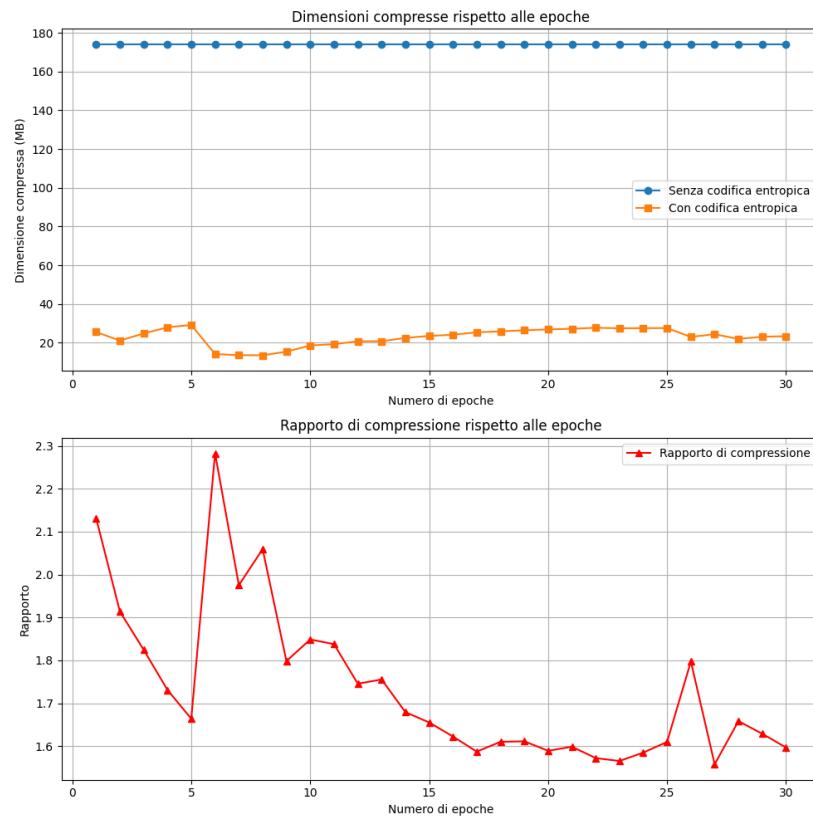
7.4.2 Dimensione dei dati & Rapporto di compressione

È stato effettuato un confronto tra le dimensioni dei dati in diversi punti della pipeline. In particolare, è stato testato l'effetto dell'uso di un codificatore entropico applicato ai dati trasformati nella loro forma latente, così come generati dal modello. Inoltre, è stato calcolato il rapporto di compressione dividendo la dimensione iniziale del dataset di test per quella finale, ottenuta dopo la ricostruzione delle immagini.

Epochs	1	2	3	4	5	6	7	8	9	10
Dim. Iniziale (MB)	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22
Dim. compresso senza codifica entropica (MB)	174.02	174.02	174.02	175.34	174.02	174.02	174.02	174.02	174.02	174.02
Dim. compresso con codifica entropica (MB)	25.51	21.13	24.74	27.98	29.15	14.14	13.55	13.50	15.38	18.58
Dim. Finale (MB)	21.69	24.15	25.33	26.70	27.77	20.25	23.39	22.45	25.70	25.00
Rapporto	2.1318	1.9139	1.8247	1.7311	1.6641	2.2827	1.9753	2.0594	1.7984	1.8488

Epochs	11	12	13	14	15	16	17	18	19	20
Dim. Iniziale (MB)	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22
Dim. compresso senza codifica entropica (MB)	174.02	174.02	174.02	174.02	174.02	174.02	174.02	174.02	174.02	174.02
Dim. compresso con codifica entropica (MB)	19.23	20.73	20.76	22.47	23.50	24.10	25.38	25.85	26.41	26.88
Dim. Finale (MB)	25.15	26.47	26.32	27.52	27.92	28.49	29.11	28.70	28.67	29.09
Rapporto	1.8379	1.7456	1.7556	1.6794	1.6554	1.6226	1.5874	1.6103	1.6114	1.5893

Epochs	21	22	23	24	25	26	27	28	29	30
Dim. Iniziale (MB)	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22	46.22
Dim. compresso senza codifica entropica (MB)	174.02	174.02	174.02	174.02	174.02	174.02	174.02	174.02	174.02	174.02
Dim. compresso con codifica entropica (MB)	27.17	27.72	27.44	27.52	27.55	22.98	24.39	21.94	23.10	23.34
Dim. Finale (MB)	28.92	29.39	29.52	29.16	28.71	25.72	29.66	27.87	28.37	28.95
Rapporto	1.5988	1.5723	1.5655	1.5851	1.6100	1.7975	1.5580	1.6585	1.6290	1.5969

Tabella 7.29: Tabella delle dimensioni per Epoch con il rapporto di compressione.**Figura 7.11:** Rapporto di compressione per ogni Epoch.

7.4.3 Tempi di compressione

Sono stati registrati anche i tempi di compressione, considerando separatamente il tempo necessario per la compressione, quello aggiuntivo richiesto dalla codifica entropica e il tempo complessivo risultante.

Epoche	1	2	3	4	5	6	7	8	9	10
Tempo di Compressione (s)	64.00	59.41	59.87	62.65	65.45	65.43	61.93	63.47	62.87	61.92
Tempo Codifica Entropica (s)	31.54	32.82	33.89	25.92	25.67	26.19	26.07	26.21	25.64	26.25
Tempo Totale (s)	95.54	92.23	93.76	88.57	91.12	91.62	88.00	89.68	88.51	88.17

Epoche	11	12	13	14	15	16	17	18	19	20
Tempo di Compressione (s)	65.38	74.06	64.34	68.68	67.18	70.06	70.79	72.28	72.56	69.29
Tempo Codifica Entropica (s)	25.96	25.19	31.39	26.85	26.13	25.50	25.99	26.57	27.45	26.44
Tempo Totale (s)	91.34	99.25	95.72	95.53	93.31	95.55	96.78	98.86	100.01	95.73

Epoche	21	22	23	24	25	26	27	28	29	30
Tempo di Compressione (s)	69.19	73.51	67.47	69.91	69.37	72.04	70.73	68.42	69.24	69.72
Tempo Codifica Entropica (s)	26.50	27.12	26.54	26.09	25.81	25.88	26.66	26.13	26.43	25.11
Tempo Totale (s)	95.69	100.62	94.00	96.00	95.18	97.92	97.39	94.55	95.67	94.83

Tabella 7.30: Tempi di compressione e codifica entropica per epoca.

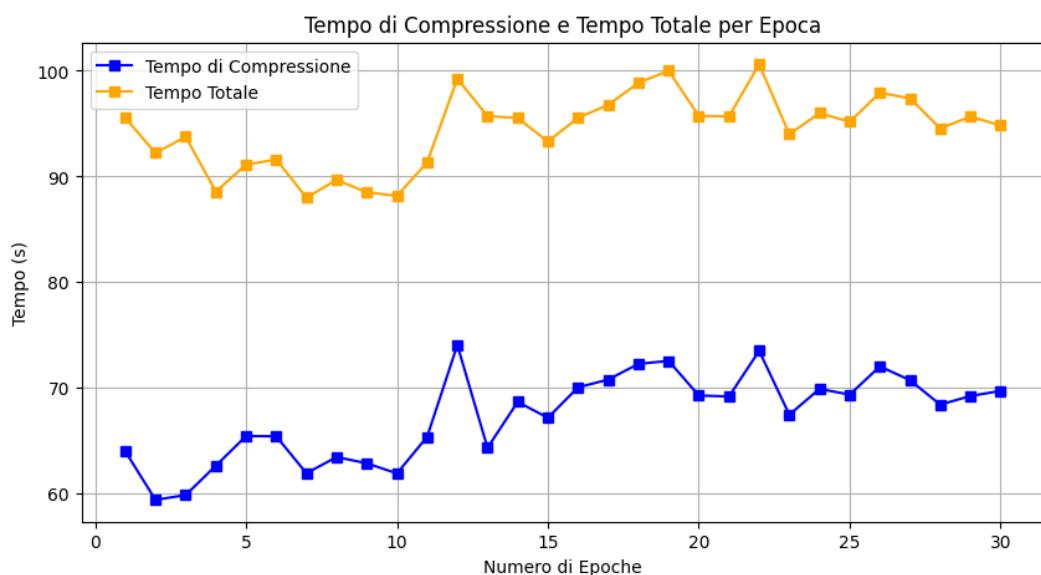


Figura 7.12: Tempo di compressione e totale per ogni epoch.

7.4.4 Metriche SSIM & PSNR

Le metriche SSIM e PSNR sono state calcolate confrontando le immagini del dataset di test con quelle processate e ricostruite da ciascun modello salvato tramite checkpoint durante la fase di addestramento. Questo approccio consente di monitorare l'evoluzione delle metriche visive in relazione alla loss registrata a ogni epoca di training.

Epochs	1	2	3	4	5	6	7	8	9	10
SSIM	0.807223	0.808263	0.837779	0.846401	0.857456	0.768931	0.808735	0.814435	0.825814	0.820059
PSNR	23.080706	23.457380	25.606309	25.765440	26.653386	18.444387	23.632753	23.343533	25.344902	25.045430
Epochs	11	12	13	14	15	16	17	18	19	20
SSIM	0.820981	0.819906	0.834162	0.837852	0.835027	0.842921	0.848830	0.853307	0.855639	0.857455
PSNR	25.153869	25.272871	25.865402	26.219113	25.773821	26.357748	26.629119	26.837197	26.887446	26.915344
Epochs	21	22	23	24	25	26	27	28	29	30
SSIM	0.860958	0.865396	0.864806	0.867825	0.834639	0.811821	0.831888	0.816515	0.838149	0.842964
PSNR	26.942715	27.363051	27.193375	26.941089	23.492196	24.268306	25.700318	25.134393	26.099779	26.237189

Tabella 7.31: Risultati delle metriche SSIM e PSNR per ogni Epoch.

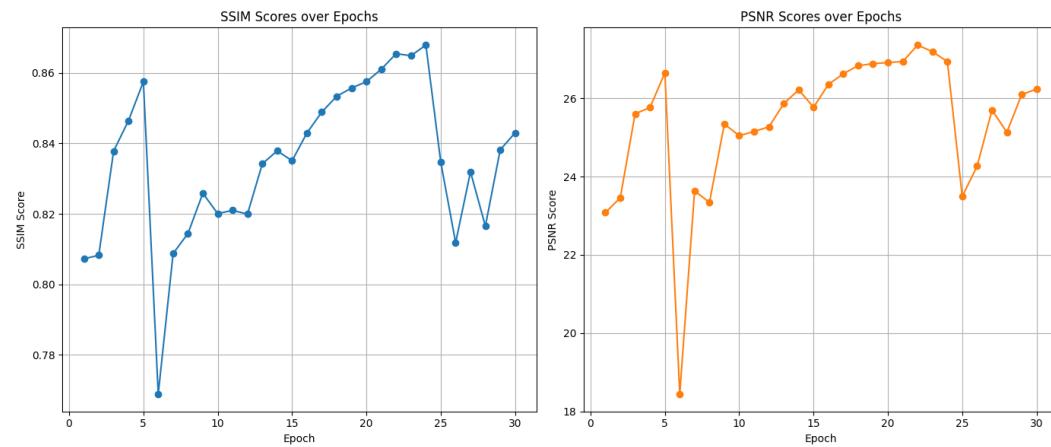


Figura 7.13: SSIM e PSNR sul dataset di riferimento.

CAPITOLO 8

Analisi dei Risultati e Conclusioni

8.1 Analisi

8.1.1 Analisi codec lossless

Partendo dalla fase di ri-esecuzione dei test condotti nelle precedenti sperimentazioni, si osserva che il comportamento dei codec analizzati è rimasto sostanzialmente invariato, fatta eccezione per i tempi di esecuzione. Questa variazione è probabilmente attribuibile all'utilizzo di macchine e ambienti di sviluppo differenti, sia per quanto riguarda le versioni dei codec che per le librerie e le implementazioni adottate.

Nel contesto della compressione di immagini, i codec lossless **MS-RLE** e **Dirac** non mostrano significative differenze rispetto a quelli precedentemente testati. In nessun caso, infatti, le loro prestazioni hanno superato quelle di **HEVC** e **AV1**, che già nelle sperimentazioni precedenti si erano distinti per rapporti di compressione superiori. Tuttavia, sotto il profilo dei tempi di esecuzione, MS-RLE e Dirac si sono rivelati più veloci in tutte le prove effettuate.

8.1.2 Analisi codec lossy

Per quanto riguarda i codec lossy, i risultati ottenuti sono stati eccellenti in termini di riduzione delle dimensioni, grazie al buon equilibrio tra qualità visiva ed efficienza di compressione. **HEVC** si è confermato il codec più efficace nel mantenere un'elevata qualità visiva durante i test, mentre **VP9** e **AV1** hanno mostrato un comportamento complessiva-

mente soddisfacente, offrendo una buona qualità percepita e dimensioni dei file compressi competitive.

8.1.3 Analisi PCA e SVD

Le tecniche di riduzione della dimensionalità hanno rappresentato un approccio alternativo per la compressione dei dati, consentendo di ridurre il rumore e preservare le componenti principali. I test condotti sui dataset hanno evidenziato una correlazione diretta tra il numero di componenti selezionate e le metriche SSIM e PSNR, che tendono mediamente a migliorare all'aumentare delle componenti principali considerate.

Tuttavia, trattandosi di modelli statistici generici, non ottimizzati per specifici dataset, hanno mostrato limitazioni significative nel trattamento di dataset di dimensioni ridotte, come **Dice**, **Mannequin** e **Fish**. In questi casi, si sono riscontrati valori di SSIM e PSNR inferiori rispetto a quelli ottenuti con dataset più ampi, evidenziando una maggiore difficoltà nel rappresentare accuratamente i dati compressi. Inoltre, l'uso di questi metodi genera immagini che possono presentare errori localizzati a livello di pixel, richiedendo in alcuni casi attività di post-processing per migliorare la qualità visiva.

La principale differenza tra PCA e SVD risiede nella complessità computazionale: **PCA** si è rivelata più veloce, mentre **SVD** ha garantito una rappresentazione più accurata nei dataset caratterizzati da una maggiore variabilità. Entrambi i metodi si sono dimostrati sensibili alla scelta del numero di componenti principali, evidenziando la necessità di trovare un equilibrio tra compressione e qualità delle immagini ricostruite.

L'uso della compressione entropica sui file serializzati con la libreria Zstandard ha permesso una riduzione aggiuntiva delle dimensioni dei dati, ottimizzando il consumo di spazio per la trasmissione tra le due parti. Questo passaggio non ha introdotto variazioni significative nei tempi di esecuzione, che restano pressoché invariati, con differenze nell'ordine dei centesimi di secondo.

8.1.4 Confronto tra approcci lossy e riduzione di dimensionalità

Il confronto tra i metodi lossy e le tecniche di riduzione della dimensionalità ha messo in luce vantaggi e limiti distinti. I codec lossy si sono dimostrati particolarmente efficaci nella compressione ad alte prestazioni, offrendo un ottimo compromesso tra qualità visiva e riduzione delle dimensioni dei file. Al contrario, PCA e SVD hanno mostrato una maggiore

flessibilità nell’adattare il livello di compressione ai requisiti specifici, ma si sono rivelati meno efficienti nella gestione di dataset di piccole dimensioni.

Dal punto di vista delle metriche PSNR e SSIM, i codec lossy hanno generalmente raggiunto valori superiori rispetto a quelli ottenuti con PCA e SVD. Tuttavia, grazie alla loro natura statistica, PCA e SVD si sono distinti per la capacità di preservare le caratteristiche principali dei dati compressi, pur a fronte di un costo computazionale più elevato. Questo rende i due approcci particolarmente adatti in contesti dove la conservazione delle informazioni essenziali è prioritaria rispetto alla pura efficienza.

Abbiamo testato i codec lossy sulla stessa porzione di dati utilizzata dalle tecniche PCA e SVD, garantendo così un confronto equo tra le due metodologie. Per le valutazioni, abbiamo considerato i dataset **ArtGallery (101 frame)** e **Dice (25 frame)**, al fine di analizzare il comportamento delle tecniche di riduzione della dimensionalità su dataset di dimensioni diverse. Questa scelta ci ha permesso di confermare le differenze e i pattern osservati tra dataset piccoli e grandi.

I modelli basati su PCA e SVD che hanno registrato valori più elevati nelle metriche SSIM e PSNR sono quelli che utilizzano **60 componenti** per il dataset ArtGallery e **15 componenti** per il dataset Dice, quindi sono stati selezionati per il confronto. Come mostrato nella tabella 8.1, i codec lossy garantiscono tempi di compressione inferiori rispetto alle tecniche di riduzione della dimensionalità, evidenziando un vantaggio in termini di efficienza computazionale.

Algoritmo	Tempo di compressione (s)	
	ArtGallery	Dice
HEVC	3.70	0.71
VP9	25.12	1.72
AV1	1521.62	48.88
PCA (60 componenti)	25.95	2.04
SVD (60 componenti)	33.78	3.88

Tabella 8.1: Confronto tempi di compressione

Per quanto riguarda i rapporti di compressione, i modelli PCA/SVD che ottengono i migliori risultati in termini di SSIM e PSNR mostrano prestazioni che potrebbero compensare i tempi di esecuzione leggermente più elevati. Come evidenziato nella tabella 8.2, questi modelli offrono un buon equilibrio tra qualità della ricostruzione e costo computazionale.

Rapporto di compressione		
Algoritmo	ArtGallery	Dice
HEVC	9.6032	2.0780
VP9	2.9097	1.3691
AV1	5.5385	2.1319
PCA (60 componenti)	1975.88	875.21
SVD (60 componenti)	1975.88	875.21

Tabella 8.2: Confronto rapporti di compressione

Gli indici di similarità favoriscono leggermente i codec lossy, ma le differenze numeriche sono minime, come evidenziato nella tabella 8.3.

Algoritmo	ArtGallery		Dice	
	SSIM (%)	PSNR (db)	SSIM (%)	PSNR (db)
HEVC	0.981690	45.618785	0.997236	49.496494
VP9	0.994111	51.419748	0.999568	59.064846
AV1	0.991833	49.760834	0.999338	56.354438
PCA (60 componenti)	0.975919	31.108428	0.875017	26.551664
SVD (60 componenti)	0.976207	31.134145	0.869879	26.458677

Tabella 8.3: Confronto indici di similarità

Nel confronto visivo, le immagini processate con PCA/SVD presentano lievi imperfezioni che richiedono interventi di post-processing, mentre i frame compressi e ricostruiti con i codec video risultano immediatamente pronti all'uso.



Figura 8.1: Frame originale



Figura 8.2: Frame decompresso con VP9

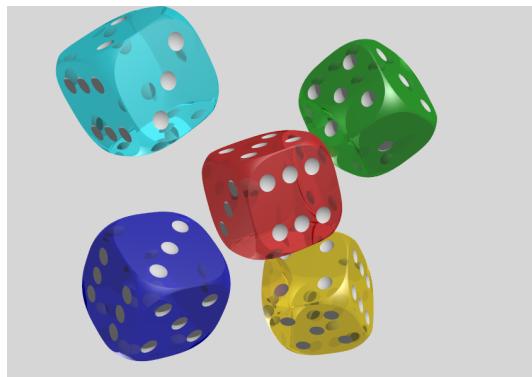


Figura 8.3: Frame decompresso con PCA



Figura 8.4: Frame decompresso con SVD

Figura 8.5: Confronto visivo su dataset ArtGallery

**Figura 8.6:** Frame originale**Figura 8.7:** Frame decompresso con VP9**Figura 8.8:** Frame decompresso con PCA**Figura 8.9:** Frame decompresso con SVD**Figura 8.10:** Confronto visivo su dataset Dice

8.1.5 Analisi autoencoder

I risultati della sperimentazione mostrano, anche in questo caso, una correlazione positiva tra la loss ottenuta dal modello e le metriche di similarità adottate, in alcune fasi del training. In particolare, tra le epoche 15 e 25 si osserva un incremento delle metriche SSIM e PSNR accompagnato da una riduzione della loss. Tuttavia, tale correlazione non si riscontra analizzando il rapporto tra i valori di compressione, la loss e le metriche di similarità. Infatti, al migliorare delle performance del modello in termini di loss e metriche SSIM/PSNR, il rapporto di compressione tende a diminuire.

I tempi di esecuzione variano significativamente a seconda dell’uso della codifica entropica sulle rappresentazioni latenti generate dalla fase di codifica. Tuttavia, questa tecnica si dimostra altamente efficiente nel ridurre lo spazio necessario per la trasmissione dei dati.

**Figura 8.11:** Frame originale**Figura 8.12:** Frame decompresso**Figura 8.13:** Confronto visivo con autoencoder

In generale, i risultati ottenuti risultano inferiori rispetto a quelli raggiunti con le tecniche precedentemente sperimentate. Questo è evidente sia dai valori di SSIM e PSNR, sia dalla qualità visiva delle immagini ricostruite, sia dai rapporti di compressione. Sebbene le metriche indichino una somiglianza media superiore all'80%, le immagini ricostruite non risultano qualitativamente allineate ai risultati delle altre tecniche. Inoltre, pur offrendo un discreto risparmio in termini di spazio grazie alla compressione, le immagini appaiono significativamente sfocate e poco precise rispetto agli originali.

8.2 Conclusioni

In conclusione, sulla base dei test effettuati e delle metodologie applicate, possiamo affermare che i codec tradizionali, sia nella versione lossless che lossy, si confermano nettamente superiori rispetto alle tecniche di riduzione di dimensionalità e all'approccio basato su reti neurali. Tuttavia, è importante evidenziare il potenziale delle tecniche di riduzione di dimensionalità, che dimostrano la capacità di ottenere immagini di qualità eccellente, con errori limitati a pochissimi pixel, pur raggiungendo rapporti di compressione molto elevati. Sebbene non siano state progettate specificamente per la compressione, queste tecniche si sono rivelate particolarmente efficaci durante la sperimentazione e possono rappresentare una valida alternativa in determinati contesti.

Per quanto riguarda l'approccio con autoencoder, i risultati ottenuti sono ancora lontani dai livelli raggiunti dai codec tradizionali sia in termini di metriche di similarità sia per la qualità visiva delle immagini ricostruite.

Come sviluppo futuro, sarà fondamentale lavorare ulteriormente sugli autoencoder per migliorarne le prestazioni, sia dal punto di vista della qualità visiva che delle metriche di valutazione, così da renderli una soluzione più competitiva rispetto alle tecniche tradizionali.

Bibliografia

- [1] J. Harfield, "What is light field photography and how does it work?" 2023. [Online]. Available: <https://www.makeuseof.com/what-is-light-field-photography/> (Citato a pagina 1)
- [2] "What is the light field?" [Online]. Available: <http://lightfield-forum.com/what-is-the-lightfield/> (Citato a pagina 3)
- [3] H. Eeckhaut, B. Schrauwen, M. Christiaens, and J. Campenhout, "Speeding up dirac's entropy coder." (Citato a pagina 12)
- [4] Wikipedia contributors, "Run-length encoding — Wikipedia, the free encyclopedia," 2025, [Online; accessed 13-February-2025]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Run-length_encoding&oldid=1273094395 (Citato a pagina 13)
- [5] A. M. A. Ibrahim and M. E. Mustafa, "Comparison between (rle and huffman) algorithms for lossless data compression," International journal of innovative technology and research, vol. 3, no. 1, pp. 1808–1812, 2015. (Citato a pagina 13)
- [6] D. Bank, N. Koenigstein, and R. Giryes, Autoencoders. Cham: Springer International Publishing, 2023, pp. 353–374. [Online]. Available: https://doi.org/10.1007/978-3-031-24628-9_16 (Citato a pagina 16)
- [7] MIT Media Lab Light Fields, "Dragon and bunnies." [Online]. Available: <https://web.media.mit.edu/~gordonw/SyntheticLightFields/> (Citato a pagina 23)
- [8] ——, "Messerschmitt." [Online]. Available: <https://web.media.mit.edu/~gordonw/SyntheticLightFields/> (Citato a pagina 23)

- [9] ——, “Dice.” [Online]. Available: <https://web.media.mit.edu/~gordonw/SyntheticLightFields/> (Citato a pagina 23)
- [10] ——, “Fish.” [Online]. Available: <https://web.media.mit.edu/~gordonw/SyntheticLightFields/> (Citato a pagina 23)
- [11] Max Planck Institut Informatik, “Car.” [Online]. Available: <https://lightfields.mpi-inf.mpg.de/Dataset.html> (Citato a pagina 23)
- [12] ——, “Room.” [Online]. Available: <https://lightfields.mpi-inf.mpg.de/Dataset.htm> (Citato a pagina 23)
- [13] ——, “Cobblestone.” [Online]. Available: <https://lightfields.mpi-inf.mpg.de/Dataset.html> (Citato a pagina 23)
- [14] ——, “Mannequin.” [Online]. Available: <https://lightfields.mpi-inf.mpg.de/Dataset.html> (Citato a pagina 24)
- [15] ——, “Blob.” [Online]. Available: <https://lightfields.mpi-inf.mpg.de/Dataset.html> (Citato a pagina 24)
- [16] Y. Collet and M. Kucherawy, “Zstandard Compression and the ‘application/zstd’ Media Type,” RFC 8878, Feb. 2021. [Online]. Available: <https://www.rfc-editor.org/info/rfc8878> (Citato a pagina 29)