POLITECNICO DI TORINO

Corso di Laurea Magistrale in
Ingegneria Informatica

Tesi di Laurea Magistrale

Supporto della compressione video AV1 in dispositivi mobili e analisi prestazionale

Relatore
Prof. Enrico Masala

Candidato
Federico Valerio Serraino
matricola: s261049

Luglio 2021

Sommario

Nella società odierna, la fruizione di contenuti multimediali, quali immagini e video, ricopre un ruolo chiave. Il *video streaming* rappresenta una fetta considerevole di tutto il traffico globale e ottenere dei miglioramenti nella compressione dei contenuti, anche di un paio di punti percentuali, implica enormi vantaggi per la rete (minor traffico), le organizzazioni coinvolte nel settore (maggiore risparmio economico per la memorizzazione e trasmissione dei contenuti) e gli utenti finali (migliore esperienza durante la fruizione dei servizi).

Scopo dell'elaborato è studiare le caratteristiche del nuovo formato di compressione video AV1 (AOMedia Video 1), sviluppato per essere *royalty-free* e all'avanguardia, capire qual è lo stato dell'arte delle performance delle più promettenti implementazioni attualmente disponibili e testarle in ambiente desktop e mobile, effettuando un confronto con le prestazioni ottenute dai formati AVC (Advanced Video Coding) e HEVC (High Efficiency Video Coding), gli standard più comunemente usati per la registrazione, compressione e distribuzione di contenuti video.

Considerando l'attuale scarsa implementazione del *video codec* AV1 nei dispositivi mobili, è stato sviluppato un apposito lettore multimediale per Android in grado di supportare il nuovo formato di compressione.

Inoltre, al fine di eseguire in maniera efficiente ed automatica le codifiche, le decodifiche e la raccolta dei dati riguardanti le prestazioni ottenute, sono stati sviluppati ed utilizzati degli appositi *shell script* e programmi C di supporto.

Indice

Elenco delle tabelle

Elenco delle figure

Abbreviazioni

AOM Alliance for Open Media

AV1 AOMedia Video 1

AVC Advanced Video Coding

CBR Constant Bit Rate

CD Compact Disc

CDEF Constrained Directional Enhancement Filter

CIF Common Intermediate Format

CODEC enCOder e DECoder

CPU Central Processing Unit

CTU Coding Tree Unit

DASH Dynamic Adaptive Streaming over HTTP

DCT Discrete Cosine Transform

DPCM Differential Pulse-Code Modulation

FPS Frames Per Second

FR Full Reference

HD High Definition

HDR High Dynamic Range

HEVC High Efficiency Video Coding

HVS Human Visual System

ISDN Integrated Services Digital Network

JPEG Joint Photographic Experts Group

MB MacroBlocco

MC-DPCM Motion-Compensated DPCM

ME Motion Estimation

MSE Mean Squared Error

MV Motion Vector

OSS Open Source Software

PSNR Peak Signal-to-Noise Ratio

PU Prediction Unit

QCIF Quarter Common Intermediate Format

QP Quantization Parameter

RAV1E Rust AV1 Encoder

RLE Run-Length Encoding

SAO Sample Adaptive Offset

SD Standard Definition

SoC System on Chip

SVM Support Vector Machine

SVoD Subscription Video on Demand

SVT-AV1 Scalable Video Technology for AV1

TU Transform Unit

UHD Ultra High Definition

VBR Variable Bit Rate

VMAF Video Multi-method Assessment Fusion

WCG Wide Color Gamut

Capitolo 1

Introduzione

1.1 Video trends

La fruizione di contenuti multimediali, quali immagini e video, ricopre un ruolo chiave all'interno della società odierna. Nel corso degli anni, il volume di video messi in rete è cresciuto in maniera esponenziale e per comprenderne la rilevanza è sufficiente prendere in considerazione le seguenti statistiche:

- Ogni secondo 1 milione di minuti di contenuti video attraversa la rete [1].
- Il 78% delle persone guarda video online ogni settimana, il 55% guarda video online ogni giorno [2].
- 6 persone su 10 preferiscono guardare video online piuttosto che la televisione [3].
- YouTube è il secondo sito web più visitato al mondo [4].
- Ogni giorno su YouTube vengono visualizzate più di 1 miliardo di ore di video [5].
- I contenuti video rappresentano quasi il 50% di tutto il traffico sulle reti mobili [6].
- È stato stimato che entro il 2022 i video online rappresenteranno oltre l'82% di tutto il traffico Internet, circa 15 volte in più rispetto al 2017 [1].

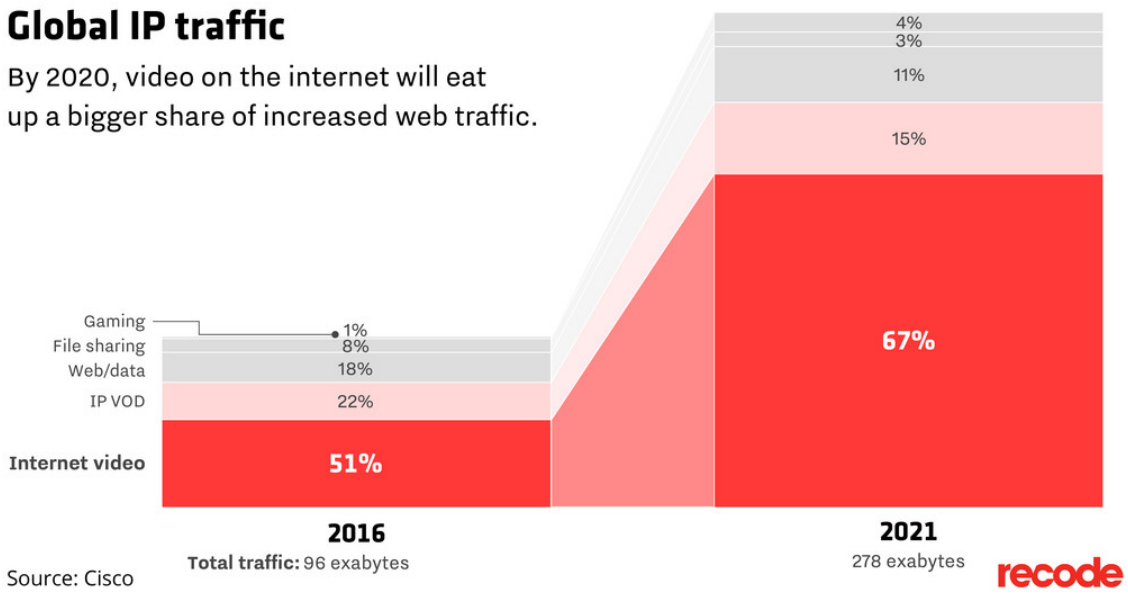


Figura 1.1: Traffico Internet Globale 2016 vs 2021

1.2 Evoluzione dei contenuti video e introduzione ai formati di compressione

Il termine **video** viene utilizzato per indicare una sequenza di immagini, chiamate tipicamente fotogrammi (*frames*), che si susseguono ad una velocità costante nel tempo, detta *frame-rate*. Grazie al fenomeno chiamato **persistenza della visione**, il cervello umano interpreta una sequenza di immagini che si succedono come un flusso in movimento.

Originariamente, i contenuti video venivano memorizzate come segnale analogico su un nastro magnetico. Nel periodo in cui sul mercato è arrivato il *compact disc* (CD), utile per memorizzare l'audio analogico in maniera digitale, è presto diventato possibile anche il salvataggio e la distribuzione di sequenze video in formato digitale.

Il passaggio al digitale ha introdotto numerosi vantaggi, tra i quali: una migliore qualità dei contenuti, una maggiore affidabilità nelle trasmissioni e una notevole flessibilità a livello applicativo. Tuttavia, in un primo periodo, il formato digitale non poteva competere con il formato

Fonte Figura 1.1 [7].

analogico, poiché il volume di dati generato dal processo di digitalizzazione risultava eccessivamente elevato e non compatibile con i sistemi di archiviazione e trasmissione.

Per tale ragione, a partire dalla fine degli anni '80, sono stati sviluppati i primi **formati di compressione video**. Con il passare degli anni, il *video streaming* ha ricoperto una fetta sempre più considerevole di tutto il traffico Internet e ottenere dei miglioramenti, anche di un paio di punti percentuali, nella compressione dei contenuti implica enormi vantaggi per:

- la **rete**: minor carico di traffico;
- le **organizzazioni coinvolte nel settore**: maggiore risparmio economico per la memorizzazione e trasmissione dei contenuti;
- gli **utenti finali**: migliore esperienza durante la fruizione dei servizi.

Al giorno d'oggi, i formati più comunemente usati per la registrazione, compressione e distribuzione di contenuti video sono **AVC** (Advanced Video Coding) e **HEVC** (High Efficiency Video Coding), due standard la cui implementazione è protetta da brevetti. Alla base dello sviluppo degli standard proprietari vi è un *business model* strutturato nel seguente modo: i titolari dei brevetti consentono l'uso dei loro prodotti in cambio del pagamento di ingenti *royalties*, le quali, a loro volta, vengono utilizzare per finanziare lo sviluppo di nuove tecnologie per la prossima generazione di standard.

Tuttavia, tale sistema ha iniziato a sgretolarsi quando, con l'introduzione di HEVC, sono sorte alcune problematiche relative al fatto che, anche a diversi anni dalla data di rilascio dello standard, numerose organizzazioni si contendono i diritti di licenza sui brevetti. Il risultato finale è che si è creata un'estrema incertezza riguardo i termini di licenza e quanto si debba effettivamente pagare per poter utilizzare le tecnologie brevettate HEVC.

Per far fronte ai problemi introdotti dai brevetti, diverse grandi compagnie hanno intrapreso lo sviluppo di nuovi formati di compressione video ***royalty-free***. I progressi di tali *video codec* stavano producendo risultati piuttosto promettenti, tuttavia gli sforzi individuali necessari per portare avanti il lavoro avevano iniziato a soffocarne lo sviluppo e una possibile adozione su larga scala.

Per tale ragione, nel 2015, è stata fondata l'Alliance for Open Media (AOM), un consorzio composto da diverse grandi aziende che hanno

deciso di unire le proprie risorse e i propri brevetti nel campo del *video coding* con l'intento di promuovere lo sviluppo e l'adozione di un nuovo formato di compressione congiunto, chiamato **AV1** (AOMedia Video 1), mirato per essere all'avanguardia e *royalty-free*.

1.3 Scopo e struttura della tesi

Scopo dell'elaborato è studiare le caratteristiche del nuovo formato AV1, capire qual è lo stato dell'arte delle performance delle più promettenti implementazioni attualmente disponibili e testarle in ambiente desktop e mobile, effettuando un confronto con le prestazioni ottenute dagli standard AVC e HEVC.

Considerando l'attuale scarsa implementazione del *video codec* AV1 nei dispositivi mobili, è stata sviluppata l'applicazione **pAV0NE**, un lettore multimediale per Android in grado di supportare il nuovo formato di compressione video.

Inoltre, al fine di eseguire in maniera efficiente ed automatica le codifiche, le decodifiche e la raccolta dei dati riguardanti le prestazioni ottenute, sono stati sviluppati ed utilizzati degli appositi **shell script** e **programmi C** di supporto.

La tesi si articola nel seguente modo:

- Nel primo capitolo viene proposta una panoramica sull'evoluzione dei contenuti video, facendo un focus sulle tendenze odierne, e vengono presentate le tematiche trattate nell'elaborato.

- Il secondo capitolo pone l'attenzione sulle principali strategie alla base della compressione dei contenuti video e sono presentati i più rilevanti formati standard di compressione.

- Il terzo capitolo riguarda l'analisi prestazionale dei più promettenti codificatori e decodificatori AV1 disponibili in ambiente desktop. I risultati sono confrontati con quelli ottenuti dagli standard AVC e HEVC.

- Nel quarto capitolo vengono presentate le caratteristiche dell'applicazione pAV0NE, sviluppata per offrire supporto della compressione video AV1 nei dispositivi mobili.

- Nel quinto capitolo viene eseguita l'analisi prestazionale dei più promettenti decodificatori AV1 in ambiente mobile. I risultati ottenuti sono confrontati con le performance dei decoder AVC e HEVC nativamente installati nei dispositivi.

- Infine, il sesto capitolo conclude l'elaborato, focalizzando l'attenzione sui risultati ottenuti, evidenziando i punti di forza e debolezza dei vari formati di compressione video e i possibili sviluppi futuri.

Capitolo 2

Codifica video

2.1 Rilevanza della codifica video

La rappresentazione di contenuti multimediali in formato digitale richiede una grande quantità di dati. Nonostante il continuo incremento delle capacità di memorizzazione e trasmissione, senza adottare opportune strategie di compressione, il processo di digitalizzazione finirebbe col generare un volume di dati che risulta essere eccessivamente elevato e non compatibile con la maggior parte dei sistemi di archiviazione e di trasmissione odierni. Ad esempio, una sequenza video **non compressa** a risoluzione *Full HD* (1080p) può arrivare ad occupare oltre 1.2 Gbps.

La codifica, o compressione, video è il processo responsabile della conversione di un video in formato digitale ad uno stream numerico di dimensioni più adatte per una sua memorizzazione e trasmissione.

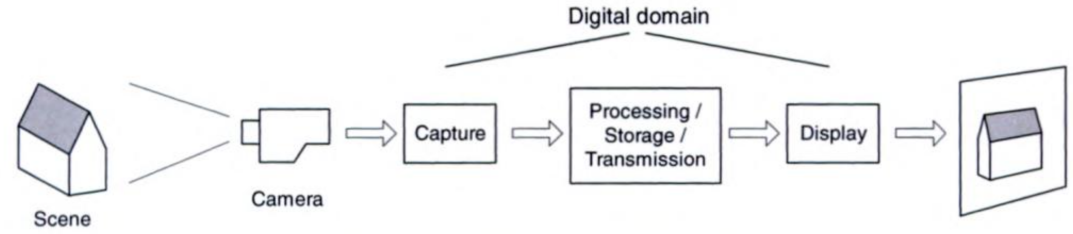


Figura 2.1: Esempio sistema di video digitalizzazione: cattura, elaborazione e visualizzazione

Il termine ***video codec*** è usato per indicare un dispositivo o software responsabile della compressione (***encoder***) e decompressione (***decoder***)

Fonte Figura 2.1 e riferimenti [8].

di una sequenza digitale. La coppia composta da codificatore e decodificatore è indicata con il termine ***CODEC*** (enCOder/DECoder).

D'ora in avanti, i termini **codifica** e **compressione** verranno utilizzati in modo ambivalente, per fare riferimento al concetto presentato.

2.2 Strategie di compressione

I processi che garantiscono considerevoli rapporti di compressione vengono indicati con il termine ***lossy***, in quanto, per ottenere una miglior compressione, accettando di perdere alcune informazioni, che, tuttavia, implicano un'inevitabile inferiore qualità del contenuto elaborato.

Scopo principale di un sistema di *video coding* è ridurre il più possibile il volume dei dati processati, cercando di mantenere un livello "*accettabile*" di qualità. Per raggiungere tale proposito, vengono adoperate delle apposite strategie di compressione.

2.2.1 Tecniche per la compressione di immagini

Una prima strategia implementabile per ridurre le dimensioni dei contenuti video è trattare ogni fotogramma che caratterizza una sequenza come un'immagine indipendente. In questo modo, diventa possibile utilizzare sul fotogramma le tecniche di compressione tipiche delle immagini, che, come nel caso del formato JPEG, garantiscono dei discreti rapporti di compressione (10 : 1), mantenendo una buona qualità del contenuto.

Tali tecniche sono caratterizzate da 4 step basilari [9] :

1) Sottocampionamento della Crominanza

Per codificare informazioni riguardanti i colori e la luminosità dei pixel che compongono un'immagine è necessario utilizzare un apposito modello rappresentativo. Il modello **RGB** (Red-Green-Blue) consente di rappresentare ogni pixel tramite tre numeri interi, usati per indicare le quantità di rosso, verde e blu (combinando questi tre colori si può ottenere ogni altro colore). Un'altra possibile rappresentazione è ottenuta utilizzando il modello **YUV**, in cui la componente Y è usata per indicare l'intensità luminosa (o luminanza) di ogni pixel, mentre le componenti U e V servono a rappresentare le informazioni riguardanti i colori (crominanza). Il modello YUV veniva originariamente usato

per rappresentare le immagini analogiche ed è ricavato direttamente dal modello RGB effettuando delle trasformazioni lineari:

- $Y = R \times r + G \times g + B \times b$ (con $r \sim 0.3; g \sim 0.6; b \sim 0.1$)
- $U = B - Y$
- $V = R - Y$

Per le immagini digitali, invece, viene solitamente impiegato il modello **Y′CbCr**, che è simile al modello YUV ma presenta alcuni opportuni offset nelle componenti. In tale modello, Y′ indica la luminanza, mentre le componenti CbCr sono usate per rappresentare la crominanza.

Considerando che il sistema visivo umano (HVS) è maggiormente sensibile alla luminanza (Y′), è possibile ridurre la quantità di informazioni necessarie per rappresentare le componenti cromatiche (CbCr) senza alterare significativamente la qualità percepita. Lo schema di sottocampionamento più frequentemente usato è **4:2:0**, in cui a 4 campioni di Y′ corrispondono solamente 1 campione di Cb ed 1 campione di Cr. Tale schema consente di ottenere una compressione pari al 50% rispetto al contenuto originale.

2) DCT (Discrete Cosine Transform)

Trasformata discreta del coseno, un'operazione matematica che consente di trasformare il segnale dal dominio dello spazio al dominio delle frequenze spaziali. Tale tipologia di trasformata permette di calcolare i coefficienti necessari per rappresentare il segnale mediante la somma pesata di funzioni cosinusoidali di base (formula 2.1).

$$B(k_1, k_2) = \sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} 4A(i,j) cos[\frac{\pi k_1}{2N_1}(2i+1)] cos[\frac{\pi k_2}{2N_2}(2j+1)] \quad (2.1)$$

In cui:

- N_1 e N_2 sono le dimensioni in pixel dell'immagine originale
- $A(i, j)$ è l'intensità dei pixel nella posizione i, j
- $B(k_1, k_2)$ è il coefficiente risultante

Nonostante la trasformata non causa perdita di informazioni, è un'operazione chiave e la motivazione alla base del suo impiego è che il HVS presenta una differente sensibilità alle diverse frequenze spaziali (maggiore è la quantità di dettagli più elevata risulta la frequenza spaziale).

Per eseguire la DCT è necessario, dapprima, suddividere l'immagine in blocchi da 8x8 pixel, e, a questo punto, la trasformata viene applicata a ciascuno dei blocchi di luminanza e crominanza. Il risultato finale dell'operazione è una matrice 8x8 di coefficienti (Figura 2.2), in cui il primo valore in alto a sinistra, chiamato coefficiente DC, è la componente continua del segnale, ottenuta mediante la media di tutti i blocchetti, mentre gli altri valori, definiti coefficienti AC, servono ad indicare le altre frequenze spaziali.

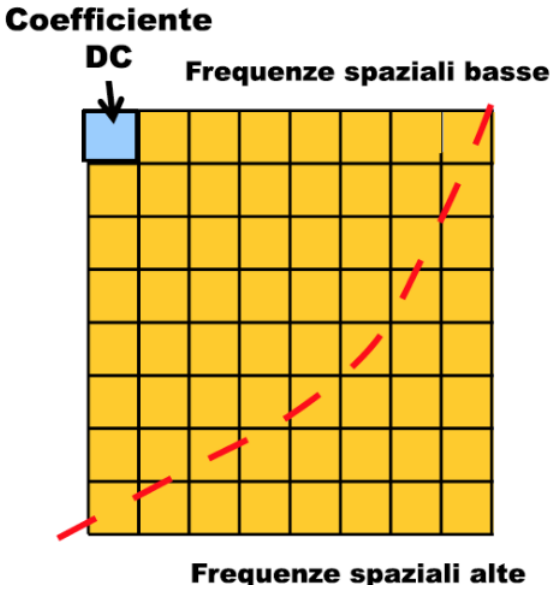


Figura 2.2: Matrice dei coefficienti

3) Quantizzazione

La quantizzazione è il processo tipicamente utilizzato per mappare valori appartenenti ad un intervallo continuo in altri valori distribuiti su un intervallo discreto e limitato. L'intervallo può essere suddiviso in livelli aventi ampiezza uguale (quantizzazione uniforme) oppure differente (quantizzazione non-uniforme).

Il concetto di quantizzazione può anche essere applicato ai coefficienti derivati dall'applicazione della DCT. In questo caso, con quantizzazione si intende l'approssimazione dei coefficienti ottenuta moltiplicando ogni valore per un fattore, appartenente al range [0,1], ed effettuando

un arrotondamento. Per consentire una miglior regolazione della compressione, tutti i coefficienti vengono anche moltiplicati per un ulteriore fattore, chiamato parametro di quantizzazione (Q_p).

Sfruttando il fatto che il HVS è maggiormente sensibile alle basse frequenze spaziali, per ottenere un'efficace compressione si esegue una quantizzazione più fine (moltiplicazione per fattori di quantizzazione prossimi a 1) per la componente DC e le componenti AC a basse frequenze, mentre viene effettuata una quantizzazione più grossolana per le componenti ad alte frequenze.

4) Codifica dei coefficienti

Dopo aver scartato i dettagli meno importanti grazie all'utilizzo della DCT e della quantizzazione, è possibile ridurre ulteriormente il numero di bit necessari per codificare i dati utilizzando una serie di tecniche aggiuntive:

- ***Zig-zag re-ordering***: viene utilizzato per disporre in maniera consecutiva gli zeri derivanti dal processo di quantizzazione della matrice dei coefficienti. Considerando che i coefficienti delle alte frequenze presentano una maggiore probabilità di assumere valori nulli, la matrice viene ordinata, tramite un riordinamento a *zig-zag* (Figura 2.3), e trasformata in un vettore in cui i coefficienti delle basse frequenze risultano posizionati prima dei coefficienti delle alte frequenze.

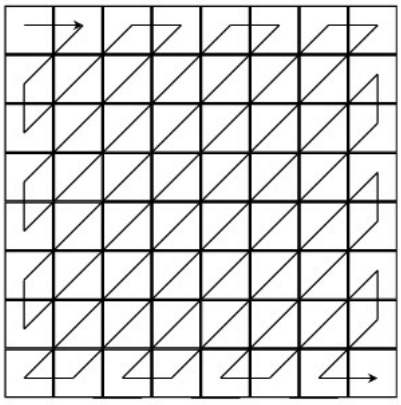


Figura 2.3: Zig-zag re-ordering

Fonte Figura 2.3 [10].

- ***Run-length encoding*** (RLE): il vettore risultate dallo *zig-zag reordering* contiene, solitamente, un elevato numero di zeri consecutivi. La RLE viene utilizzata per raggruppare i valori in coppie $(skip, value)$, in cui il campo *'skip'* serve ad indicare il numero di zeri consecutivi, mentre il campo *'value'* indica il prossimo valore diverso da zero. La fine del blocco è indicata dalla coppia (0,0).

- ***Codifica Entropica***: si basa sul principio di codificare una sequenza sfruttando le ricorrenze dei simboli che la caratterizzano. Tradizionalmente viene effettuata tramite la codifica di Huffman, una codifica binaria a lunghezza variabile rappresentata mediante un albero decisionale composto da zeri e uni. In tale codifica, il simbolo con maggiore probabilità è rappresentato utilizzando il codice più breve, mentre al simbolo meno probabile è assegnata la codifica più lunga.

2.2.2 Tecniche alla base dei sistemi di codifica video

Le tecniche usate per la codifica delle immagini ottengono dei rapporti di compressione pari a $10:1$, che, tuttavia, risultano ancora non essere sufficienti per garantire un'adeguata fruizione dei contenuti video.

Per tale ragione, al fine di effettuare un'efficiente compressione, si può sfruttare il fatto che sequenze consecutive di fotogrammi risultano, tipicamente, estremamente simili fra loro. Tali somiglianze, indicate con il termine di **ridondanze temporali**, possono essere codificate come delle piccole differenze numeriche tra un campione ed il successivo utilizzando una codifica differenziale **DPCM** (*differential pulse-code modulation*). Inoltre, nei video naturali spesso capita che alcuni oggetti o porzioni di un frame rimangono simili anche in frame successivi, ma presentano dei leggeri spostamenti. Per cui, anche questa considerazione può essere sfruttata per migliorare ulteriormente la codifica differenziale, adoperando delle apposite tecniche di **compensazione del moto**.

Le due precedenti considerazioni, che sono alla base delle tecniche di compressione di tutti i principali sistemi di codifica video, vengono unite ed utilizzate per effettuare una codifica **MC-DPCM** (*Motion-Compensated* DPCM), che consente, nel caso in cui gli elementi di un frame risultano simili ma in una differente posizione rispetto a elementi precedentemente codificati, di codificare solamente la quantità di moto ed eventuali piccole differenze nell'aspetto degli elementi [9].

Durante l'esecuzione della codifica MC-DPCM, ogni frame è suddiviso in dei blocchi, tipicamente di dimensioni 16x16 pixel. Tali blocchi vengono chiamati **macroblocchi** (MB) e sono processati in maniera indipendente.

Per ogni MB presente all'interno di un frame n, viene eseguita la ricerca di un'area simile nel frame $n-1$. Questa operazione è nominata ***motion estimation*** (ME) e la ricerca può essere effettuata in tutto il frame $n-1$ (*full-scan*) oppure può essere limitata ai MB adiacenti alla posizione di partenza, che hanno una maggiore probabilità di presentare delle somiglianze.

A questo punto, si effettua la codifica dei valori che rappresentano il moto del MB del frame n rispetto all'area di riferimento scelta nel frame $n-1$. Il vettore utile ad indicare il moto del MB è chiamato ***motion vector*** (MV). Infine, nel caso in cui il MB e l'area selezionata presentano delle ulteriori discrepanze nell'aspetto, viene codificata anche la loro differenza, che prende il nome di ***prediction residual***.

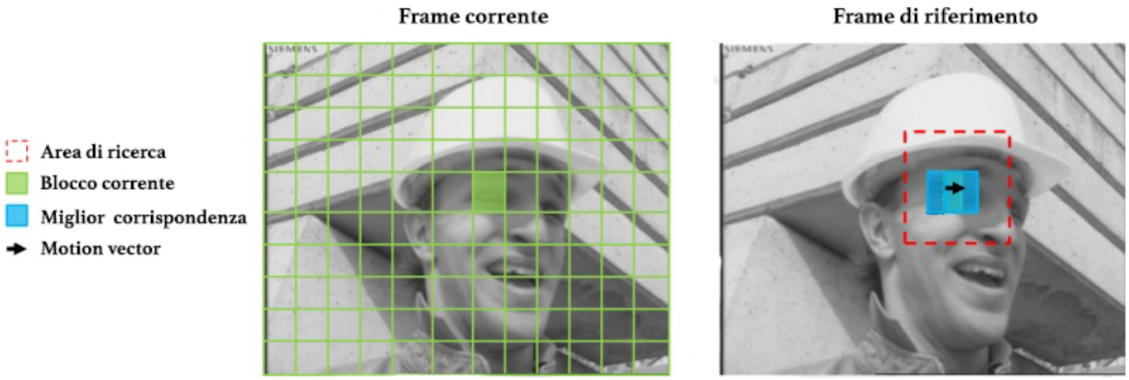


Figura 2.4: Esempio di Motion Estimation

Nel caso in cui il codificatore ha a disposizione anche dei fotogrammi futuri, è possibile effettuare delle predizioni volte a codificare eventuali nuovi oggetti che entrano nella scena. In questo caso, avendo a disposizione sia frame passati sia frame futuri, vengono eseguite delle **predizioni bi-direzionali**, che consentono di ridurre ulteriormente il *prediction residual* garantendo una migliore predizione.

Fonte Figura 2.4 [11].

2.3 Caratteristiche codifica video

Sulla base delle tecniche presentate nella sezione 2.2, la codifica video può essere effettuata nel seguente modo:

- per il primo fotogramma: codifica indipendente, sfruttando le tecniche tipiche della compressione delle immagini (suddivisione in blocchi, sottocampionamento della crominanza, DCT, quantizzazione, *zig-zag re-ordering*, RLE e codifica entropica);
- per i fotogrammi successivi: utilizzo dell'algoritmo MC-DPCM, con *motion estimation*, ricerca dei *motion vectors* e calcolo del *prediction residual*. Se nessuna delle aree prese in considerazione è reputata abbastanza soddisfacente, allora viene eseguita una codifica indipendente.

2.3.1 Tipologie di frame

I fotogrammi che caratterizzano una sequenza video sono solitamente raggruppati in tre categorie [12] :

- **I** (Intra): in nessuno dei macroblocchi che compongono il frame sono utilizzate strategie di predizione.
- **P** (Predictive): nella codifica del frame sono usate tecniche di predizione, ma solamente rispetto a riferimenti passati. I macroblocchi del frame possono essere di tipo I e P.
- **B** (Bi-predictive): vengono utilizzate tutte le tipologie di predizione (riferimenti nel passato e futuro) e i macroblocchi possono essere di tipo I, P e B.

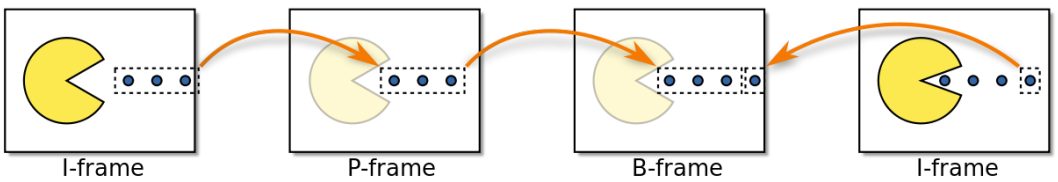


Figura 2.5: Esempi frame di tipo I, P e B

Fonte Figura 2.5 [13].

Nel caso in cui si utilizzano i frame di tipo B, è necessario codificare le aree di riferimento ricavate dai fotogrammi futuri prima del frame B corrispondente. Si viene così a generare un ordine di codifica dei frame (***codec order***) che risulta essere differente rispetto all'ordine in cui essi vengono naturalmente mostrati (***display order***). Il codificatore dovrà, quindi, inviare al decodificatore i frame nell'ordine in cui essi dovranno essere correttamente processati.

2.3.2 Struttura codificatore

Quando si fa uso della codifica DPCM, la predizione deve essere effettuata rispetto ai dati che vengono processati dal decodificatore, è, quindi, responsabilità del codificatore eseguire le predizioni basandosi sulla versione ricostruita del frame [12].

Nella Figura 2.6 viene rappresentato un esempio di diagramma a blocchi del codificatore.

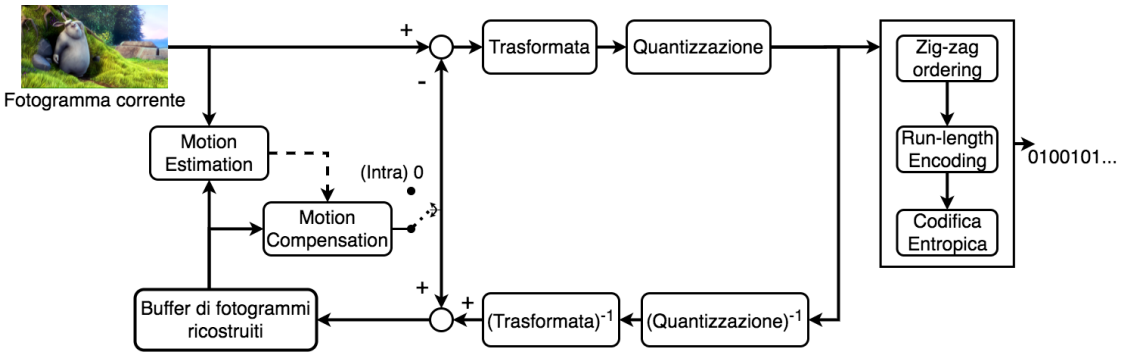


Figura 2.6: Diagramma a blocchi codificatore

La *motion estimation* è un procedimento computazionalmente pesante e rappresenta la parte significativa della complessità di una codifica video. Complessivamente, le operazioni eseguite dal codificatore sono molto più onerose rispetto a quelle eseguite dal decodificatore, per tale ragione, la codifica video è un procedimento di tipo **asimmetrico**.

La codifica video risulta, quindi, essere maggiormente adatta ad uno scenario di tipo *broadcast*, in cui la codifica è effettuata solamente una volta (dall'emittente), e la decodifica viene eseguita più volte (da ogni utente).

2.3.3 Struttura decodificatore

La decodifica è procedimento più semplice rispetto alla codifica e le principali operazioni svolte dal decoder, ovvero quelle che si occupano della ricostruzione dei fotogrammi, vengono effettuate anche all'interno del codificatore.

Nella Figura 2.7 è rappresentato un esempio di diagramma a blocchi del decodificatore.

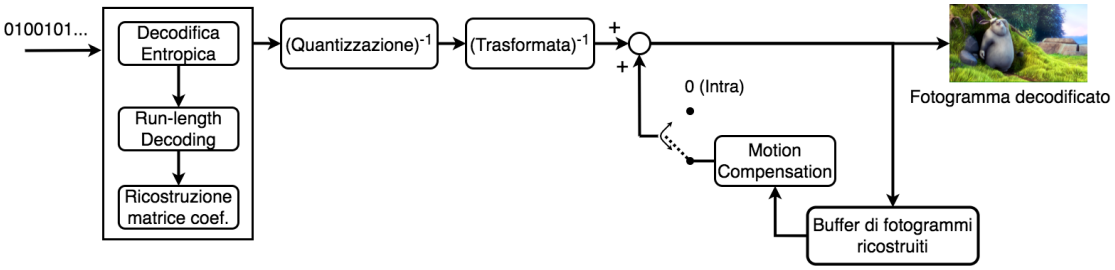


Figura 2.7: Diagramma a blocchi decodificatore

Gli standard di codifica garantiscono che qualsiasi decoder conforme ricostruisce i frame nella medesima maniera.

2.3.4 Codifica multi-pass

Nel caso in cui è sufficiente effettuare una codifica in modalità '*offline*', come per memorizzare dei contenuti localmente o trasmettere dei *video streaming on-demand*, il tempo necessario per l'esecuzione non è molto rilevante e diventa possibile adottare delle strategie di codifica ***multi-pass***, in cui:

- in un primo passaggio: si effettua lo studio delle caratteristiche del contenuto video da codificare;
- in un secondo passaggio: viene scelto, sulla base delle informazioni precedentemente acquisite, il modo in cui è maggiormente conveniente allocare i bit che si hanno a disposizione;
- da ultimo: si esegue la codifica con i parametri scelti.

2.4 Formati standard di codifica video

2.4.1 H.261

È il primo standard di codifica video appartenente alla famiglia ***H.26x***. Progettato dal Video Coding Experts Group (VCEG) dell'ITU-T, è stato rilasciato nel 1988 ed è stato sviluppato per essere utilizzato nelle trasmissioni ISDN, in cui i dati sono multipli di 64 kbit/s.

Lo standard supporta due formati immagine: CIF (luminanza 352x288 e crominanza 176x144) e QCIF (luminanza 176x144 e crominanza 88x72). Entrambi i formati usano uno schema di sottocampionamento 4:2:0.

H.261 fu il primo codificatore video ad utilizzare come unità di elaborazione i macroblocchi e le principali caratteristiche impiegate dallo standard sono [14] :

- **Predizioni**: basate sulle differenze esistenti tra il frame corrente ed il precedente. Vengono impiegati frame di tipo I e P.
- **Trasformata**: DCT, con suddivisione in blocchi da 8x8, usata per ridurre le ridondanze spaziali.
- **Quantizzazione**: consente di ottenere una miglior compressione rappresentato i coefficienti della DCT con precisione adeguata.
- **Codifica Entropica**: viene utilizzata la codifica di Huffman (*non-lossy*).

H.261 è stato il primo standard utile in termini pratici e rappresenta la pietra miliare nel campo dello sviluppo della codifica video. Tutti i successivi formati di compressione video si basano sul progetto di H.261.

2.4.2 H.264 / AVC

Advanced Video Coding, noto anche come H.264 e MPEG-4 Part 10, è uno standard di compressione video sviluppato dal VCEG ITU-T in collaborazione con ISO/IEC JTC1 Moving Picture Experts Group (MPEG). Scopo del progetto è creare uno standard in grado di fornire una buona qualità video a *rate* di codifica notevolmente inferiori rispetto i precedenti standard. La prima versione è stata rilasciata nel 2003 e nelle successive edizioni sono state aggiunte alcune estensioni alle sue

funzionalità, come il supporto per risoluzioni fino 4K (4096×2160) a 60 frame al secondo (fps) [15].

AVC è protetto da vari brevetti, la cui licenza è amministrata dall'azienda MPEG LA, e l'uso commerciale delle tecnologie brevettate AVC necessita il pagamento di *royalties*.

Lo standard è basato su una codifica orientata alla suddivisione in blocchi e le principali innovazioni implementate sono:

- **Trasformata**: utilizza una differente tipologia di trasformata, di dimensioni 4x4 e simile ad una DCT 4x4, che consente di lavorare con i numeri interi più semplici da gestire.

- **Predizioni**: sono state estese le modalità di predizione. Ogni MB utilizza un maggior numero di MVs (fino a 16). Si effettuano predizioni anche all'interno dei frame di tipo I, utilizzando i macroblocchi precedentemente codificati (***intra-prediction***). Le aree di riferimento non sono più limitate all'ultimo frame decodificato. Anche i frame di tipo B possono essere usati come riferimento. I riferimenti dei frame B possono essere tutti nel passato o nel futuro.

- **VCL e NAL**: Video Coding Layer e Network Abstraction Layer, sono livelli introdotti nello standard per ottenere una separazione tra l'esecuzione della codifica e il trasferimento dei dati.

- **Parameter Sets**: consentono di codificare in maniera separata importanti dati utili in fase di decodifica. Fanno parte degli elementi del NAL e possono essere trasmessi anche *out-of-band*.

- **Deblocking Filter**: filtro che permette di ridurre la *"bloccosità"* degli artefatti causata dal processamento dell'immagine come una composizione di macroblocchi indipendenti.

- **Codifica Entropica Adattativa**: la probabilità dei simboli da codificare cambia in base alle ricorrenze dei simboli precedentemente codificati, garantendo una migliore stima.

Ad oggi, nonostante lo standard abbia più di 15 anni, è il formato più comunemente utilizzato per la registrazione, compressione e distribuzione di contenuti video.

2.4.3 VP8

Formato di compressione video sviluppato da On2-Technologies nel 2008. È stato successivamente rilasciato in maniera aperta e *royalty-free* quando la compagnia è stata acquistata da Google [16].

VP8 presenta caratteristiche simili ad AVC:

- utilizza DCT su blocchi 4x4;
- i macroblocchi posso avere dimensioni da 4x4 a 16x16 pixel;
- per le predizioni vengono usati fino a un massimo di 3 frame di riferimento;
- supporta solamente sequenze video con schema di sottocampionamento della crominanza 4:2:0 e 8-bit per rappresentare i colori dei pixel.

Tuttavia, VP8 risulta essere non molto adeguato per codificare contenuti ad alte risoluzioni (HD) e spesso ottiene delle prestazioni inferiori rispetto AVC.

L'importanza dello standard risiede nel fatto che è stato rilasciato in maniera aperta e senza la necessità di dover pagare *royalties* o tasse di licenza per essere implementato. Tale evento si verificò dopo che la *Free Software Foundation* (FSF) pubblicò una lettera aperta [17] a Google con lo scopo di invitarli a rilasciare gratuitamente VP8 e di spingerne l'utilizzo nei loro prodotti, come YouTube, così da poter gradualmente sostituire gli standard proprietari.

2.4.4 H.265 / HEVC

High Efficiency Video Coding, noto anche come H.265 e MPEG-H Part 2, è uno standard di compressione video rilasciato nel 2013 con il proposito di diventare il successore di AVC. Rispetto a quest'ultimo, garantisce un guadagno nella compressione dei contenuti che varia dal 25% al 50%, mantenendo lo stesso livello di qualità percepita [18].

Supporta risoluzioni fino 8K UHD (8192×4320) a 120 fps e le innovazioni chiave introdotte dallo standard sono:

- **Disaccoppiamento** fra i meccanismi di predizione, trasformata e divisione dell'immagine in blocchi. Tale separazione consente l'introduzione di tre unità indipendenti (CTU,PU,TU).

- **Unità di codifica**: la dimensione dei macroblocchi non è più fissa (16x16) ma può variare da 8x8 fino a 64x64 pixel, garantendo un miglior adattamento alle aree da codificare.
- **Trasformata**: fa uso di più tipologie di trasformate, le cui dimensioni variano da 4x4 a 16x16, con lo scopo di ottenere una migliore precisione dei coefficienti.
- **Filtro SAO** (*Sample Adaptive Offset*): una nuova tipologia di filtro che consente di rimuovere possibili artefatti nei bordi, introdotti dalla trasformata, e ridurre la distorsione.
- **Processamento in parallelo**: offre strumenti per processare il contenuto in parallelo, così da poter sfruttare al meglio i sistemi multi-core.

L'uso commerciale delle tecnologie brevettate HEVC richiede il pagamento di ingenti *royalties*. Rispetto ad AVC, la suddivisione delle licenze risulta essere abbastanza complessa. Dalla Figura 2.8 è possibile notare che, di circa 45 titolari di brevetti, solamente 2/3 hanno aderito a uno dei 3 patent pool (grandi detentori di brevetti).

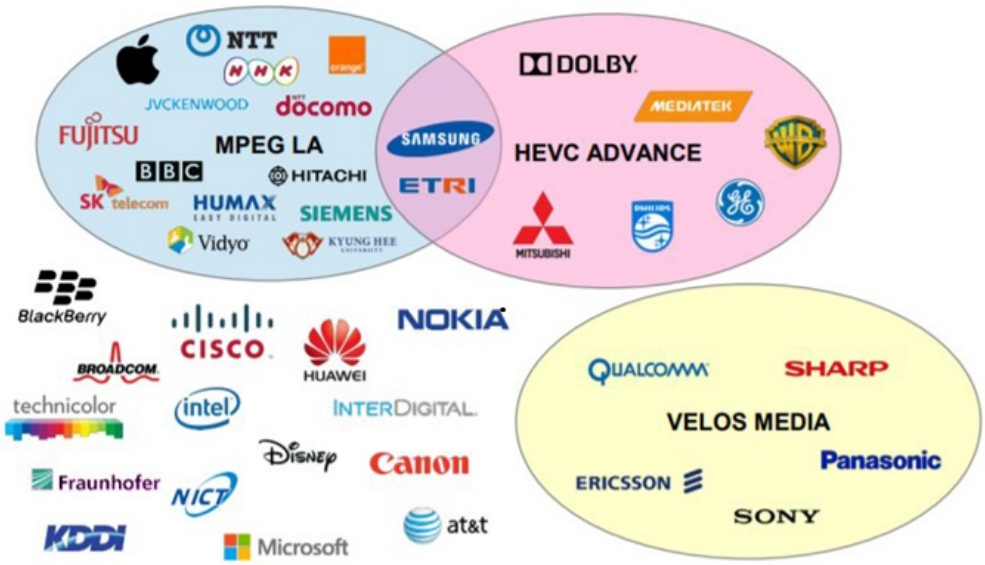


Figura 2.8: Detentori brevetti HEVC

Risultato finale di tale organizzazione è che si è creata un'estrema incertezza riguardo i termini di licenza e quanto si debba effettivamente pagare per poter utilizzare le tecnologie brevettate HEVC.

Fonte Figura 2.8 e riferimenti [19].

2.4.5 AV1

Per far fronte ai problemi introdotti dai termini di licenza relativi l'utilizzo delle tecnologie brevettate HEVC, diverse gradi aziende hanno intrapreso lo sviluppo di nuovi *video codec* ***royalty-free***. Google, successivamente a VP8, ha iniziato lo sviluppo di VP9 e VP10, Cisco ha cominciato ad implementare Thor nei propri prodotti di videoconferenza e Xiph, in collaborazione con Mozilla, si è occupato dello sviluppo di Daala [20].

Tali *video codec* stavano producendo dei risultati piuttosto promettenti, ma gli sforzi individuali necessari per portare avanti il lavoro avevano iniziato a soffocarne lo sviluppo e una possibile adozione su larga scala. Per tale ragione, nel 2015 sette grandi aziende (Amazon, Cisco, Google, Intel, Microsoft, Mozilla e Netflix) hanno deciso di unire i propri brevetti nel campo del *video coding* e fondare l'*Alliance for Open Media* (AOM), un consorzio nato con l'intento di promuovere lo sviluppo e l'adozione di un nuovo *video codec* congiunto [21].

Il progetto ha dato origine a AOMedia Video 1 (**AV1**), un formato di compressione video sviluppato per essere:

- **Royalty-free**: aperto, interoperabile e senza la necessità di dover pagare royalties o tasse di licenza per essere implementato.
- **Scalabile**: in grado funzionare su ogni dispositivo moderno a qualunque banda.
- **Flessibile**: implementabile in contenuti commerciali e non.
- **Ottimizzato**: per poter al meglio operare in servizi di *video streaming* e relative applicazioni.
- E in grado di offrire:
  - **una miglior compressione**: utilizzando, a parità di qualità percepita, un minor quantitativo di bit rispetto ai precedenti standard.
  - **video real-time ad alta qualità**: supportando caratteristiche come 4K UHD, HDR e WCG nei video real-time.

La prima versione di AV1 è stata rilasciata nel 2018 e le principali innovazioni introdotte sono [22] :

- **Unità di codifica**: i macroblocchi hanno dimensioni da 4x4 fino a 128x128 pixel e posso avere anche una forma rettangolare (ad esempio 4x16). È, inoltre, possibile partizionare ogni macroblocco in 10 differenti modalità.

- **Predizioni**: vengono impiegati più modelli per effettuare le predizioni.

  - ***Intra-frame***: usa un meccanismo di filtraggio ricorsivo che consente di basare le previsioni dei prossimi pixel non direttamente dal contenuto originale ma su ciò che è stato appena predetto. Per effettuare le predizioni delle componenti della crominanza vengono usate le informazioni già codificate della luminanza, tale meccanismo fornisce dei buoni risultati in scene con movimenti veloci.
  - ***Inter-frame***: vengono usati fino a 7 frame di riferimento. È possibile codificare un moto globale (*Global Motion*) di un intero frame (in termini di traslazione, rotazione e ridimensionamento). Viene eseguita una interpolazione fra il MV del blocco corrente e quelli dei blocchi precedentemente codificati. Facendo uso della *Wrapped-Motion*, è quasi come se ogni pixel avesse un proprio MV.

- **Trasformata**: fa uso di 4 differenti tipologie di trasformate, che, combinando tra loro righe e colonne dei frame, possono diventare 16. Inoltre, la dimensione delle trasformate è stata ampliata e può variare da 4x4 a 64x64.

- **Quantizzazione**: fa uso di nuove matrici ottimizzate per la quantizzazione. I parametri di quantizzazione assegnati ad ogni frame dispongono, a loro volta, di parametri individuali per i piani della crominanza e fanno uso della predizione spaziale.

- **Loop Filtering**: viene utilizzato un sistema di filtraggio ricorsivo composto da:

  - ***Deblocking filter***: per ridurre la bloccosità degli artefatti.
  - ***CDEF*** (*Constrained Directional Enhancement Filter*): consente di compensare gli artefatti ad anello che si generano nei bordi degli elementi e di mantenere i dettagli dell'immagine più naturali.
  - ***Loop restoration***: utile per rimuovere eventuali sfocature causate dall'elaborazione dei macroblocchi.

- **Codifica Entropica**: viene impiegata l'*Adaptive Multi-Symbol Arithmetic Coding*, una codifica che supporta fino a 16 valori per rappresentare ogni simbolo. Tale strategia permette un più veloce adattamento per i primi simboli e di aggiornare le probabilità dopo la codifica di ogni simbolo (non frame).

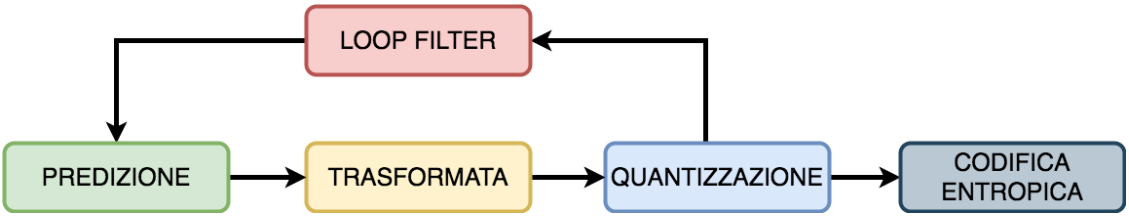


Figura 2.9: Ciclo di codifica AV1

La popolarità del formato AV1 è in crescente aumento. Numerose aziende hanno deciso di unirsi all'*Alliance for Open Media* (Figura 2.10) e di integrare il nuovo formato di compressione video nei propri servizi.

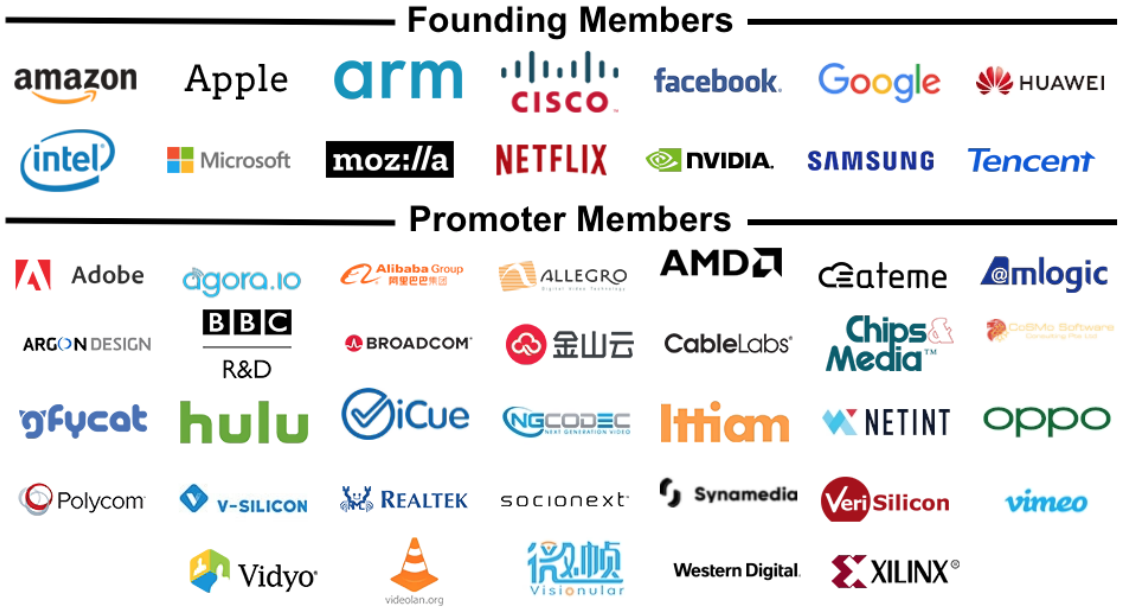


Figura 2.10: Membri Alliance for Open Media

Google ha già implementato AV1 in numerosi servizi, come ad esempio YouTube. In una conferenza online tenuta nel 2020 (Figura 2.11) è stato dichiarato che a breve sarà integrato su tutta la gamma. Ad oggi, risulta essere integrato anche in numerosi smartphone e Android-TV.

Fonte Figura 2.10 [25].

Oltre a ciò, **Facebook** ha effettuato dei primi test prestazionali nel 2018 [23] e **Netflix**, da febbraio 2020, ha abbandonato l'utilizzo di VP9 in favore di AV1, dichiarando di voler espandere l'impiego del nuovo *video codec* a più casi d'uso. Al momento sta anche lavorando con dei partner di dispositivi e chipset per creare del supporto hardware [24].

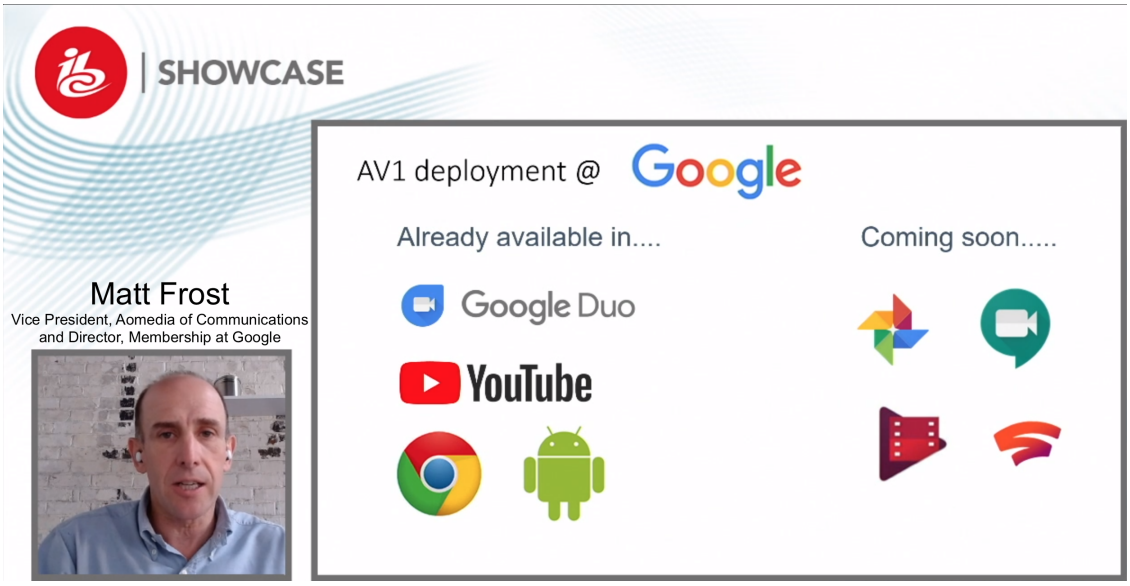


Figura 2.11: AV1 nei servizi Google, Matt Frost, conferenza del 2020

Fonte Figura 2.11 e riferimenti [26].

Capitolo 3

Analisi prestazionale in ambiente desktop

Nel capitolo sono valutate le prestazioni dei più promettenti codificatori e decodificatori AV1 esistenti in ambiente desktop. I risultati ottenuti sono confrontati con le performance degli standard di compressione AVC e HEVC.

3.1 Parametri presi in esame

3.1.1 Qualità oggettiva

Quando si utilizzano codifiche video di tipo ***lossy***, è necessario valutare la qualità percepita dopo la compressione dei contenuti. Per far ciò è possibile adoperare due differenti approcci:

- eseguire dei test **soggettivi**, svolti con l'ausilio di utenti;
- utilizzare dei metodi **oggettivi**, ovvero algoritmi che cercano di approssimare l'opinione degli utenti.

I test soggettivi forniscono le migliori indicazioni, ma richiedono molto tempo e sono abbastanza dispendiosi. Per tale ragione, per effettuare le valutazioni qualitative sono stati impiegati dei metodi oggettivi, che risultano di facile utilizzo, poco dispendiosi e possono essere adoperati direttamente all'interno di algoritmi volti ad ottimizzare le prestazioni dei codificatori.

Gli algoritmi utilizzati alla base delle analisi sono 2 metodi oggettivi classificati come ***full reference*** (FR), poiché necessitano del contenuto originale per poter effettuare una valutazione qualitativa.

1) PSNR

Peak Signal-to-Noise Ratio, storicamente è il più rilevante tra i metodi oggettivi, serve ad indicare il degrado di un'immagine o di una sequenza video tramite il rapporto fra la massima potenza del segnale e la potenza del rumore, che influisce negativamente sulla rappresentazione. Il valore del PSNR (formula 3.1) è basato sullo scarto quadratico medio (MSE), che è ricavato dalla differenza pixel-per-pixel rispetto al fotogramma di riferimento (formula 3.2) [9].

$$PSNR_{(dB)} = 10 \log \frac{(2^n - 1)^2}{MSE} \tag{3.1}$$

$$MSE = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [d(x,y) - i(x,y)]^2 \tag{3.2}$$

In cui:

- $2^n - 1$ è il massimo valore assumibile dai pixel dell'immagine rappresentata usando n bit.
- M e N sono la larghezza e l'altezza in pixel di un singolo frame.
- $i(x,y)$ e $d(x,y)$ sono il frame originale e il frame compresso.

Dal momento che molti segnali presentano una gamma piuttosto ampia, il valore del PSNR viene solitamente espresso in termini di scala logaritmica in decibel (dB). In linea di massima, i valori variano da 20 a 50, e, tipicamente, più è elevato il PSNR maggiore sarà la somiglianza con il fotogramma originale.

2) VMAF

Video Multi-method Assessment Fusion, attualmente rappresenta lo stato dell'arte per quanto riguarda le metodologie di calcolo della qualità video [27]. Sviluppato e rilasciato come open source nel 2016 da Netflix, VMAF è un algoritmo di valutazione percettiva e, così come suggerisce il nome, trae beneficio dall'utilizzo congiunto di molteplici metriche, tra le quali spiccano [28] :

- VIF (*Visual Information Fidelity*): utile per monitorare la perdita di fedeltà dell'informazione.
- DLM (*Detail Loss Metric*): usata per valutare la perdita di dettagli e l'eventuale introduzione di alterazioni che possono distrarre l'attenzione dell'osservatore.
- MCPD (*Mean Co-Located Pixel Difference*): per misurare la differenza temporale della componente luminosa nei vari fotogrammi.

Le metriche sopraccitate vengono combinate utilizzando una regressione *SVM-based* (*Support Vector Machine*) per fornire una valutazione su una scala da 0 a 100 per ogni frame analizzato. Il valore 100 indica una qualità identica rispetto al fotogramma originale, 70 viene interpretato come una valutazione discreta, mentre valori pari o inferiori a 20 rappresentano una qualità pessima [29].

I punteggi di tutti i frame sono, infine, sommati e utilizzati per calcolare la media che determina la valutazione complessiva del contenuto.

3.1.2 Velocità di codifica e decodifica

Oltre ad ottenere degli ottimi rapporti di compressione, mantenendo una buona qualità percepita, anche la complessità computazionale e il tempo necessario per eseguire le codifiche e le decodifiche hanno una sostanziale rilevanza. Per tale ragione, nell'analisi prestazionale sono stati monitorati anche:

- *User time*: lasso di tempo che la CPU ha trascorso in user mode.
- *System time*: lasso di tempo che la CPU ha trascorso in kernel mode.
- *Wall clock time*: tempo effettivamente trascorso dall'inizio alla fine dell'esecuzione.

3.2 Panoramica dei più promettenti video codec AV1

Nella Tabella 3.1 sono elencati i più emergenti *video codec* AV1 open source (OSS) e commerciali.

Nome	Enc	Dec	OSS	Descrizione
AOM [30]	✓	✓	✓	Sviluppato direttamente dall'Alliance for Open Media, è stato introdotto come il *video codec* AV1 di riferimento. Ha consentito di poter apprezzare le potenzialità offerte dal formato.
SVT-AV1 [31]	✓	✓	✓	Codec creato da Intel in collaborazione con Netflix. È stato progettato per offrire un buon compromesso tra qualità e velocità di codifica. Pensato per essere scalabile e implementabile in un'ampia gamma di applicazioni di codifica e decodifica video.
RAV1E [32]	✓		✓	Rust AV1 Encoder, sviluppato dal team di Mozilla e Xiph.org. È stato progettato per ricoprire diversi casi d'uso ed essere preferito ad AOM, l'encoder di riferimento, qualora quest'ultimo risulti essere eccessivamente lento.
Aurora 1 [33]	✓			Codificatore prodotto da Visionular. È presentato come il più avanzato e performante encoder AV1 presente sul mercato, come riportato nel MSU Codec Study del 2019 [34].
EVE-AV1 [35]	✓			Sviluppato da Two Orioles, è presentato come un codificatore in grado di fornire un livello di qualità per il *video streaming on-demand* senza precedenti, garantendo una riduzione del 20% nel *rate* di codifica rispetto gli altri encoder AV1.

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

Cisco AV1 [36]	✓			Progettato per essere flessibile ed adatto ad operare in CPU solite lavorare con codificatori AVC, ottenendo miglioramenti qualitativi ad un minore *rate* di codifica.
DAV1D [37]		✓	✓	Sviluppato dalla organizzazione non-profit VideoLAN. L'obiettivo del progetto è fornire un decoder cross-platform volto al raggiungimento della massima velocità possibile, così da poter ovviare alla temporanea mancanza di supporto hardware. Attualmente è il decodificatore AV1 maggiormente implementato.
GAV1 [38]		✓	✓	Decoder AV1 progettato da Google per operare in maniera ottimale sui dispositivi Android *arm-powered*.
AV1 Video Extension [39]		✓		Framework di Microsoft Media Foundation. Consente la riproduzione di contenuti multimediali AV1 in ambiente Windows 10.

Tabella 3.1: I più promettenti video codec AV1

3.3 Scelte progettuali per l'analisi

3.3.1 Caratteristiche elaboratore

L'analisi prestazionale è stata eseguita mediante l'utilizzo di un pc-desktop Ubuntu v18.04.5 LTS con le seguenti specifiche tecniche:

- **CPU**: Intel Core i7-7700 @ 3.60GHz - 4 cores.
- **RAM**: 64 GB - 4 Bank DDR4 da 16GB @ 2400MHz.

3.3.2 Sequenze video testate

Il contenuto testato non è soggetto a copyright. I filmati **non compressi** impiegati per le analisi sono stati estrapolati da sequenze disponibili su Xiph.org [40] e Sjtu.edu.cn [41].

Nella Tabella 3.2 sono presentate le caratteristiche delle sequenze video testate sull'elaboratore.

Nome sequenza	Risoluzione	Frame-rate	N° frame
Big buck bunny	704 x 480	24 Hz	300
Elephants dream			
Stockholm	1280 x 720	60 Hz	
Shields		50 Hz	
Sunflower	1920 x 1080	25 Hz	
Tractor			
Marathon	3840 x 2160	30 Hz	150

Tabella 3.2: Sequenze video testate sull'elaboratore

Inoltre ogni sequenza presenta:

- schema di sottocampionamento della crominanza: **4:2:0**
- profondità di colore, ossia il numero di bit usati per rappresentare il colore di un pixel: **8 bit**.

3.3.3 Materiale utilizzato e relative configurazioni

Per eseguire le codifiche, le decodifiche e i confronti nelle performance dei formati di compressione video, sono stati adoperati i software **open source** elencati in Tabella 3.3.

Di seguito sono riportate anche le configurazioni utilizzate per eseguire le codifiche, alla base delle quali vi è un buon *trade-off* tra tempo di codifica e qualità risultante, in maniera da evitare operazioni eccessivamente onerose mantenendo una buona qualità percepita dopo la codifica dei contenuti.

Software	Libreria	Versione
FFmpeg [42]		N-99801-gfbb44bc
	libx264	152-r2854-e9a5903
	libx265	2.6
	libaom-av1	2.0.0
SVT-AV1		0.8.5-66-g4c27d404
RAV1E		0.4.0-alpha
DAV1D		0.8.0-20-g802790f
GAV1		0.16.3
VMAF		1.5.3

Tabella 3.3: Software impiegati per eseguire le codifiche, le decodifiche e valutare le prestazioni in ambiente desktop

Configurazioni codificatori:

- **x264** (AVC): `ffmpeg -i <input.y4m> -vcodec libx264 -b <bitrate> -preset slow -tune psnr -f mpeg2video <output.h264>`
- **x265** (HEVC): `ffmpeg -i <input.y4m> -vcodec libx265 -b <bitrate> -preset slow -tune psnr -f mpeg2video <output.h265>`
- **aom** (AV1): `ffmpeg -i <input.y4m> -vcodec libaom-av1 -b <bitrate> -cpu-used 3 -enable-intrabc true -tune psnr -f mpeg2video <output.av1>`
- **svt-av1** (AV1): `SvtAv1EncApp -i <input.y4m> --preset 4 --rc 1 --tbr <bitrate> -b <output.ivf>`
- **rav1e** (AV1): `rav1e <input.y4m> -s 1 --tiles 4 --tune Psnr -b <bitrate> -o <output.ivf>`

Nelle codifiche è stato utilizzato un algoritmo di controllo del *rate* in modalità **VBR** (*bit-rate* variabile), che consente, a differenza di quanto avviene nelle strategie a *bit-rate* costante (CBR), di usare un *rate* di codifica più elevato, dunque maggiore spazio per la memorizzazione, nelle scene più complesse del contenuto multimediale.

3.4 Risultati ottenuti

Di seguito verranno presentati i più rilevanti risultati ottenuti dell'analisi dei parametri di interesse menzionati nella sezione 3.1.

Per poter al meglio valorizzare le prestazioni di ogni codificatore e non essendoci particolari vincoli temporali, sono state eseguite anche delle codifiche ***multi-pass***, in cui, in un primo passaggio, viene effettuato lo studio della sequenza processata e, in un secondo passaggio, è eseguita la codifica sulla base delle caratteristiche raccolte. Tuttavia, l'encoder ***rav1e*** ha prodotto, in tale modalità, dei contenuti non valutabili, in quanto la riproduzione non è fluida. Per tale ragione, non è stato preso in considerazione nelle analisi prestazionali in modalità *multi-pass*.

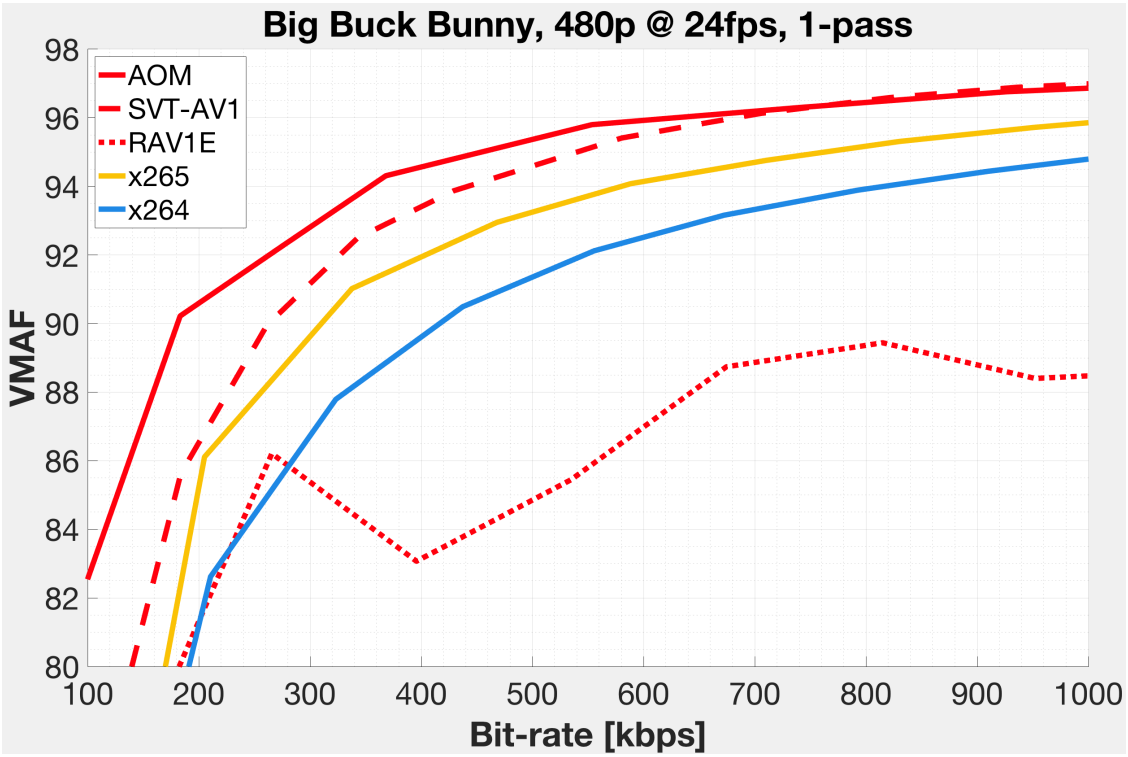
Al fine di eseguire in maniera efficiente ed automatica tutte le codifiche, le decodifiche e la raccolta dei dati riguardanti le prestazioni ottenute, sono stati sviluppati ed utilizzati degli appositi **shell script** e **programmi C** di supporto. È possibile reperire gli script realizzati tramite la repository GitHub [43] e, inoltre, alcuni esempi sono stati riportati in Appendice A.

Tutti i dati ricavati dai test sono stati memorizzati nella repository GitHub [44] e nella presentazione dei risultati i contenuti vengono suddivisi in base alla risoluzione.

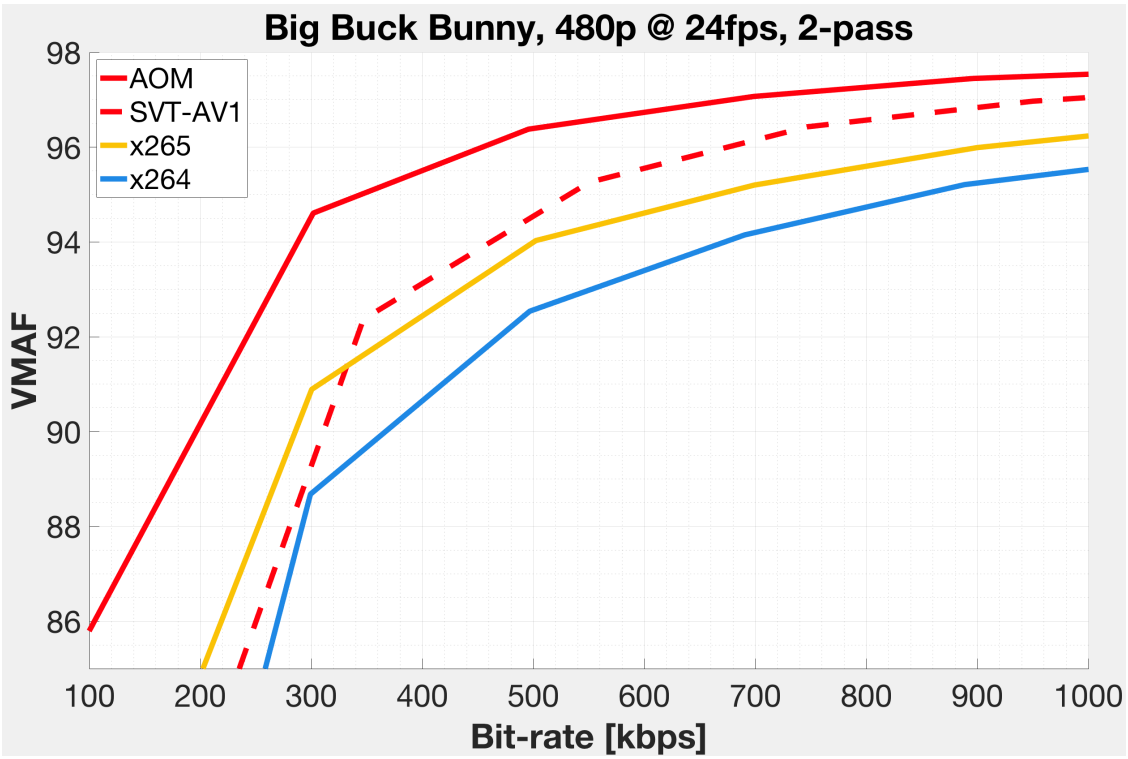
3.4.1 Standard Definition (480p)

Le prime sequenze analizzate sono due brevi cortometraggi d'animazione aventi una risoluzione di 704x480 (SD) e *frame-rate* pari a 24fps.

Nella Figura 3.1 sono riportate le curve rate-quality relative ai valori di VMAF ottenuti codificando la sequenza ***big buck bunny*** in modalità *1-pass* e *2-pass*. È possibile notare che l'encoder rav1e (AV1) ottiene delle prestazioni nettamente inferiori rispetto gli altri codificatori. D'altro canto, aom (AV1) raggiunge le valutazioni migliori e garantisce, a parità di valori, un risparmio complessivo nel rate di codifica superiore al 60% rispetto a x264 (AVC) e al 40% rispetto a x265 (HEVC). Infine, svt-av1 (AV1) ha un andamento inizialmente simile a x265, ma aumentando il *rate* di codifica migliorano anche le prestazioni dell'encoder, fino ad uguagliare aom.

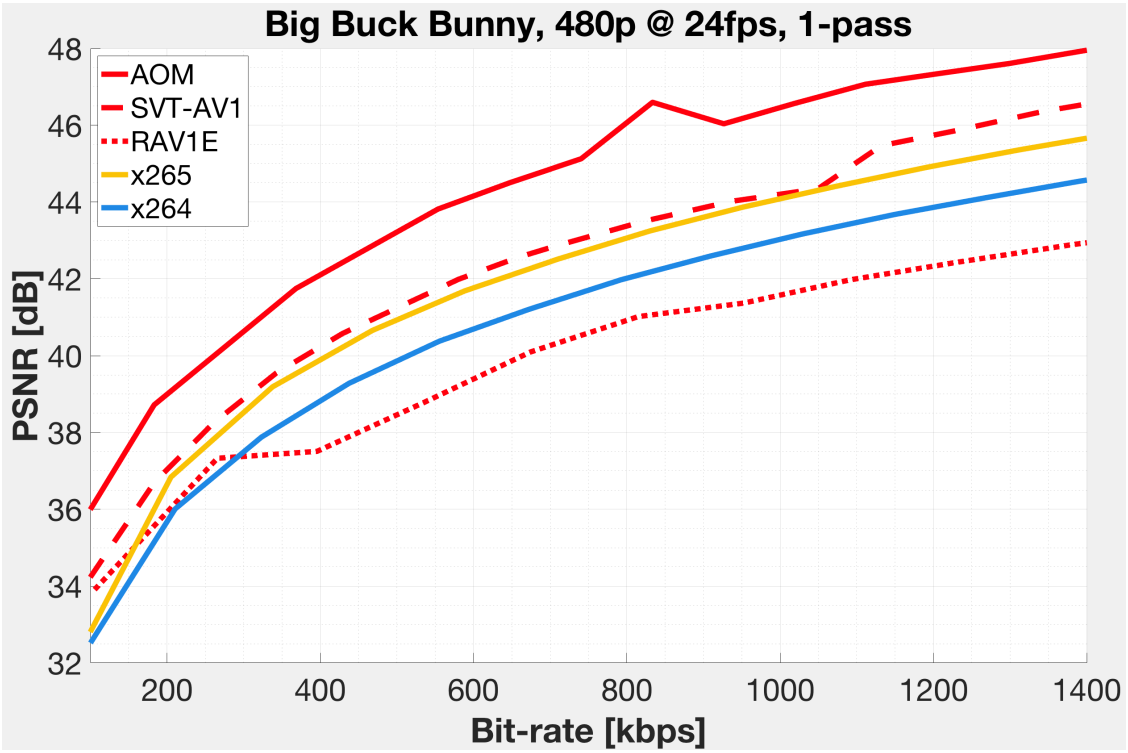


(a) codifica 1-pass

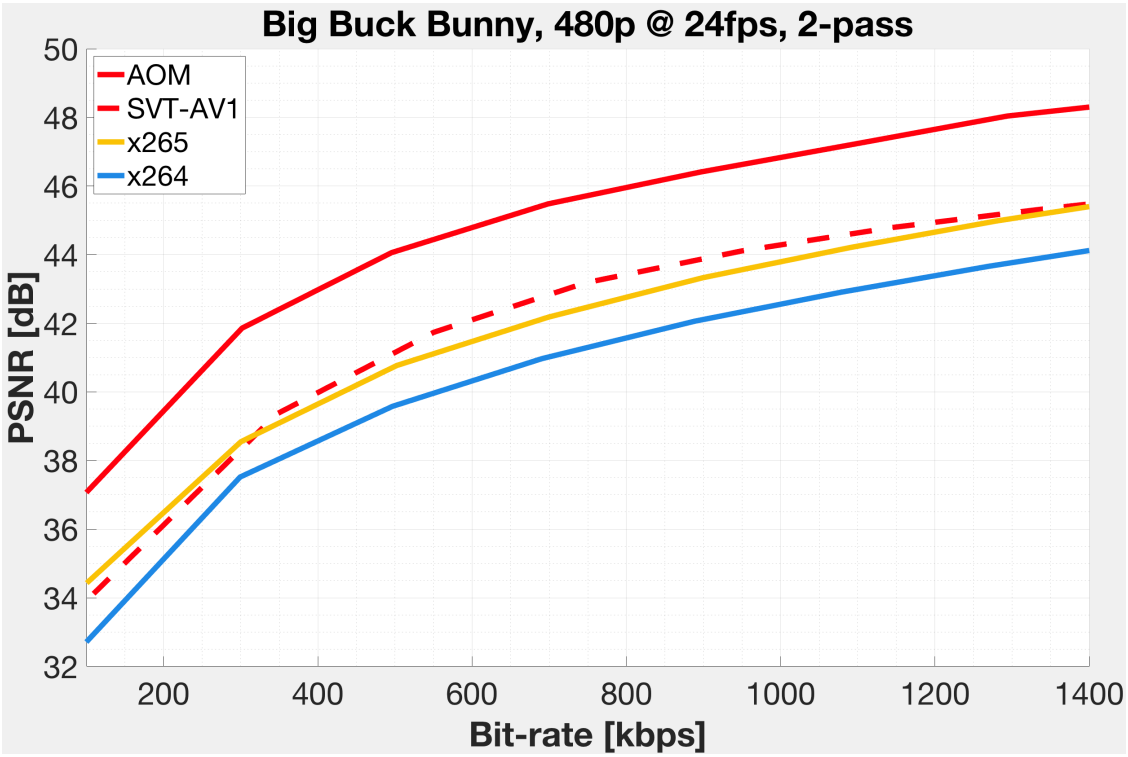


(b) codifica 2-pass

Figura 3.1: Big buck bunny (480p), grafici confronto valori VMAF



(a) codifica 1-pass



(b) codifica 2-pass

Figura 3.2: Big buck bunny (480p), grafici confronto valori PSNR

Nei grafici in Figura 3.2 sono riportate le curve *rate-quality* relative ai valori di PSNR. I risultati ottenuti, espressi in decibel, sono simili ai precedenti e la codifica aom in modalità *2-pass* garantisce un risparmio nel *rate* di codifica che varia dal 50-70% rispetto a x265 e del 65-75% rispetto a x264.

Un ulteriore e interessante confronto può essere effettuato osservando direttamente un fotogramma estratto da contenuti in formato AVC, HEVC e AV1 allo stesso rate di codifica. Dalla Figura 3.3 è facilmente possibile notare come nella porzione di fotogramma in formato AV1 il piumaggio risulta essere molto più dettagliato ed il cielo presenta una "*bloccosità*" praticamente inesistente.

Le valutazioni complessive, in termini di PSNR e VMAF, dei contenuti video utilizzati sono riportate nella Tabella 3.4.

Encoder	**Formato**	**Bit-rate**	**PSNR**	**VMAF**
x264	AVC	191.76 kbps	35.57 dB	81.32
x265	HEVC	183.21 kbps	36.31 dB	84.64
aom	AV1	183.20 kbps	38.72 dB	90.22

Tabella 3.4: Big buck bunny (480p), valutazioni sequenze AVC, HEVC e AV1 allo stesso rate di codifica

Nella tabella sottostante è, invece, riportato il tempo medio impiegato per effettuare le codifiche, espresso in termini di minuti che la CPU ha trascorso in *user e kernel mode*.

Encoder	**Formato**	**1-pass time**	**2-pass time**
x264	AVC	0.15 min	0.17 min
x265	HEVC	0.51 min	0.87 min
aom	AV1	4.93 min	5.25 min
svt-av1		6.28 min	6.77 min
rav1e		6.83 min	–

Tabella 3.5: Big buck bunny (480p), tempo medio codifiche multi-pass

(a) AV1 vs AVC

(b) AV1 vs HEVC

Figura 3.3: Big buck bunny (480p), confronto fotogramma in formato AVC, HEVC e AV1

Dai dati nella Tabella 3.5, si evince che i codificatori AV1 hanno velocità di codifica nettamente inferiori. In particolar modo, aom, l'encoder che ha prodotto i migliori risultati, risulta essere oltre 40 volte più lento rispetto x264 e circa 10 volte più lento rispetto x265.

Nella Tabella 3.6 sono riportate le tempistiche medie impiegate per eseguire le decodifiche, espresse in termini di secondi che la CPU ha trascorso *in user e kernel mode*. In tal caso, non vi sono considerevoli differenze fra i vari decodificatori, e, fra tutti, spiccano dav1d e gav1 (AV1), che ottengono dei risultati migliori rispetto a x265 (HEVC).

Decoder	**Formato**	**Dec time**
x264	AVC	0.46 sec
x265	HEVC	0.65 sec
dav1d	AV1	0.51 sec
gav1		0.63 sec
svt-av1		0.78 sec
aom		1.34 sec

Tabella 3.6: Big buck bunny (480p), tempo medio decodifica

La seconda sequenza studiata è ***elephants dream***, e nella Figura 3.4 vengono rappresentate le curve *rate-quality* relative ai valori VMAF ottenuti tramite codifiche in modalità *1-pass*. Con tale contenuto, le prestazioni ottenute dall'encoder rav1e migliorano anche se restano le peggiori. Mentre, aom e svt-av1 presentano performance molto simili e garantiscono, a parità di valutazioni, un risparmio nel *rate* di codifica superiore al 20% rispetto a x265 e al 35% rispetto a x264.

Da ultimo, nella Tabella 3.7 sono riportati i tempi medi necessari per eseguire le codifiche della sequenza sopraccitata, espressi in termini di minuti effettivamente trascorsi dall'inizio alla fine dell'esecuzione (*wall clock*) e di minuti trascorsi dalla CPU *in user + kernel mode*. È possibile notare che svt-av1, sfruttato al meglio i multi-core dell'elaboratore per il processamento del contenuto in parallelo, riesce ad ottenere un tempo medio che è solamente 3 volte superiore rispetto a x265.

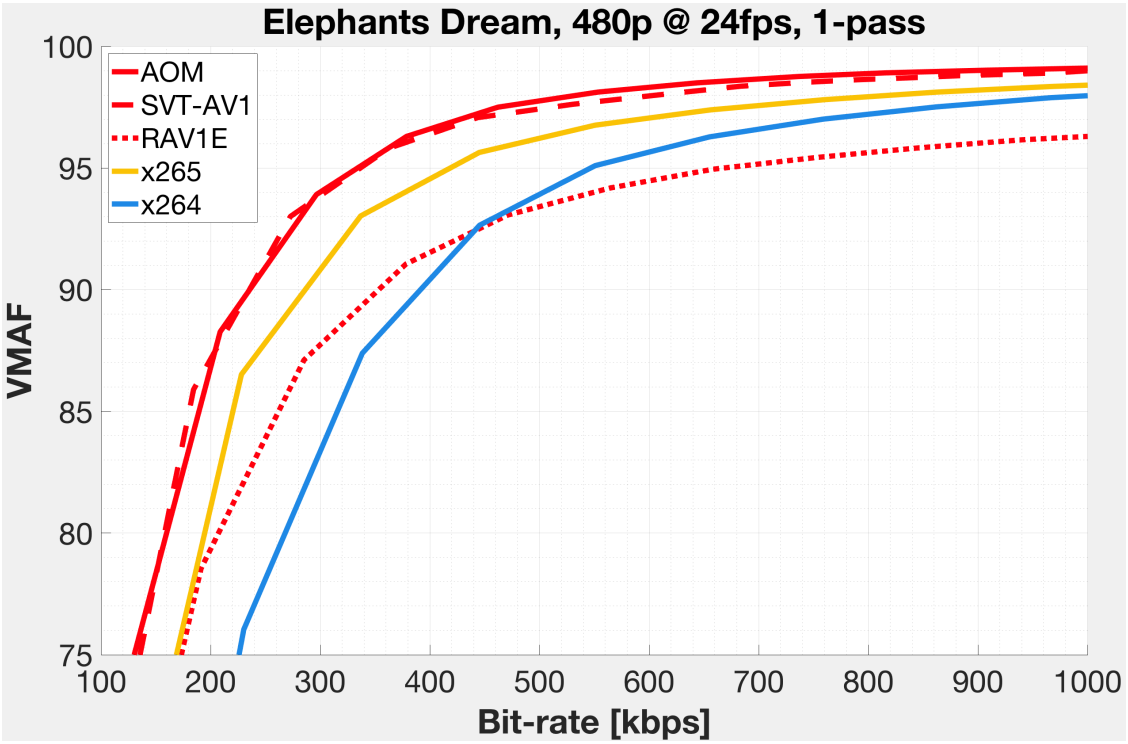


Figura 3.4: Elephants dream (480p), grafico confronto valori VMAF

Encoder	Formato	Usr + Sys time	Wall clock
x264	AVC	0.24 min	0.05 min
x265	HEVC	1.19 min	0.40 min
aom	AV1	4.45 min	2.08 min
svt-av1		6.08 min	1.28 min
rav1e		8.59 min	2.96 min

Tabella 3.7: Elephants dream (480p), tempo medio codifica

3.4.2 High Definition (720p)

La prima sequenza ad alta risoluzione (HD) studiata è ***shields***, un filmato, con *frame-rate* pari a 50fps, riguardante una guida turistica che presenta una serie di scudi medievali. Attraverso le curve *rate-quality* relative ai valori di VMAF (Figura 3.5) e PSNR (Figura 3.6) è possibile stabilire, anche in questo caso, che il codificatore aom ottiene le valutazioni migliori, garantendo un risparmio nel *rate* di codifica quasi pari al 70% rispetto a x264 e superiore al 40% nei confronti di x265.

I codificatori rav1e e svt-av1 ottengono delle prestazioni che si collocano nel mezzo fra x265 e x264.

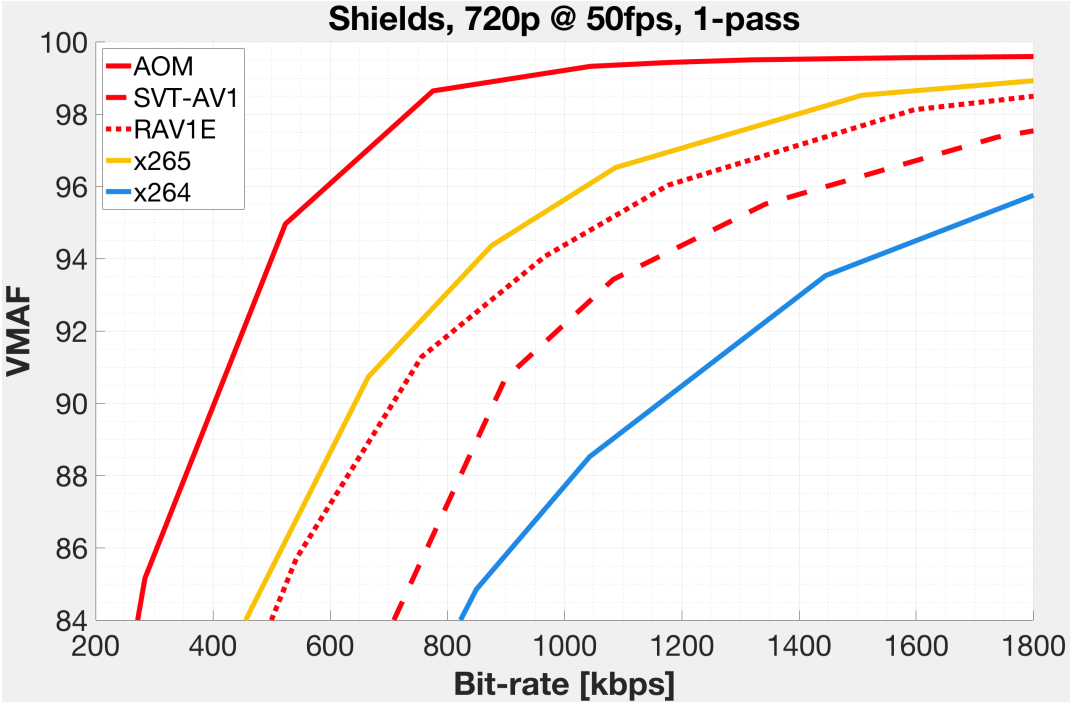


Figura 3.5: Shields (720p), grafico confronto valori VMAF

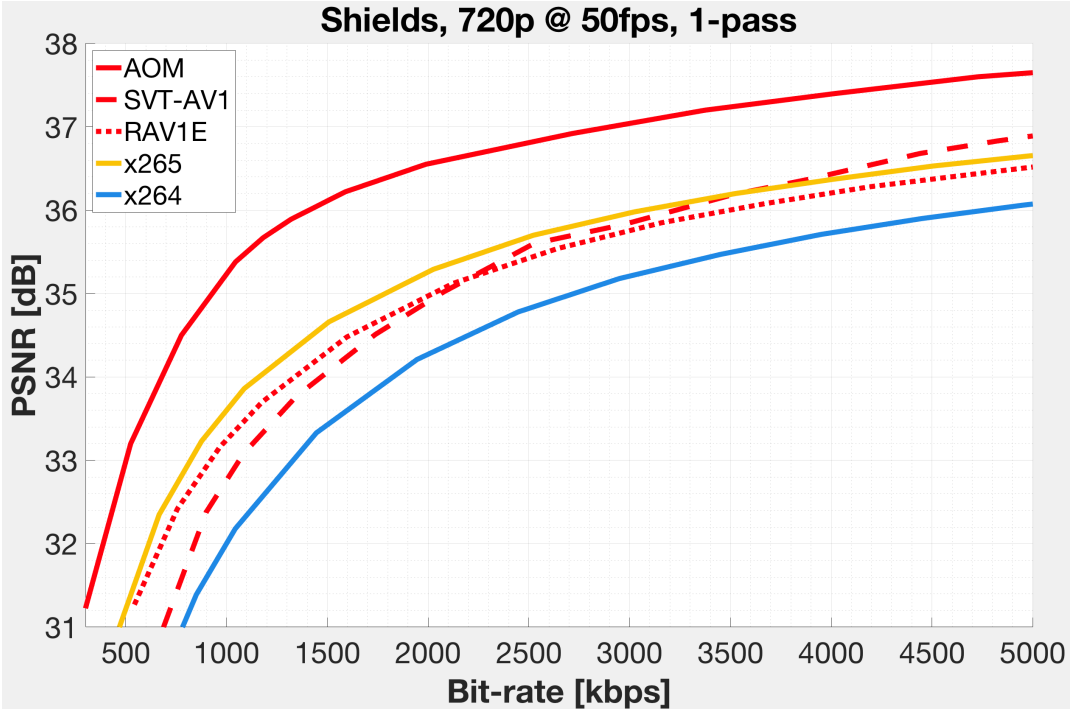


Figura 3.6: Shields (720p), grafico confronto valori PSNR

Nella Tabella 3.8 è riportato il tempo medio necessario per eseguire le codifiche, espresso in minuti che la CPU ha trascorso in *user + kernel mode* e al relativo *wall clock*. Anche in termini di velocità di codifica, aom ottiene prestazioni migliori rispetto gli altri codificatori AV1, ma è circa 50 volte più lento rispetto x264 e quasi 4 volte rispetto x265.

Encoder	**Formato**	**Usr + Sys time**	**Wall clock**
x264	AVC	0.35 min	0.07 min
x265	HEVC	1.52 min	0.34 min
aom	AV1	17.19 min	7.31 min
svt-av1		20.80 min	3.44 min
rav1e		30.45 min	8.81 min

Tabella 3.8: Shields (720p), tempo medio codifica

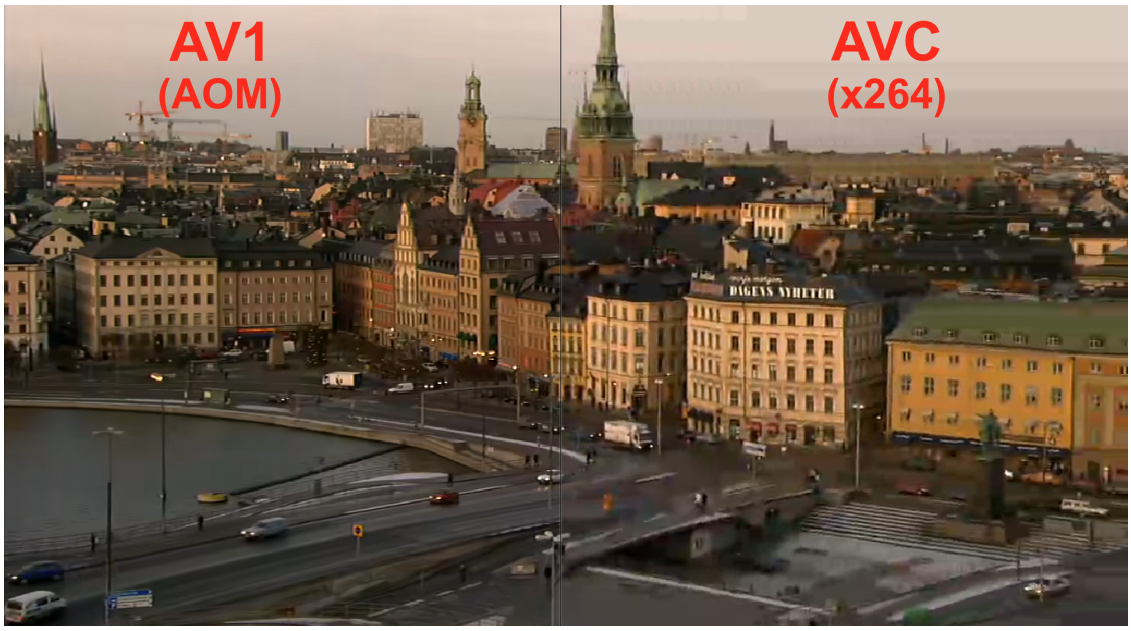
Invece, nella Tabella 3.9, sono riportate le tempistiche medie, espresse in termini di secondi che la CPU ha trascorso in *user e kernel mode*, impiegate per effettuare le decodifiche.

Decoder	**Formato**	**Dec time**
x264	AVC	1.16 sec
x265	HEVC	1.59 sec
dav1d	AV1	1.46 sec
gav1		1.61 sec
svt-av1		2.20 sec
aom		4.34 sec

Tabella 3.9: Shields (720p), tempo medio decodifica

Il secondo video analizzato a risoluzione HD è ***stockholm***, una panoramica sulla vecchia città di Stoccolma. Tale sequenza presenta un *frame-rate* pari a 60fps, che è più elevato rispetto ai contenuti finora analizzati.

Nella Figura 3.7 sono riportate delle porzioni di uno stesso fotogramma estratto da contenuti in formato AV1, AVC e HEVC allo stesso *rate* di codifica.



(a) AV1 vs AVC

(b) AV1 vs HEVC

Figura 3.7: Stockholm (720p), confronto fotogramma in formato AVC, HEVC e AV1

È possibile notare che nella porzione in formato AV1 il cielo presenta una "*bloccosità*" inesistente e i dettagli dei palazzi, dei veicoli e della statua collocata nella piazza sono molto più nitidi.

Nella Tabella 3.10 sono riportate le valutazioni, in termini di PSNR e VMAF, relative alle sequenze da cui sono stati estrapolati i fotogrammi.

Encoder	Formato	Bit-rate	PSNR	VMAF
x264	AVC	348.12 kbps	28.63 dB	55.73
x265	HEVC	347.62 kbps	31.88 dB	82.59
aom	AV1	347.91 kbps	33.34 dB	90.69

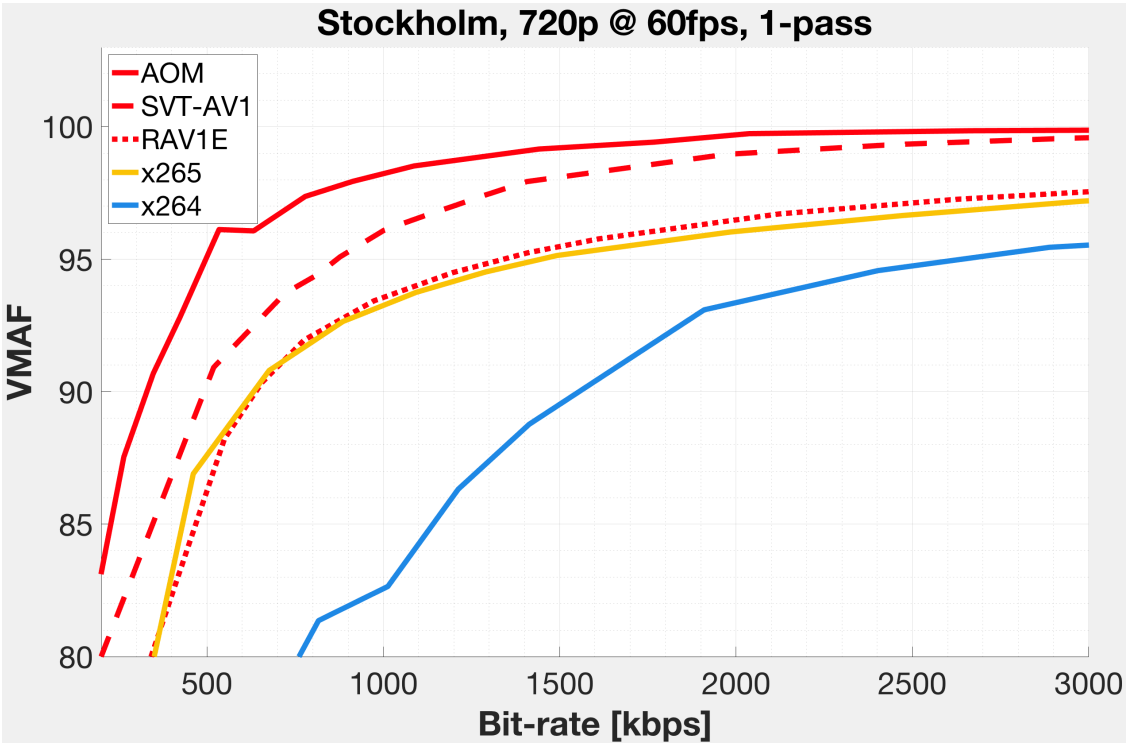
Tabella 3.10: Stockholm (720p), valutazioni sequenze AVC, HEVC e AV1 allo stesso rate di codifica

Nella Tabella 3.11 sono, invece, riportati i tempi medi necessari per eseguire le codifiche e decodifiche della sequenza ***stockholm***, misurati in termini di minuti (codifiche) e secondi (decodifiche) che la CPU ha trascorso in *user e kernel mode*. In tal caso, il codificatore aom è circa 5 volte più lento rispetto a x265 e 20 volte rispetto a x264, mentre il decoder dav1d offre performance migliori perfino di x264.

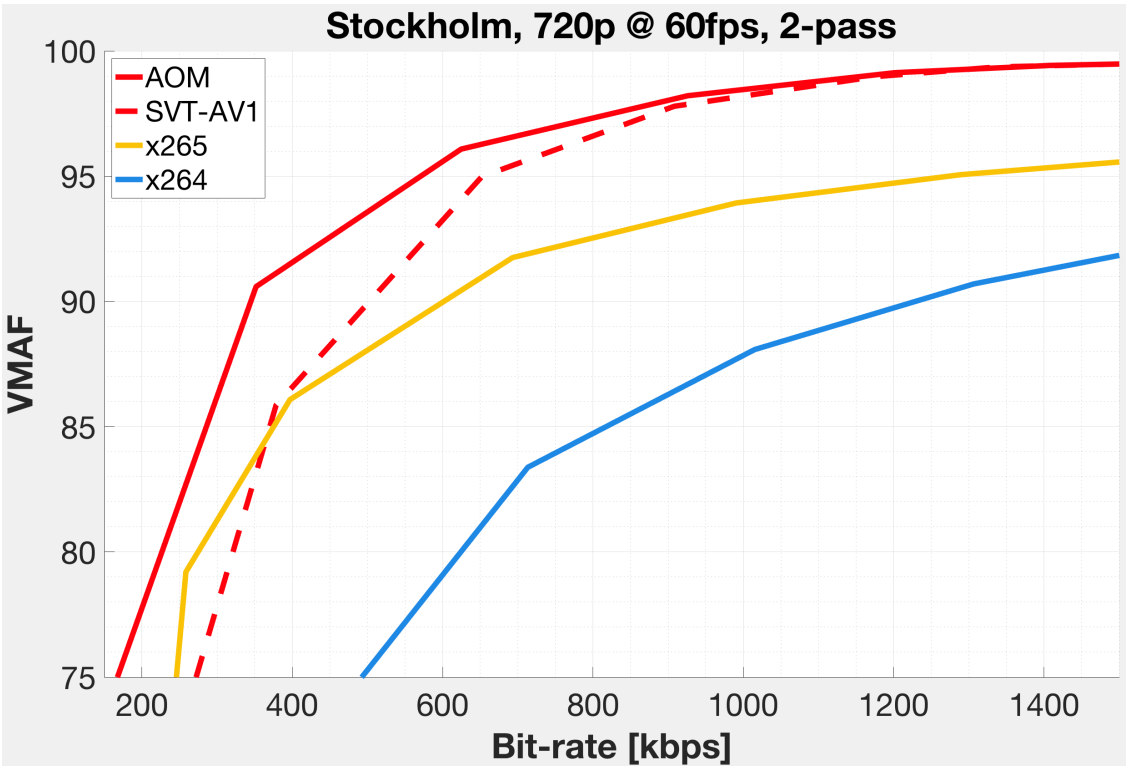
Enc/Dec	Formato	Enc time	Dec time
x264	AVC	0.41 min	1.20 sec
x265	HEVC	1.78 min	2.03 sec
aom	AV1	8.77 min	2.64 sec
svt-av1		14.79 min	1.34 sec
rav1e		16.02 min	–
dav1d		–	0.88 sec
gav1		–	1.25 sec

Tabella 3.11: Stockholm (720p), tempo medio codifiche e decodifiche

Infine, in Figura 3.8 sono rappresentate le curve *rate-quality* relative ai valori di VMAF ottenuti tramite codifiche *multi-pass*. Con tale sequenza, tutti i codificatori AV1 ottengono delle valutazioni più elevate rispetto ai codificatori x265 e x264.



(a) codifica 1-pass



(b) codifica 2-pass

Figura 3.8: Stockholm (720p), grafici confronto valori VMAF

3.4.3 Full HD (1080p)

La risoluzione Full HD è, generalmente, la più impiegata per la riproduzione e la trasmissione di contenuti ad alta definizione. La prima sequenza Full HD (1920 x 1080 @ 25fps) ad essere analizzata è ***tractor***, una breve ripresa di un trattore durante l'aratura di un campo. Nella Figura 3.9 sono mostrate delle porzioni di uno stesso fotogramma estratto da sequenze codificate in AVC, HEVC e AV1. Osservando i fotogrammi è possibile notare come i colori nella porzione in formato AV1 sono molto più accesi, specialmente il rosso della ruota posteriore, e il campo presente nello sfondo è molto più nitido rispetto alle porzioni codificate in AVC e HEVC. Ogni sequenza presenta un rate di codifica simile e nella Tabella 3.12 sono riportati i dati e le valutazioni delle rispettive sequenza.

Encoder	Formato	Bit-rate	PSNR	VMAF
x264	AVC	1010.11 kbps	31.61 dB	60.19
x265	HEVC	1003.84 kbps	34.64 dB	79.04
aom	AV1	999.32 kbps	35.88 dB	84.23

Tabella 3.12: Tractor (1080p), valutazioni sequenze AVC, HEVC e AV1 allo stesso rate di codifica

Nella Tabella 3.13 è, invece, riportato il tempo medio necessario per eseguire le codifiche e le decodifiche. Aumentando la risoluzione dei contenuti incrementano anche i tempi di codifica, ma i rapporti fra i vari codificatori rimangono pressoché invariati. Difatti, aom è circa 4 volte più lento rispetto a x265 e 30 volte rispetto a x264. Mentre, i decodificatori AV1 ottengono delle buone prestazioni.

Enc/Dec	Formato	Enc time	Dec time
x264	AVC	0.89 min	3.03 sec
x265	HEVC	6.05 min	4.16 sec
aom	AV1	27.21 min	11.21 sec
svt-av1		47.51 min	5.76 sec
rav1e		47.62 min	–
dav1d		–	2.56 sec
gav1		–	3.97 sec

Tabella 3.13: Tractor (1080p), tempo medio codifica e decodifica

(a) AVC vs AV1

(b) HEVC vs AV1

Figura 3.9: Tractor (1080p), confronto fotogramma in formato AVC, HEVC e AV1

Nelle Figure 3.10 e 3.11 sono rappresentate le curve *rate-quality* relative alle valutazione di VMAF e PSNR. Con tale sequenza: x264 ha una risposta nettamente inferiore rispetto gli altri codificatori, svt-av1 e rav1e hanno un andamento simile a x265 e aom raggiunge le valutazioni più elevate, garantendo un risparmio medio nel rate di codifica pari al 48% rispetto x264 e del 17% rispetto x265.

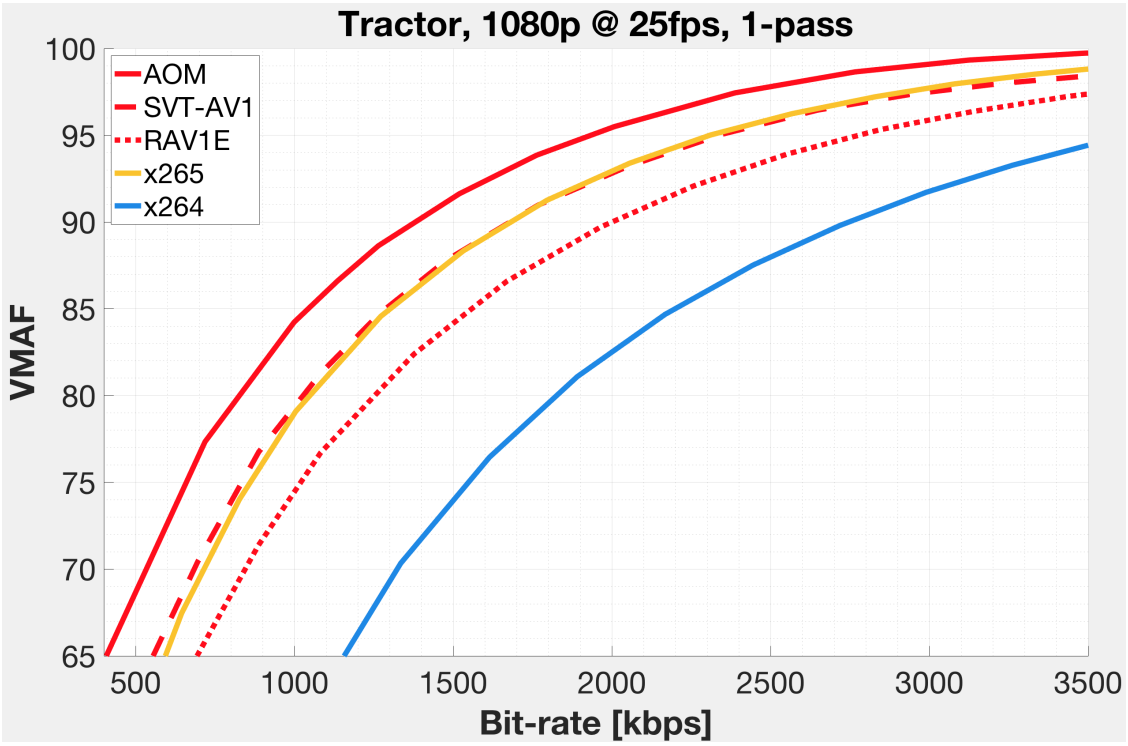


Figura 3.10: Tractor (1080p), grafico confronto valori VMAF

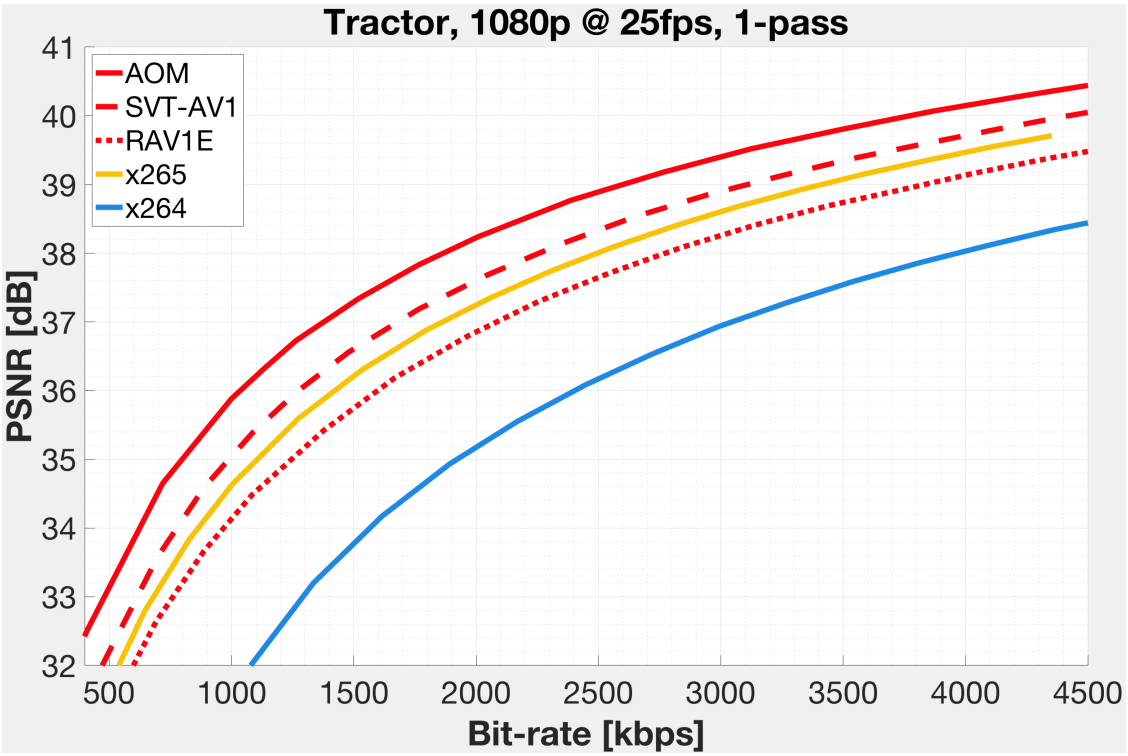


Figura 3.11: Tractor (1080p), grafico confronto valori PSNR

La seconda sequenza in Full HD analizzata è ***sunflower***, un breve filmato di un'ape al di sopra di un girasole. Il contenuto ha *frame-rate* pari a 25Hz e in Figura 3.12 sono rappresentate le curve *rate-quality* relative alle valutazioni VMAF. Con tale sequenza, il risparmio nel *rate* di codifica ottenuto dall'encoder aom è significativo, impiegando, a parità di valori, il 68% e il 47% di bit in meno rispetto ai codificatori x264 e x265.

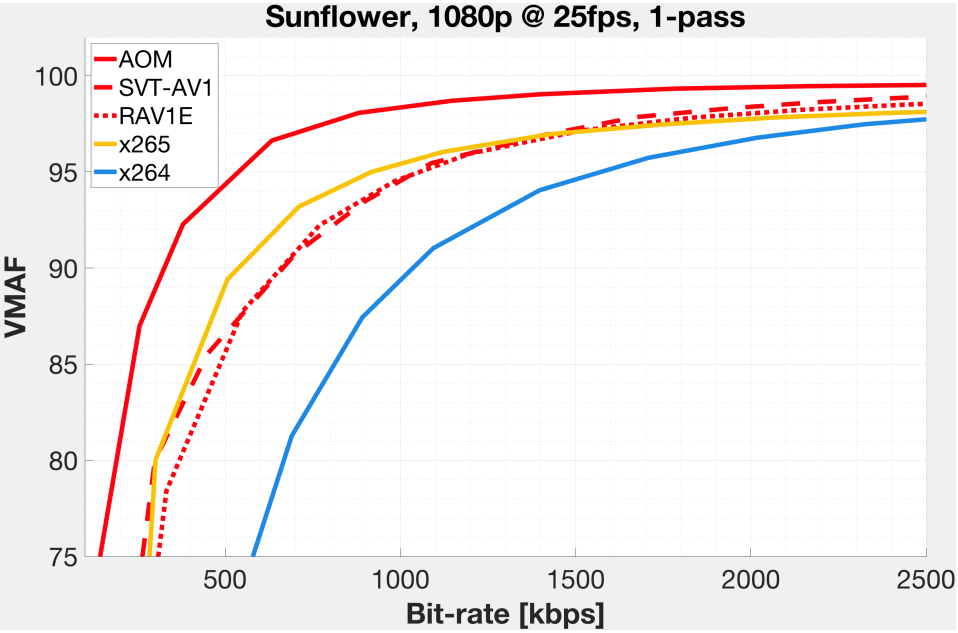


Figura 3.12: Sunflower (1080p), grafico confronto valori VMAF codifiche *1-pass*

Nella Tabella 3.14 sono, infine, riportati i tempi necessari per eseguire le codifiche in modalità *1-pass* e *2-pass*. Aom risulta oltre 20 volte più lento rispetto a x264 e circa 4 volte rispetto a x265.

Encoder	Formato	1-pass time	2-pass time
x264	AVC	0.92 min	1.05 min
x265	HEVC	6.40 min	7.38 min
aom	AV1	15.76 min	18.16 min
svt-av1		30.74 min	33.17 min
rav1e		39.91 min	–

Tabella 3.14: Sunflower (1080p), tempo medio codifiche 1-pass e 2-pass

3.4.4 4K (2160p)

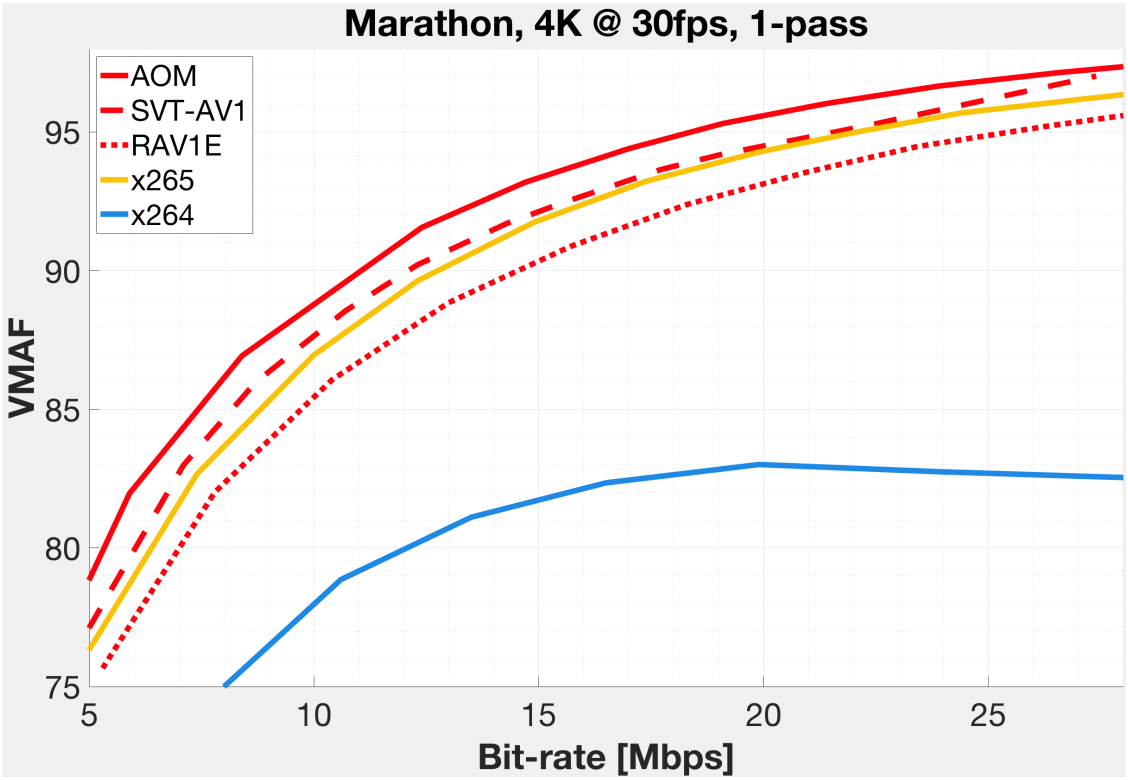
La quantità di contenuti a risoluzione 4K, definita anche come Ultra HD, è cresciuta considerevolmente nel corso degli ultimi anni. I primi banchi di prova utilizzati per far conoscere il formato e dimostrarne i vantaggi sono stati le Olimpiadi invernali del 2018 in Corea del Sud e il Campionato Mondiale di calcio, svolto nello stesso anno, in Russia, in cui tutti i match sono stati girati in 4K UHD e distribuiti via satellite, via cavo e tramite servizi di SVoD (*Subscription Video on Demand*).

Considerando che la quantità di dati necessari per la riproduzione di contenuti in 4K è nettamente superiore rispetto a contenuti in Full HD (il 4K è quattro volte più grande rispetto al Full HD), è stata analizzata solamente una sequenza video, chiamata ***marathon***, avente risoluzione 3840 x 2160 a 30fps. Nella Figura 3.13 e nella Figura 3.14 sono rappresentate le curve *rate-quality* relative rispettivamente ai valori di VMAF e PSNR ottenuti dalle codifiche della sequenza in modalità *multi-pass*. È possibile notare che il codificatore x264 ha una risposta nettamente inferiore rispetto agli altri codificatori, specialmente con codifiche in modalità *1-pass*. Aom, invece, continua ad ottenere le valutazioni migliori garantendo, in modalità *2-pass*, un risparmio medio nel rete di codifica pari al 35% rispetto a x264 e al 17% rispetto a x265.

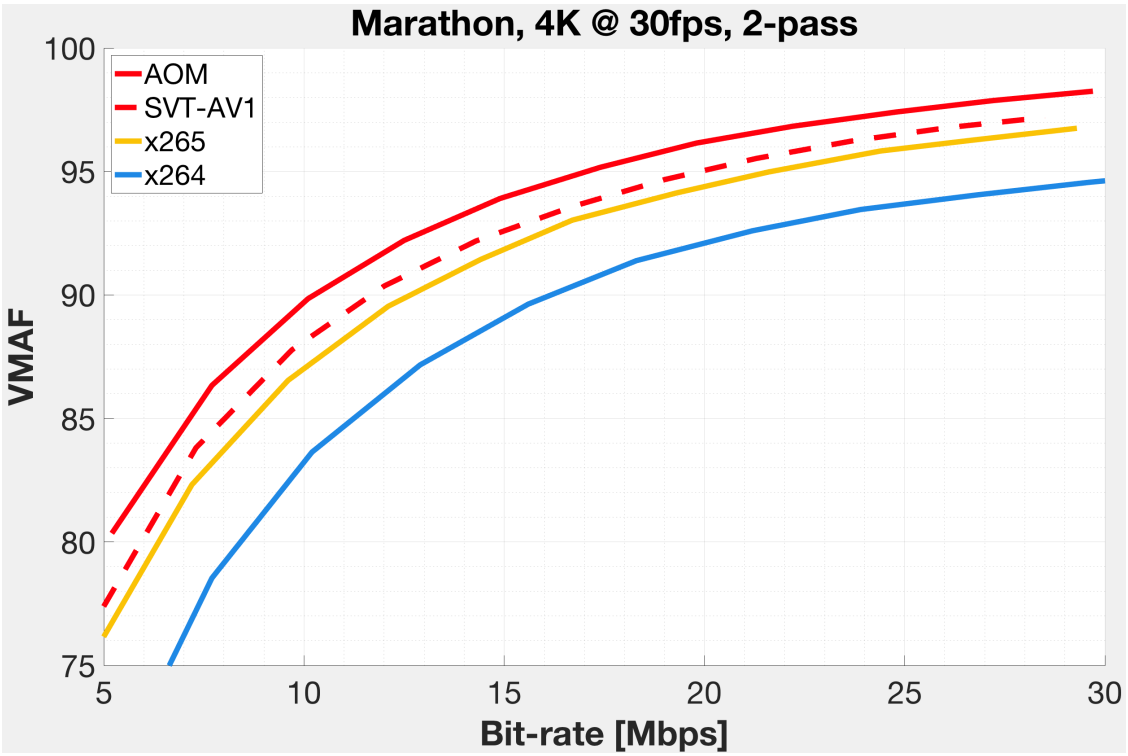
Nella Tabella 3.15 è riportato il tempo medio necessario per eseguire le codifiche. Il tempo medio impiegato per la codifica dei contenuti AV1 inizia ad essere significativo (circa un'ora e mezza per un filmato di appena cinque secondi), ma i rapporti fra gli encoder rimangono pressoché invariati. Difatti, aom risulta circa 4 volte più lento rispetto a x265 e 30 volte rispetto a x264.

Encoder	Formato	1-pass time	2-pass time
x264	AVC	2.54 min	3.04 min
x265	HEVC	19.27 min	22.93 min
aom	AV1	81.37 min	91.27 min
svt-av1		90.31 min	93.04 min
rav1e		93.92 min	–

Tabella 3.15: Marathon (4K), tempo medio codifiche multi-pass

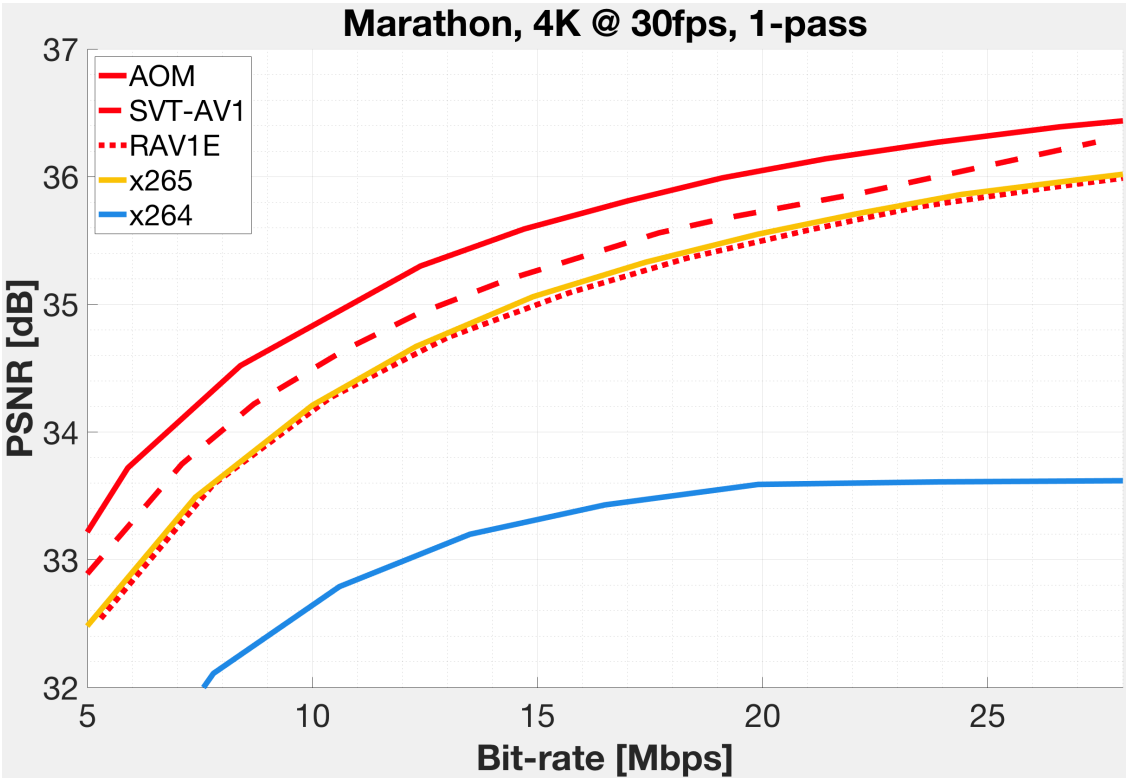


(a) codifica 1-pass

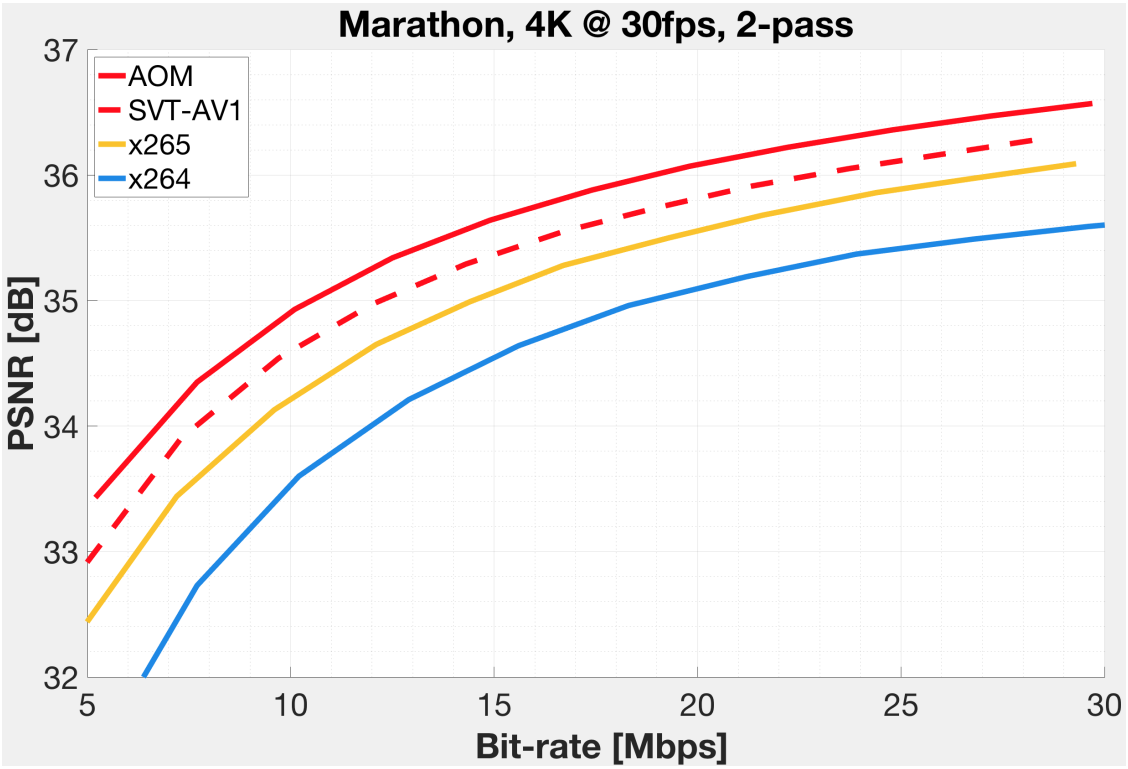


(b) codifica 2-pass

Figura 3.13: Marathon (4K), grafici confronto valori VMAF



(a) codifica 1-pass



(b) codifica 2-pass

Figura 3.14: Marathon (4K), grafici confronto valori PSNR

In ultima analisi, nella Tabella 3.15 sono riportati i tempi medi impiegati per effettuare le decodifiche. Attraverso tali statistiche, è possibile constatare che i decoder dav1d e gav1 ottengono, anche con contenuti a risoluzione 4K, delle ottime prestazioni, risultando molto più veloci del decoder x265.

Decoder	**Formato**	**Dec time**
x264	AVC	8.21 sec
x265	HEVC	14.77 sec
dav1d	AV1	5.84 sec
gav1		8.34 sec
svt-av1		13.43 sec
aom		21.07 sec

Tabella 3.16: Marathon (4K), tempo medio decodifica

Capitolo 4

pAV0NE: un lettore multimediale per Android

La seconda parte dell'attività è focalizzata sull'impatto che un decoder AV1 possa avere nei confronti dei dispositivi mobili. A tal proposito, considerando l'attuale scarsa implementazione del *video codec* AV1 in ambiente mobile, è stato sviluppato un lettore multimediale in grado di supportare il nuovo formato di compressione video.

4.1 Scelte progettuali

L'applicazione sviluppata si chiama **pAV0NE** ed è un lettore multimediale per Android che consente la fruizione sia di contenuti video memorizzati sul proprio dispositivo sia di contenuti video in streaming adattivo MPEG-DASH, *Dynamic Adaptive Streaming over HTTP*.

4.1.1 MPEG–DASH

Standard internazionale, approvato nel 2011, che specifica un formato interoperabile per la trasmissione di contenuti *streaming* su Internet tramite dei convenzionali server HTTP. L'attenzione dello standard è rivolta alla *user experience*, offrendo un adattamento dinamico in base alle capacità delle rete e del dispositivo.

Nella Figura 4.1 sono evidenziate in rosso le parti specificate dallo standard.

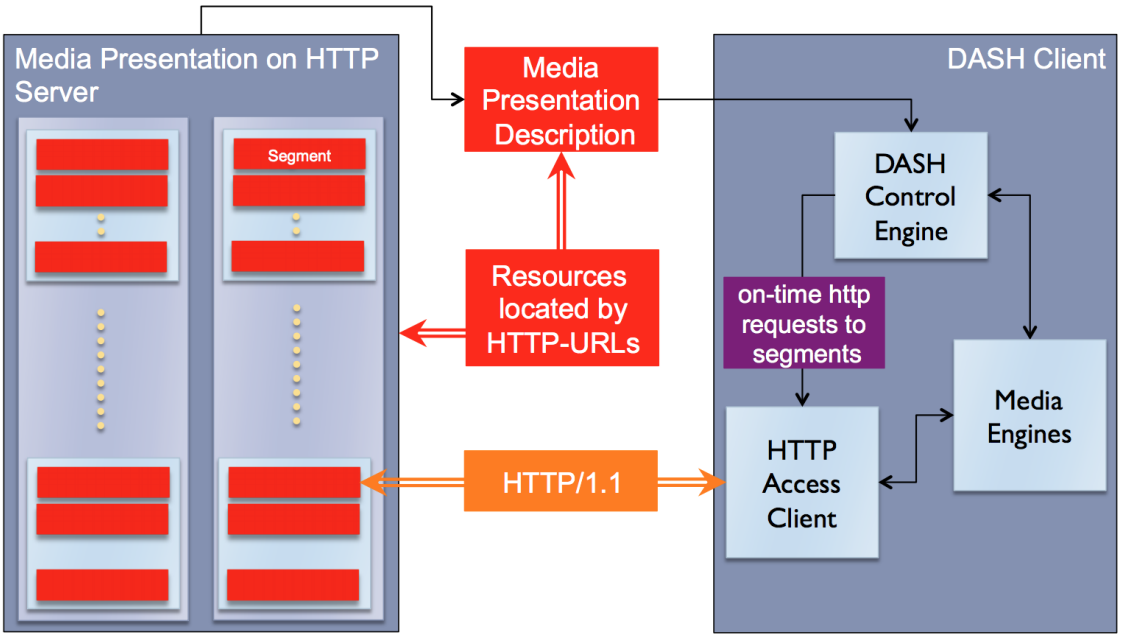


Figura 4.1: Caratteristiche specificate nello standard DASH

È possibile notare che viene specificato solamente:

- il modo in cui devono essere creati i segmenti che caratterizzano i contenuti memorizzati sul server;
- il modo in cui deve essere strutturato il *Media Presentation Description* (MPD), ovvero il manifesto da inviare al client contenente le informazioni riguardanti la codifica, la risoluzione, la banda, ecc. dei vari contenuti multimediali disponibili;
- la modalità di accesso alle risorse allocate sul server.

È bene sottolineare che la logica di adattamento non è specificata nello standard. La logica di controllo dell'applicazione viene affidata interamente al client, così facendo è possibile adoperare dei server web *stateless* dal costo estremamente moderato.

4.1.2 Strumenti per lo sviluppo

Nello sviluppo dell'applicazione, sono stati impiegati i seguenti software *open source*:

Fonte Figura 4.1 e riferimenti [45].

- ***ExoPlayer*** (v2.13.3) [46]: lettore multimediale sviluppato da Google e implementabile nei dispositivi *Android-based*. Fornisce una alternativa alle API MediaPlayer dedicate alla riproduzione di contenuti audio e video.

- ***GAV1*** (v0.16.3) [38]: decoder AV1 sviluppato da Google. È stato progettato per lavorare in maniera ottimale in dispositivi Android *Arm-powered*.

- ***Server HTTP Apache*** (v2.4.29) [47]: server che fornisce servizi in sincronia con gli attuali standard HTTP. È stato usato per la trasmissione di contenuti multimediali DASH.

4.2 Specifiche e design dell'applicazione

4.2.1 Caratteristiche generali

pAV0NE (Figura 4.2) è un lettore multimediale per Android sviluppato in **Kotlin** e basato su ExoPlayer. Consente la riproduzione sia di contenuti video memorizzati sul proprio dispositivo sia di contenuti video in streaming adattivo MPEG-DASH.

Figura 4.2: Logo pAV0NE

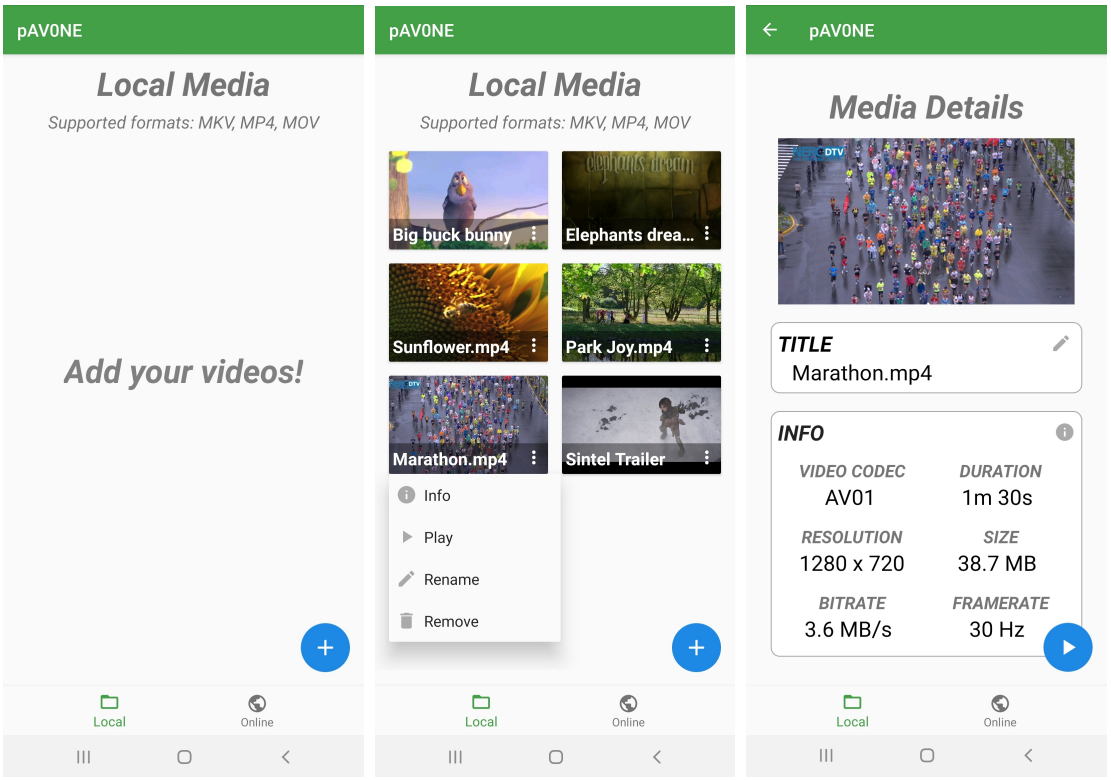
Grazie all'implementazione del decoder gav1, è in grado di supportare contenuti video in formato AV1. Oltre a ciò, può interfacciarsi anche con i più comuni formati di compressione video, come AVC e HEVC, usufruendo dei decodificatori nativamente presenti nei dispositivi. I formati contenitori supportati sono MKV, MP4 e MOV.

È possibile reperire il codice sorgente dell'applicazione nella repository GitHub [48].

4.2.2 Riproduzione contenuti locali

La schermata di Home dell'applicazione (Figura 4.3a) è rivolta alla riproduzione di contenuti multimediali memorizzati sul proprio dispositivo. Consente di aggiungere i propri filmati, che, dopo essere stati selezionati, vengono mostrati in una disposizione a griglia.

La persistenza dei contenuti caricati nell'applicazione è ottenuta attraverso l'implementazione della classe RoomDatabase di Android, che permette di creare e gestire database in maniera efficiente ed affidabile, sfruttando tutta la potenza di SQL.



(a) (b) (c)

Figura 4.3: pAV0NE: screenshots schermate video locali

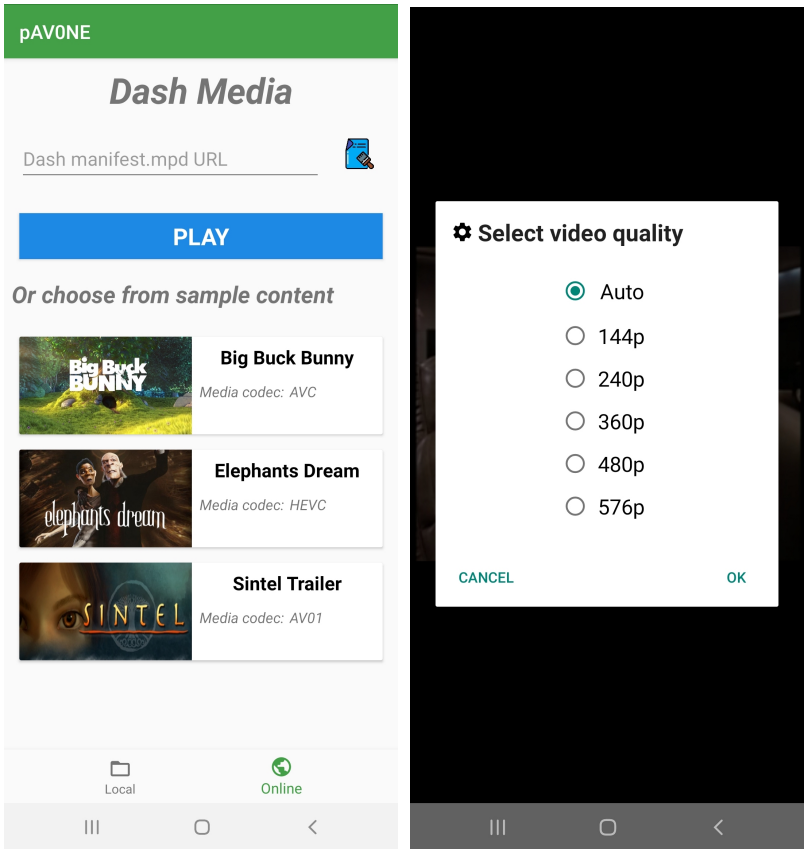
Facendo click sull'apposito menù a tendina disponibile per ogni contenuto (Figura 4.3b), è possibile accedere alle caratteristiche della sequenza e visualizzare dettagli come il *video codec* e la risoluzione (Figura 4.3c). Tali informazioni sono ricavate attraverso l'implementazione della classe MediaExtractor di Android.

Una volta avviato il playback, maggiormente apprezzabile in modalità a schermo intero (Figura 4.4), è possibile mandare la riproduzione in avanti, in indietro o in loop.

Figura 4.4: pAV0NE: screenshot schermata riproduzione video locale

4.2.3 Riproduzione contenuti in streaming DASH

Tramite la barra di navigazione, è possibile accedere all'area dedicata alla riproduzione di contenuti in streaming adattivo DASH (Figura 4.5a).



(a) (b)

Figura 4.5: pAV0NE: screenshots schermate video streaming DASH

Per avviare la riproduzione di uno streaming DASH, è possibile selezionare una delle sequenze presenti fra i contenuti di esempio, appositamente memorizzati su un server HTTP di supporto, oppure occorre specificare nel riquadro dedicato l'URL di uno determinato *Media Presentation Description* (manifest.mpd).

Durante la riproduzione degli streaming DASH, a differenza di quanto avviene con i contenuti memorizzati localmente, è anche possibile modificare la risoluzione del video, scegliendone una fra quelle disponibili (Figura 4.5b).

Capitolo 5

Analisi prestazionale in ambiente mobile

Nel capitolo sono valutate le prestazioni dei più promettenti decodificatori AV1 esistenti in ambiente mobile. I risultati ottenuti sono confrontati con le performance dei decoder AVC e HEVC nativamente presenti nei dispositivi.

5.1 Parametri presi in esame

I contenuti video rappresentano attualmente il 48% di tutto il traffico sulle reti mobili. Nella Figura 5.1 sono presentate le applicazioni maggiormente coinvolte nella fruizione di *video streaming*.

GLOBAL VIDEO TRAFFIC SHARE			
	Application	Downstream	Upstream
1	YouTube	47.9%	25.8%
2	TikTok	16.1%	7.8%
3	Facebook Video	14.6%	43.1%
4	Instagram	12.1%	12.0%
5	Netflix	4.3%	2.4%

Figura 5.1: Principali applicazioni mobili utilizzate per la diffusione di contenuti video

Fonte Figura 5.1 e riferimenti [6].

I parametri che, durante la fruizione dei contenuti, ricoprono una rilevanza preponderante sono legati ai consumi energetici e alla quantità di traffico network impiegato.

Attraverso le analisi svolte nel Capitolo 3, è stato possibile constatare che il formato di compressione video AV1 garantisce un sostanziale risparmio nel *rate* di codifica. Il focus del corrente capito verte, invece, sui **consumi energetici** e sul complessivo **utilizzo di CPU e memoria** necessari per la riproduzione dei contenuti in ambiente mobile.

5.2 Scelte progettuali per l'analisi

5.2.1 Caratteristiche dispositivi mobili

Per l'esecuzione dei test sono stati utilizzati due smartphone *Android-based* prodotti e rilasciati da Samsung Electronics:

1) Samsung Galaxy A31

- Data di rilascio: aprile 2020
- Architettura CPU: ARMv8
- Core: 8 @ 1700MHz - 2000MHz
- Versione SO e SDK: Q (10) - 29
- Decodificatori *hardware-accelerated*: H.263, MPEG-4, AVC, HEVC, VP8, VP9

2) Samsung Galaxy S5

- Data di rilascio: aprile 2014
- Architettura CPU: ARMv7
- Core: 4 @ 300MHz - 2457MHz
- Versione SO e SDK: Marshmallow (6.0.1) - 23
- Decodificatori *hardware-accelerated*: H.263, MPEG-4, AVC, VP8

5.2.2 Sequenze video testate

Anche in questo caso, i contenuti analizzati sono stati estrapolati da sequenze disponibili su Xiph.org [40] e Sjtu.edu.cn [41].

Nella Tabella 5.1 sono riportate le caratteristiche dei brevi filmati testati in ambiente mobile.

Nome sequenza	Risoluzione	Frame-rate	Durata
Elephants dream	704 x 480	24 Hz	125 sec
Big buck bunny	854 x 480		
Marathon_{25fps}	1280 x 720	25 Hz	76 sec
Marathon_{30fps}		30 Hz	90 sec
Park joy		50 Hz	
Sunflower	1920 x 1080	25 Hz	80 sec
Sintel Tailer	from 144p up to 1080p	24 Hz	52 sec

Tabella 5.1: Sequenze video testate su dispositivi mobili

5.2.3 Materiale utilizzato

Per effettuare le analisi sono stati impiegati i seguenti strumenti:

- **FFmpeg** [42]: utilizzato per eseguire le varie codifiche, sfruttando le librerie x264, x265 e aom.
- **MP4Box** [49]: utilizzato per convertire i contenuti multimediali in file compatibili con DASH.
- **pAV0NE**: lettore multimediale sviluppato per offrire supporto ai contenuti in formato AV1, sfruttando l'implementazione del decoder **gav1**. I dettagli dell'applicazione sono presentati nel Capitolo 4.
- **VLC media player** [50]: lettore multimediale sviluppato da VideoLAN. Consente la riproduzione della maggior parte dei file video, audio e *network streaming*. Grazie all'implementazione del decoder **dav1d**, offre supporto anche per la riproduzione di contenuti in formato AV1.

- **Android Debug Bridge** [51]: *tool command-line* che consente di connettersi e comunicare con dispositivi mobili utilizzando la rete Wi-Fi. È stato adoperato per monitorare le prestazioni ottenute durante la riproduzione dei contenuti video.

5.3 Risultati ottenuti

Nell'analisi prestazionale sono stati monitorati i consumi energetici e la complessità computazionale richiesti dall'applicazione pAV0NE per la decodifica e la riproduzione sia di contenuti video memorizzati localmente sia di contenuti video in streaming adattivo DASH. In aggiunta, sono state analizzate le performance ottenute dal lettore multimediale VLC durante la riproduzione di contenuti locali.

È bene sottolineare che i dispositivi mobili presentano, tipicamente, del supporto hardware per eseguire la codifica e la decodifica di contenuti nei più comuni formati di compressione, come AVC e HEVC, garantendo una migliore gestione dell'energia.

I dati ricavati dalle analisi sono stati memorizzati nella repository GitHub [52] e di seguito sono presentati i risultati più rilevanti.

5.3.1 Standard Definition (480p)

Le prime sequenze analizzate sono due brevi cortometraggi d'animazione aventi una *standard definition* (SD) e *frame-rate* pari a $24fps$.

Nella Figura 5.2 sono rappresentati i consumi energetici, espressi in termini di nAh, necessari per la riproduzione del filmato ***big buck bunny***. I dati sono relativi al dispositivo Samsung Galaxy A31, che presenta supporto hardware per la decodifica di contenuti sia in formato AVC sia in formato HEVC.

Dal diagramma è possibile notare che il contenuto nei formati AVC e HEVC richiede consumi energetici simili e nettamente inferiori rispetto al formato AV1, che necessita di un quantitativo d'energia quasi 3 volte superiore.

Nella Figura 5.3 sono, invece, riportate le percentuali d'utilizzo medio della CPU durante la riproduzione dei contenuti. È possibile notare come l'applicazione pAV0NE utilizza, per la riproduzione del contenuto AV1, un carico computazionale che è superiore di 4 punti percentuali

rispetto agli altri formati. Mentre, VLC, per il contenuto in formato AV1, impiega una percentuale d'utilizzo che è superiore di 6 punti percentuali rispetto agli altri formati.

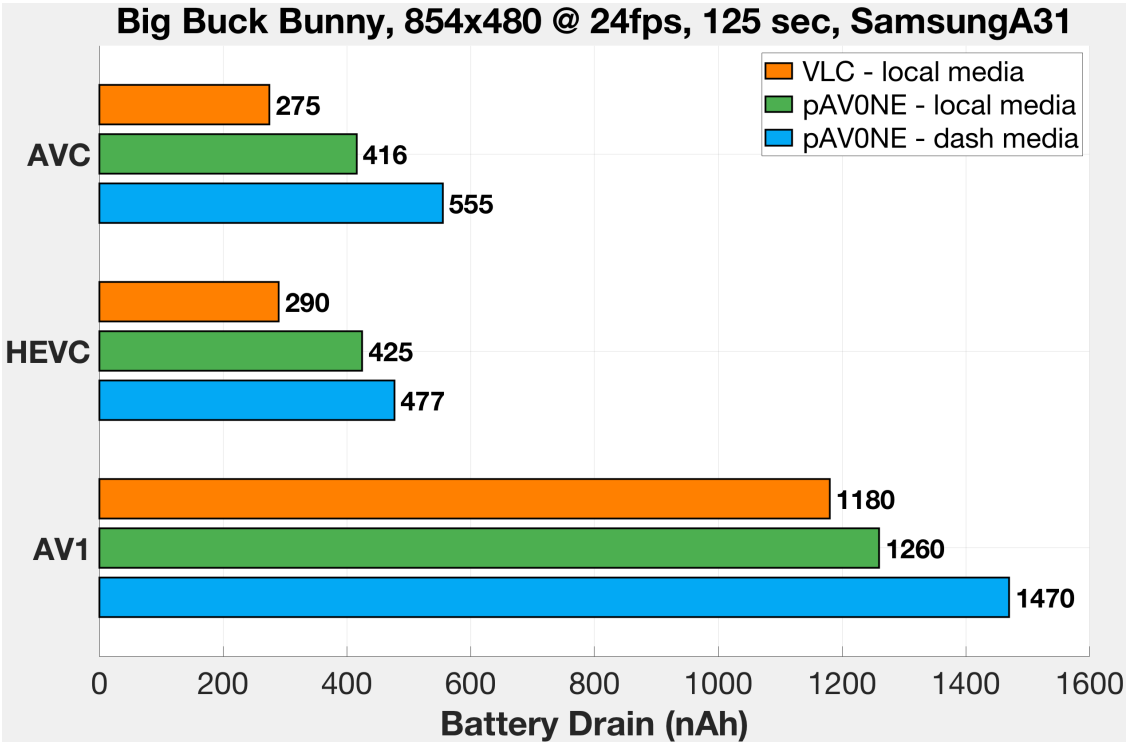


Figura 5.2: Big buck bunny (480p), consumi energetici, SamsungA31

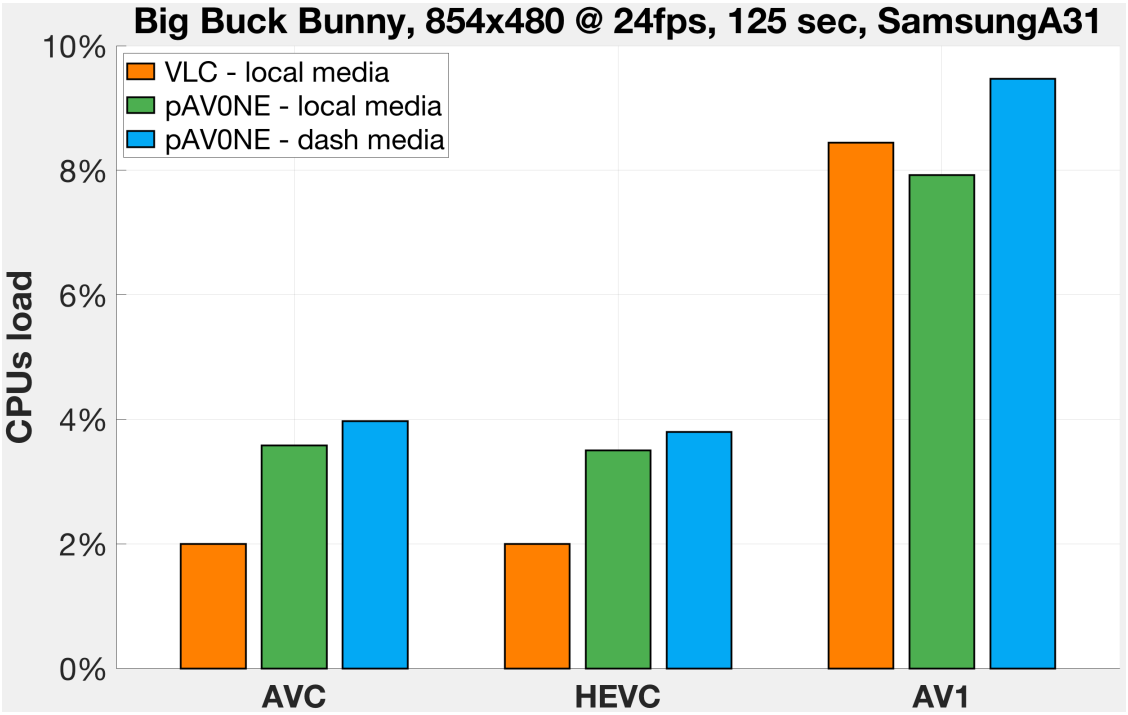


Figura 5.3: Big buck bunny (480p), carico medio CPUs, SamsungA31

Nelle Figure 5.4 e 5.5 sono, invece, rappresentati i consumi energetici, espressi in termini di *mAh*, e le percentuali d'utilizzo medio della CPU per la riproduzione della sequenza sul dispositivo Samsung Galaxy S5.

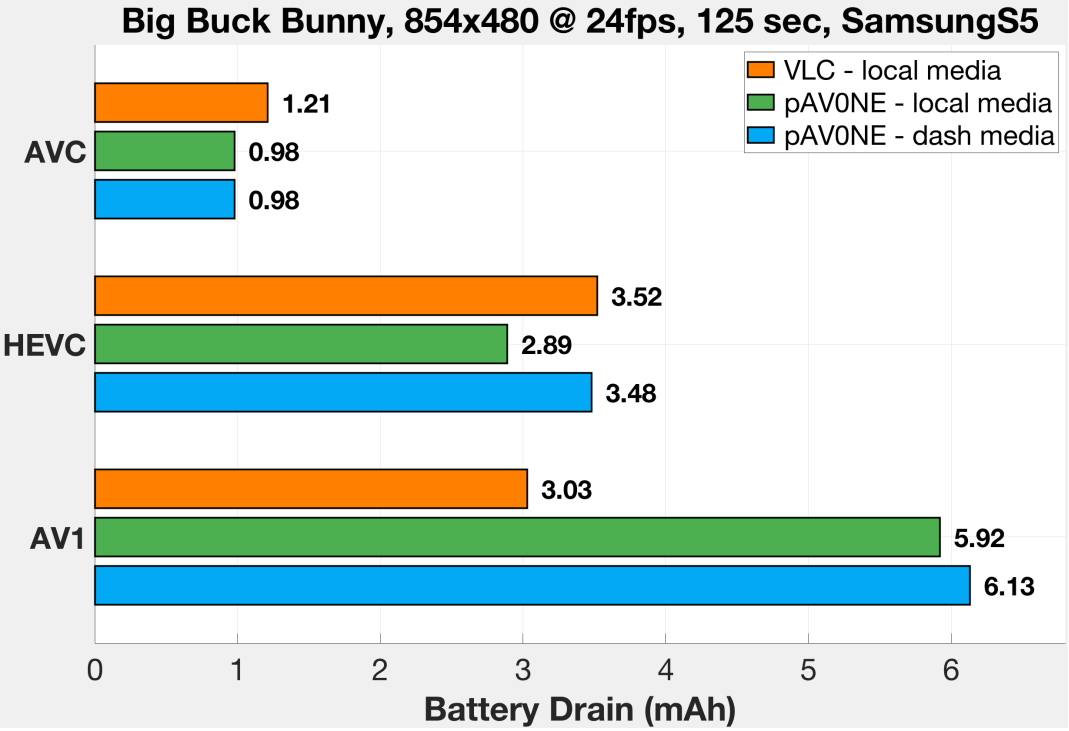


Figura 5.4: Big buck bunny (480p), consumi energetici, SamsungS5

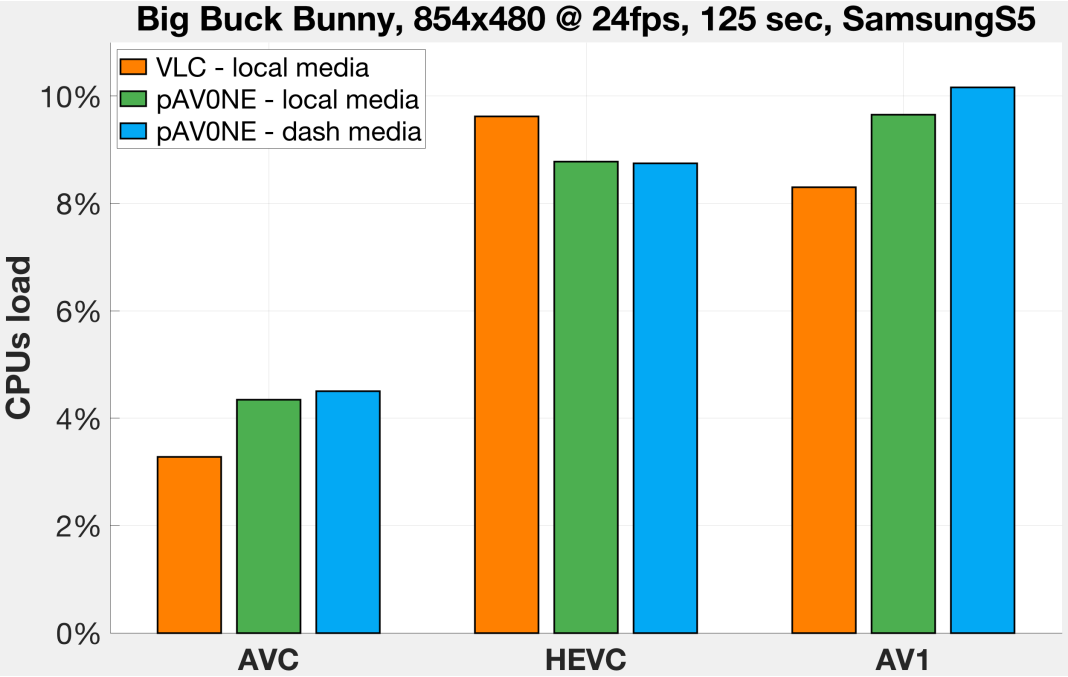


Figura 5.5: Big buck bunny (480p), carico medio CPUs, SamsungS5

Il secondo dispositivo adoperato supporta via hardware solamente la decodifica di contenuti in formato AVC. Difatti, è possibile osservare che, per riprodurre il contenuto nei formati AV1 e HEVC, il lettore multimediale VLC impiega un quantitativo di energia simile. Inoltre, il decoder AV1 implementato da VLC (dav1d) è ben ottimizzato per tale tipologia di dispositivo e richiede dei consumi energetici che sono circa la metà rispetto a quelli del decoder implementato da pAV0NE (gav1).

Nella Tabella 5.2 sono, invece, riportati i consumi energetici e le percentuali d'utilizzo medio di CPU e memoria relativi alla decodifica e riproduzione della sequenza ***elephants dream*** sul dispositivo Samsung A31. Il filmato presenta una risoluzione verticale leggermente inferiore rispetto a quello studiato in precedenza (704 contro 854), ma le considerazioni sui dati ricavati sono simili.

Formato	**App**	**Battery drain**	**% CPU**	**% MEM**
AVC	VLC	278 nAh	2.6	7.3
	pAV0NE	411 nAh	3.7	5.5
HEVC	VLC	272 nAh	2.55	7.4
	pAV0NE	418 nAh	3.45	5.5
AV1	VLC	1270 nAh	11.27	7.5
	pAV0NE	1560 nAh	10.23	6.1

Tabella 5.2: Elephants dream (480p), consumi in SamsungA31

5.3.2 High Definition (720p)

Nella Figura 5.6 e nella Tabella 5.3 sono riportati rispettivamente i consumi energetici, in termini di *nAh*, e le percentuali d'utilizzo medio di CPU e memoria per la decodifica e la riproduzione della sequenza video ***marathon*** (1280 x 720 @ 30fps) sul dispositivo Samsung A31.

Il consumo della batteria per il contenuto nei formati AVC e HEVC è sostanzialmente invariato rispetto le sequenze a *standard definition* precedentemente studiate. Mentre, i consumi energetici richiesti dal contenuto in formato AV1 sono raddoppiati in VLC e triplicati in pAV0NE, finendo col risultare circa 10 volte superiori rispetto agli altri formati.

Anche la differenza nella complessità computazionale delle operazioni inizia ad essere significativa. Il contenuto in formato AV1 impiega una

percentuale d'uso medio della CPU che è circa del 17% in pAV0NE e del 24% in VLC, contro il 3-4% richiesto dai formati AVC e HEVC.

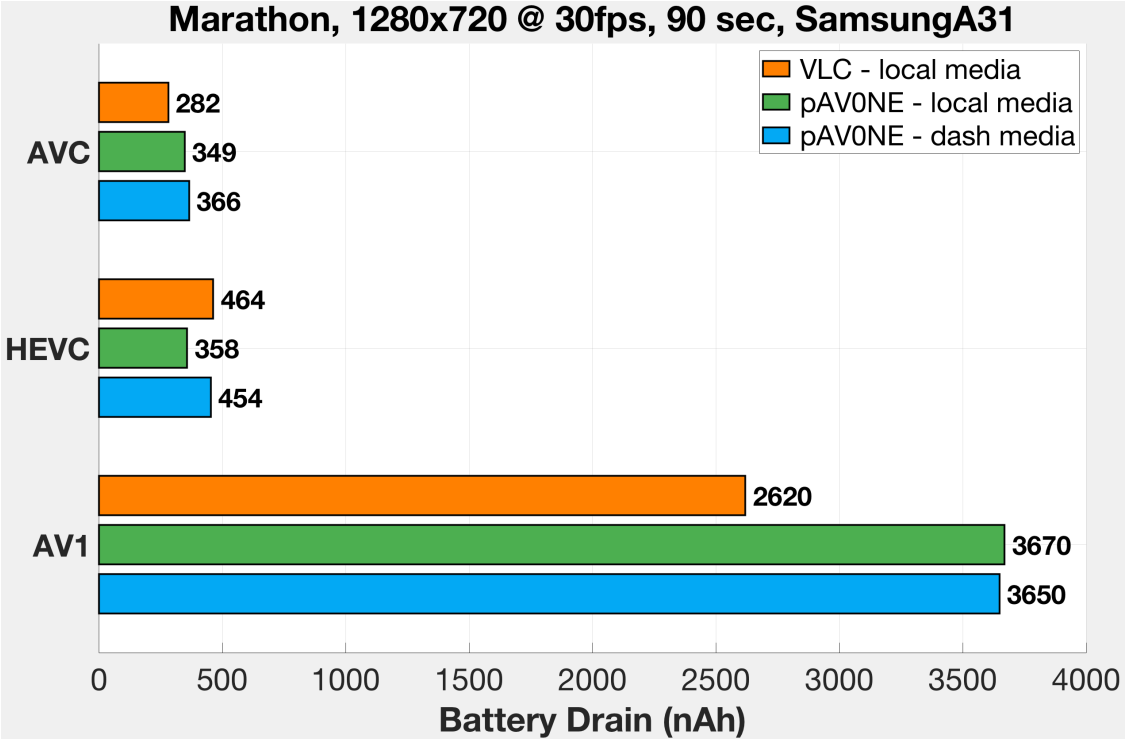


Figura 5.6: Marathon (720p), consumi energetici, SamsungA31

App	Formato	Tipo contenuto	% CPU	% MEM
VLC	AVC	locale	3.14	7.4
	HEVC		3.1	7.75
	AV1		23.77	7.5
pAV0NE	AVC	locale	3.94	6.3
		DASH	4.5	6.2
	HEVC	locale	4.04	5.2
		DASH	4.34	6.0
	AV1	locale	16.78	7.2
		DASH	16.68	7.6

Tabella 5.3: Marathon (720p), percentuali utilizzo medio di CPU e memoria, SamsungA31

Nei test eseguiti sul dispositivo Samsung S5 è stata impiegata la stessa sequenza video ma con *frame-rate* (25fps) e durata inferiori. Nelle Figure 5.7 e 5.8 sono rappresentati rispettivamente i consumi energetici, in termini di *mAh*, e le percentuali d'utilizzo medio della CPU necessari per la riproduzione del contenuto sopraccitato.

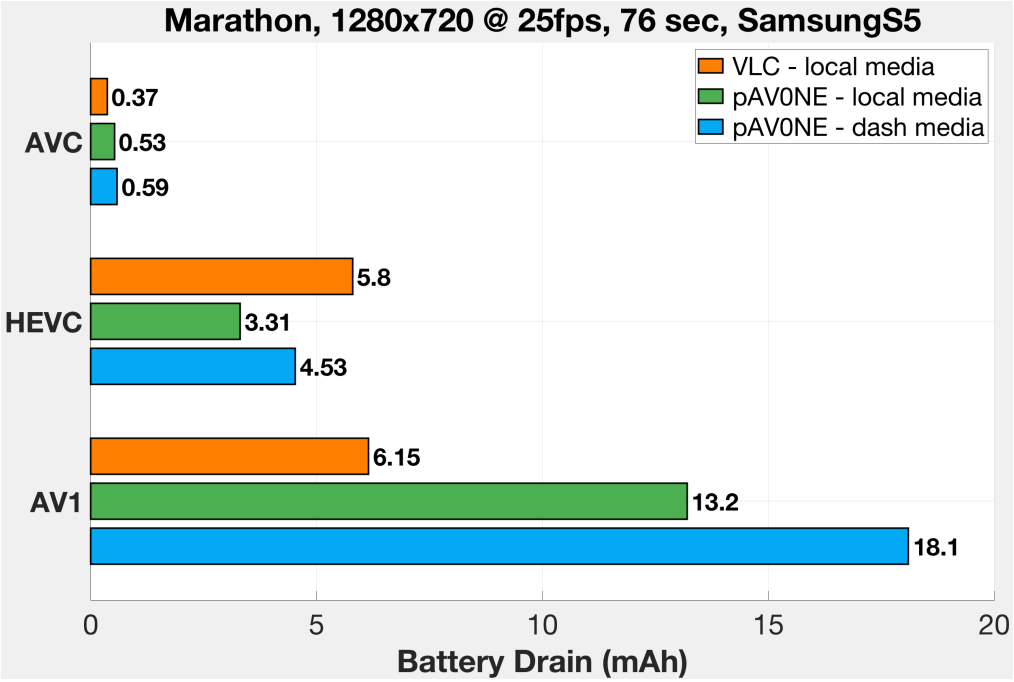


Figura 5.7: Marathon (720p), consumi energetici, SamsungS5

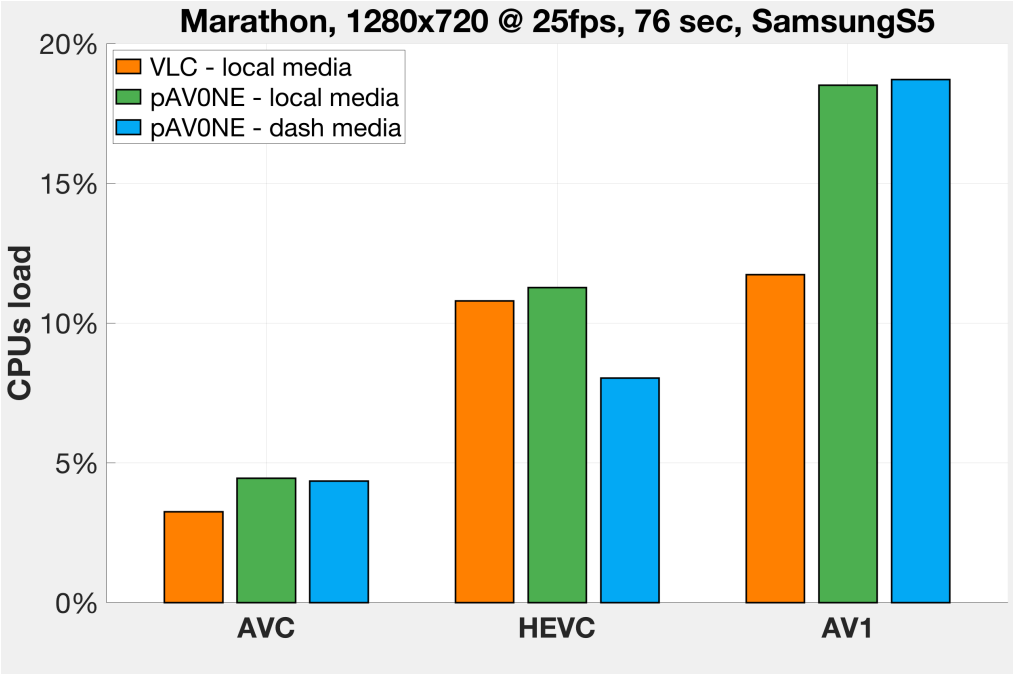


Figura 5.8: Marathon (720p), carico CPUs, SamsungS5

Dai precedenti diagrammi è possibile osservare che, utilizzando l'applicazione VLC, il consumo energetico e il carico medio della CPU per la riproduzione del contenuto in AV1 sono molto simili a quelli del contenuto in HEVC. Lo scenario è differente con l'applicazione pAV0NE, in cui il contenuto in formato AV1 necessita di un quantitativo di batteria elevato, specialmente per la riproduzione di *streaming DASH*.

Infine, nella Tabella 5.4 vengono riportati i dati relativi ai consumi energetici e alle percentuali di CPU e memoria impiegati per la riproduzione della sequenza ***park joy*** (1280 x 720 @ 60fps) sul dispositivo Samsung A31. Il contenuto ha un *frame-rate* piuttosto elevato ed il consumo richiesto dal formato AV1 è notevole, risultando circa 10 volte superiore rispetto a quello degli altri formati.

Formato	**App**	**Battery drain**	**% CPU**	**% MEM**
AVC	VLC	454 nAh	5.1	7.56
	pAV0NE	653 nAh	5.3	6.3
HEVC	VLC	412 nAh	4.9	7.8
	pAV0NE	463 nAh	5.25	5.9
AV1	VLC	3810 nAh	31.5	8.22
	pAV0NE	5290 nAh	20.03	6.9

Tabella 5.4: Park joy (720p), consumi su SamsungA31

5.3.3 Full HD (1080p)

Per testare le sequenze a risoluzione *Full HD* è stato impiegato solamente il dispositivo Samsung A31, in quanto decodificare un video Full HD in formato AV1 richiede una complessità computazionale eccessivamente elevata e non compatibile con il dispositivo Samsung S5 (riproduzione del contenuto non fluida).

Nella Figura 5.9 sono rappresentati i consumi energetici richiesti dalla sequenza ***sunflower*** (1920 x 1080 @ 25fps). La quantità di energia impiegata per la riproduzione del contenuto in formato AV1 è significativamente più elevata rispetto agli altri formati, risultando essere circa 20 volte superiore in VLC e 15 volte superiore in pAV0NE.

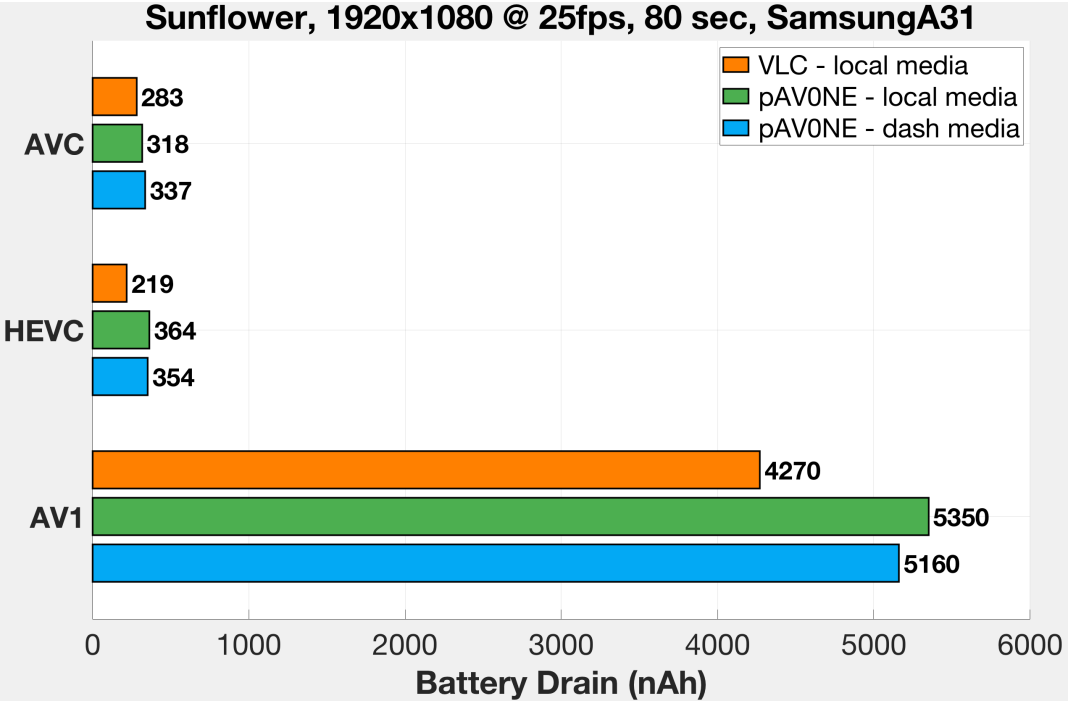


Figura 5.9: Sunflower (1080p), consumi energetici, SamsungA31

Ponendo a confronto le due applicazioni, è possibile constatare che VLC richiede dei consumi energetici leggermente inferiori, ma, così come rappresentato nella Figura 5.10, impiega, per la riproduzione del contenuto in formato AV1, una percentuale d'utilizzo medio della CPU che è superiore del 10% rispetto a pAV0NE.

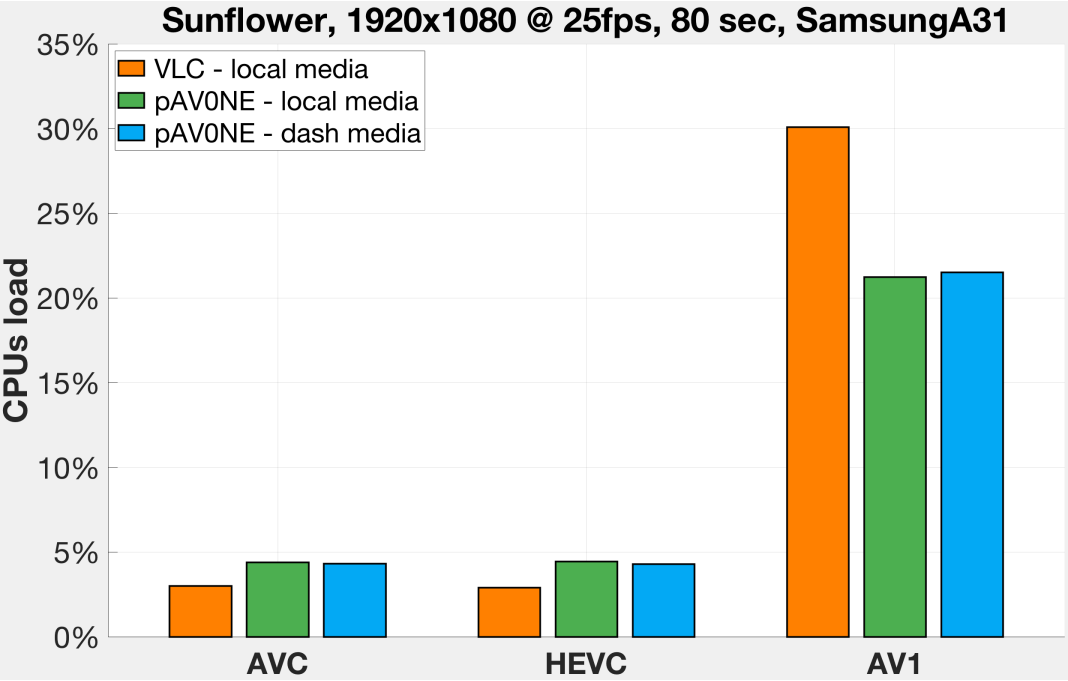


Figura 5.10: Sunflower (1080p), carico CPUs, SamsungA31

5.3.4 Sintel Trailer

La sequenza ***sintel trailer*** è stata utilizzata per monitorare i consumi energetici e l'impiego medio della CPU richiesti dall'applicazione **pAV0NE** durante la decodificare e la riproduzione del contenuto a differenti risoluzioni. Per effettuare i test è stato adoperato il dispositivo Samsung Galaxy A31 con sequenze memorizzate localmente.

Nella Figura 5.11 sono riportati i consumi energetici, in termini di µAh, necessari per la riproduzione dei contenuti. Il quantitativo d'energia richiesto dai formati AVC e HEVC è simile per tutte le risoluzioni, mentre il contenuto in formato AV1 impiega una quantità di energia che cresce in maniera esponenziale all'aumentare della risoluzione, risultando addirittura 10 volte superiore per il contenuto in Full HD (1080p).

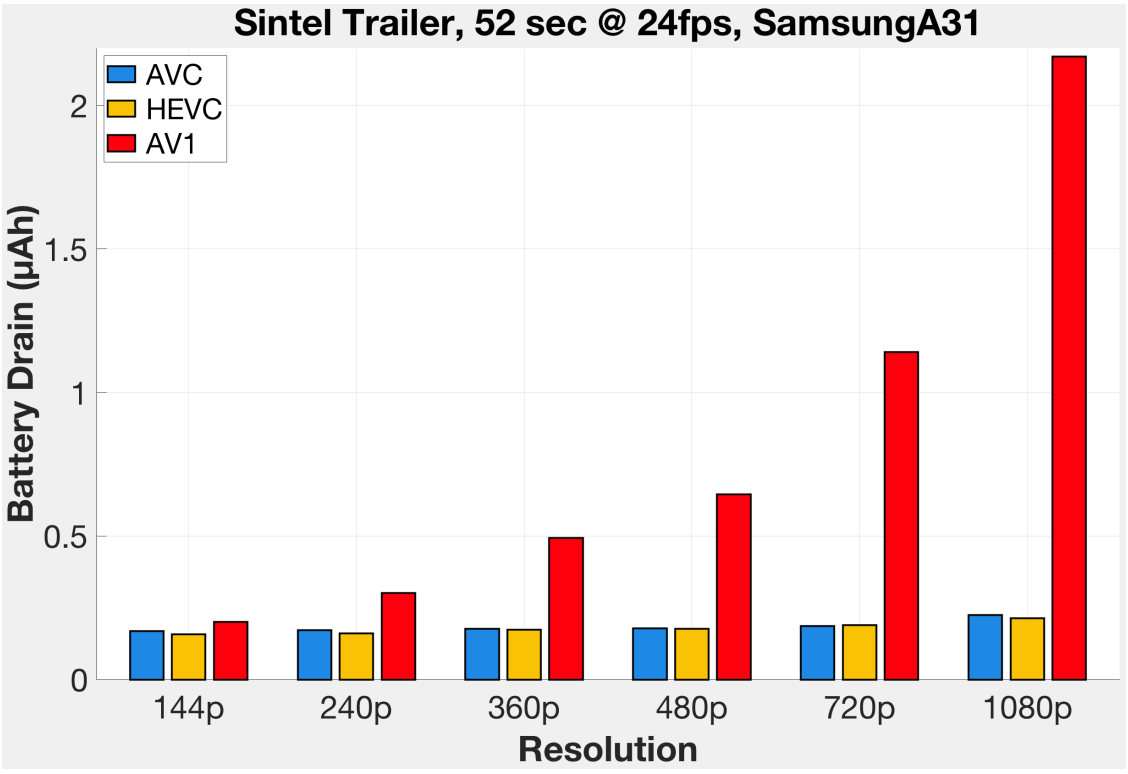


Figura 5.11: Sintel Trailer, consumi energetici, SamsungA31

In ultima analisi, nella Figura 5.12 sono rappresentate le percentuali d'utilizzo medio della CPU per la riproduzione dei contenuti. È possibile osservare che il carico medio della CPU rimane all'incirca costante durante la riproduzione di tutti i contenuti in formato AVC e HEVC, mentre per i contenuti in formato AV1 la complessità computazionale cresce linearmente all'aumentare della risoluzione.

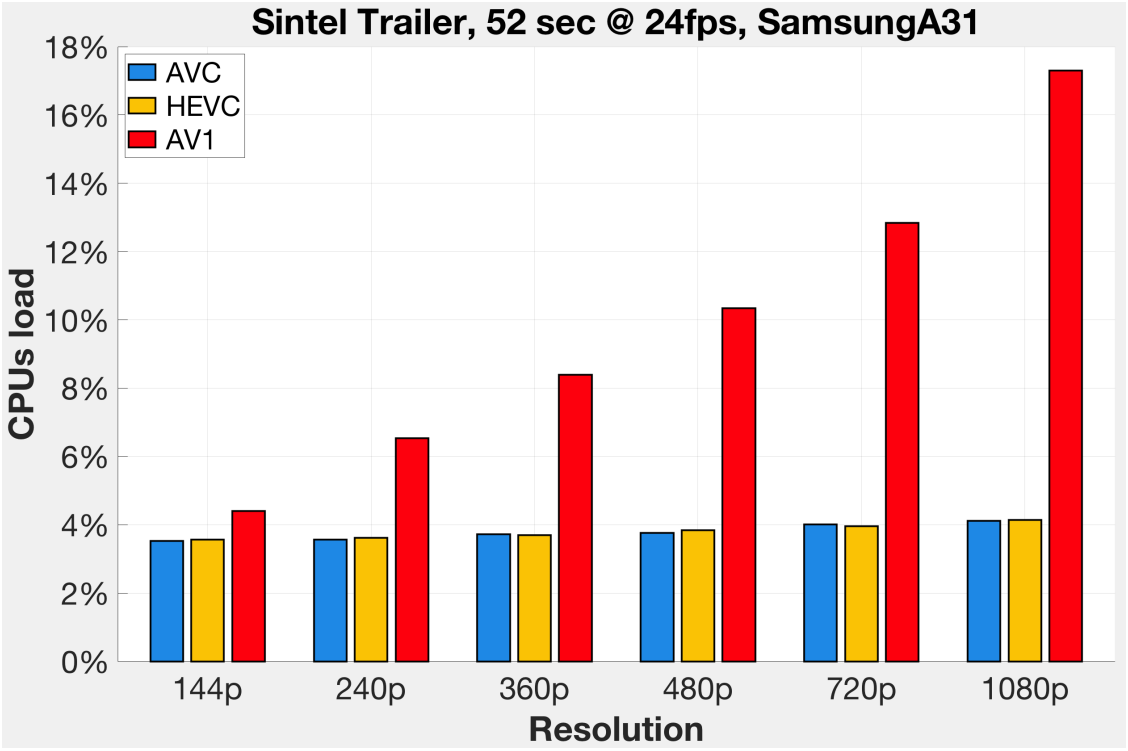


Figura 5.12: Sintel Trailer, carico CPUs, SamsungA31

Capitolo 6

Conclusioni

Gli obiettivi del presente lavoro sono stati la realizzazione di uno strumento in grado di offrire supporto della compressione video AV1 nei dispositivi mobili e la valutazione delle prestazioni del nuovo formato di codifica sia in ambiente mobile sia in ambiente desktop.

Attraverso le analisi eseguite in ambiente desktop, è stato possibile constatare che il codificatore AV1 sviluppato dall'*Alliance for Open Media* (AOM) garantisce, a parità di valutazioni qualitative in termini di VMAF e PSNR, un considerevole risparmio nel *rate* di codifica, utilizzando mediamente il 50% di bit in meno rispetto ai codificatori AVC ed il 30% di bit in meno rispetto ai codificatori HEVC. L'efficienza della compressione richiede, tuttavia, un costo computazionale significativo. Difatti, i codificatori AV1 impiegano, per eseguire le codifiche, un tempo che mediamente è 5 volte superiore rispetto ai codificatori HEVC e 30 volte superiore rispetto ai codificatori AVC. Tale inconveniente è risolvibile utilizzando piattaforme in grado di eseguire le codifiche via hardware, anche se il supporto per il formato AV1 al momento è piuttosto limitato. Lo scenario è differente per le implementazioni dei decodificatori, infatti i decoder DAV1D e GAV1 ottengono delle ottime prestazioni e, in alcuni casi, risultano perfino più veloci dei più comini decoder HEVC e AVC.

Per monitorare i consumi energetici necessari per la decodifica e la riproduzione dei contenuti video in ambiente mobile, è stata sviluppata l'applicazione pAV0NE, un lettore multimediale per Android in grado di supportare il nuovo formato di compressione video. Dall'analisi prestazionale, è emerso che i contenuti in formato AV1 necessitano di un consumo energetico considerevole, che cresce in maniera esponenziale

all'aumentare della risoluzione. Entrando più nel dettaglio, il quantitativo d'energia impiegato da un contenuto in formato AV1 a risoluzione Full HD è 10 volte superiore rispetto ai formati AVC e HEVC.

In ultima analisi, il formato di compressione video AV1 garantisce un notevole risparmio nel rate di codifica, tuttavia, a causa degli eccessivi consumi energetici richiesti dalle implementazioni attualmente disponibili in ambente mobile, per poter assistere ad una concreta adozione del nuovo formato su larga scala bisognerà attendere che i dispositivi vengano equipaggiati con *chipset* che consentano di decodificare i contenuti in maniera *hardware-accelerated*.

Appendice A

Shell script e programmi C di supporto

A.1 Esempio script codifica

```
#!/bin/bash
# BIG BUCK BUNNY - LIBAOM - ENCODING (with ffmpeg)
codec="av1"
encoder_name="AOM"
org_seq_name="big_buck_bunny_480p.yuv"
name="bunny"
w=704
h=480
n=300
fps=24
dest_dir="${encoder_name}_${name}"
mkdir ${dest_dir}
for i in {100..1100..50}
    do
(/usr/bin/time -o ${dest_dir}/enctime_${i}.txt -f "%U
   %S %E" ffmpeg -y -f rawvideo -vcodec rawvideo
   -pix_fmt yuv420p -s ${w}x${h} -r ${fps} -i
   ${org_seq_name} -vcodec ${codec} -b:v ${i}k
   -cpu-used 3 -enable-intrabc true -tune psnr -f
   mpeg2video ${dest_dir}/${name}_${i}.${codec}) &>
   ${dest_dir}/encdata_${i}.txt
    done
```

A.2 Computazione PSNR

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

int main(int argc, char **argv) {
        if (argc!=6) {
                printf("Use: %s file1.yuv file2.yuv w h
                   framenum\n",argv[0]); exit(1);
        }
        FILE *f1;
        FILE *f2;
        if ((f1=fopen(argv[1],"rb"))==NULL) {
           printf("Error opening file\n"); exit(1); }
        if ((f2=fopen(argv[2],"rb"))==NULL) {
           printf("Error opening file\n"); exit(1); }
        int w=atoi(argv[3]);
        int h=atoi(argv[4]);
        int frnum=atoi(argv[5]);
        int frsize=w*h+w*h/2;

        unsigned char *buf1=(unsigned char
           *)malloc(sizeof(unsigned char)*w);
        if (buf1==NULL) { printf("Not enough
           memory\n"); exit(1); }
        unsigned char *buf2=(unsigned char
           *)malloc(sizeof(unsigned char)*w);
        if (buf2==NULL) { printf("Not enough
           memory\n"); exit(1); }

        double cum_psnr=0.0;
        int fr;
        int j,i;
        for (fr=0; fr<frnum; fr++) {
                fseek(f1, fr*frsize, SEEK_SET);
                fseek(f2, fr*frsize, SEEK_SET);
                double msetot=0.0;
                int cnt=0;
                for (j=0; j<h; j++) {
```

```
                        fread(buf1, w, 1, f1);
                        fread(buf2, w, 1, f2);
                        for (i=0; i<w; i++) {
                                int v1 = (unsigned
                                   int)(unsigned
                                   char)buf1[i];
                                int v2 = (unsigned
                                   int)(unsigned
                                   char)buf2[i];
                                msetot += (double)
                                   (v1-v2)*(v1-v2);
                                cnt++;
                        }
                }
                double mse=msetot/cnt;
                double psnr=10.0*log10(255.0*255.0/mse);
                cum_psnr+=psnr;
        }
        double seq_psnr=cum_psnr/fr;
        printf("%.2f\n",seq_psnr);

        free(buf1);
        free(buf2);
        fclose(f1);
        fclose(f2);
        return 0;
}
```

A.3 Esempio script decodifica e raccolta dati

```
#!/bin/bash
# DECODING (with ffmpeg) AND GET DATA
# BIG BUCK BUNNY - LIBAOM
codec="av1"
encoder_name="AOM"
seq_name="bunny"
org_seq_name="big_buck_bunny_480p.yuv"
w=704
```

```
h=480
n=300
curr_dir=pwd
dest_dir="${encoder_name}_${seq_name}"
output_file="big_buck_bunny_480p.csv"
## Set param with your vmaf path
vmaf_dir="/home/ubuntu/vmaf"
## Set param with your dav1d path
dav1d_path="/home/ubuntu/dav1d/build/tools/dav1d"
## Set param with your SVT-AV1 path
svtav1_path="/home/ubuntu/svtav1"
## Set param with your GAV1 path
gav1_path="/home/ubuntu/libgav1/build/gav1_decode"
### Necessary to compute VMAF value
cd ${vmaf_dir} && source .venv/bin/activate
cd ${curr_dir}
for i in {100..1100..50} do
# DECODING
(/usr/bin/time -o ${dest_dir}/dectime_${i}.txt -f "%U
   %S %E" ffmpeg -y -i
   ${dest_dir}/${seq_name}_${i}.${codec} -vcodec
   rawvideo -f rawvideo
   ${dest_dir}/${seq_name}_${i}.yuv)
# GET BITRATE
bitrate='grep 'kbits/s' ${dest_dir}/encdata_${i}.txt |
   tail -c 50 | awk -F '=' '{print $2}' | awk -F 'k'
   '{print $1}' | sed -r 's/ //g''
# GET ENCODING TIME
enctime='./get_time ${dest_dir}/enctime_${i}.txt min |
   grep 'User+System_time' | awk -F ':' '{print $2}''
wallclock='./get_time ${dest_dir}/enctime_${i}.txt min
   | grep 'Wall_clock' | awk -F ':' '{print $2}''
# GET DECODING TIME
dectime='./get_time ${dest_dir}/dectime_${i}.txt sec |
   grep 'User+System_time' | awk -F ':' '{print $2}''
# GET PSNR
psnr='./psnr_video ${org_seq_name}
   ${dest_dir}/${seq_name}_${i}.yuv ${w} ${h} ${n}'
# GET VMAF
cd ${vmaf_dir}
```

```
(PYTHONPATH=python
   ${vmaf_dir}/python/vmaf/script/run_vmaf.py \
yuv420p ${w} ${h} \
${curr_dir}/${org_seq_name} \
${curr_dir}/${dest_dir}/${seq_name}_${i}.yuv \
--out-fmt json) &> ${curr_dir}/vmaf.json
cd ${curr_dir}
vmaf='tail -4 vmaf.json | grep 'score' | awk -F ': '
   '{print $2}' | sed -r 's/.{13}$//''
# GET DAV1D DECODING TIME
(/usr/bin/time -o ${dest_dir}/dectime_dav1d.txt -f "%U
   %S %E" ${dav1d_path} -i
   ${dest_dir}/${seq_name}_${i}.${codec} -o /dev/null)
dav1d_dectime='./get_time
   ${dest_dir}/dectime_dav1d.txt sec | grep
   'User+System_time' | awk -F ':' '{print $2}''
# GET SVT-AV1 DECODING TIME
cd ${svtav1_path}
/usr/bin/time -o
   ${curr_dir}/${dest_dir}/dectime_svtav1.txt -f "%U
   %S %E" ./SvtAv1DecApp -i
   ${curr_dir}/${dest_dir}/${seq_name}_${i}.${codec}
   -o /dev/null
cd ${curr_dir}
svtav1_dectime='./get_time
   ${dest_dir}/dectime_svtav1.txt sec | grep
   'User+System_time' | awk -F ':' '{print $2}''
# GET GAV1 DECODING TIME
output="output.ivf"
ffmpeg -y -i ${dest_dir}/${seq_name}_${i}.${codec}
   -codec copy ${dest_dir}/${output}
/usr/bin/time -o ${dest_dir}/dectime_gav1.txt -f "%U
   %S %E" ${gav1_path} ${dest_dir}/${output}
gav1_dectime='./get_time ${dest_dir}/dectime_gav1.txt
   sec | grep 'User+System_time' | awk -F ':' '{print
   $2}''
# WRITE DATA
echo "${encoder_name};${bitrate};${psnr};${vmaf};
${wallclock};${enctime};${dectime};${dav1d_dectime};
${svtav1_dectime};${gav1_dectime}">>${output_file}
done
```

Bibliografia

[1] Cisco Annual Internet Report (2018–2023) White Paper, 9 Marzo 2020, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] A. Collins, M Conley, The Ultimate Guide to Video Marketing, https://blog.hubspot.com/marketing/video-marketing

[3] Think with Google, "The latest video trends: Where your audience is watching", Aprile 2016, https://www.thinkwithgoogle.com/marketing-strategies/video/video-trends-where-audience-watching

[4] Alexa, The top 500 sites on the web, https://www.alexa.com/topsites

[5] YouTube Official Blog, YouTube for Press, https://blog.youtube/press

[6] Sandvine, The Mobile Internet Phenomena Report, Maggio 2021, https://www.sandvine.com/phenomena

[7] Recode, "An explosion of online video could triple bandwidth consumption again in the next five years", https://www.vox.com/2017/6/8/15757594/future-internet-traffic-watch-live-video-facebook-google-netflix

[8] I.E. Richardson, "Video Codec Design: Developing Image and Video Compression Systems", Wiley, 2002, ISBN: 9780471485537, https://books.google.it/books?id=8jxbbRKVbkIC

[9] H.R. Wu, K.R. Rao, "Digital Video Image Quality and Perceptual Coding", CRC Press, 2017, ISBN: 9781351836821, https://books.google.it/books?id=NApEDwAAQBAJ

[10] F. Rayar, "ImageNet MPEG-7 Visual Descriptors", Technical Report, 2017, https://www.researchgate.net/publication/313212175_ImageNet_MPEG-7_Visual_Descriptors_-_Technical_Report

[11] M. Santamaria, M. Trujillo, "A Comparison of Block-Matching Motion Estimation Algorithms", 7CCC 2012, Medellín - Colombia,

https://www.slideshare.net/mmv-lab-univalle/slides7-ccc-v8

[12] C.Fogg, D.J.LeGall, J.L.Mitchell, W.B.Pennebaker, "MPEG Video Compression Standard", Springer US, 2007, ISBN: 9780306469831, https://books.google.it/books?id=NApEDwAAQBAJ

[13] P. Aimonen, "A sequence of Intra-coded, Predicted and Bi-predicted frames in a compressed video sequence", 30-set-2009, https://it.wikipedia.org/wiki/File:I_P_and_B_frames.svg#globalusage

[14] ITU-T (1988), "H.261 : Video codec for audiovisual services at p x 384 kbit/s - Recommendation H.261 (11/88)", https://www.itu.int/rec/T-REC-H.261-198811-S/en

[15] I.E. Richardson, "The H.264 Advanced Video Compression Standard", Wiley, 2010, ISBN: 9780470516928, 2nd edition.

[16] VP8, from Wikipedia, the free encyclopedia, ultima visita il 14-06-2021, https://en.wikipedia.org/wiki/VP8

[17] Free Software Foundation, "Open letter to Google: free VP8, and use it on YouTube", https://www.fsf.org/blogs/community/google-free-on2-vp8-for-youtube

[18] V. Sze, M. Budagavi, G.J. Sullivan, "High Efficiency Video Coding (HEVC): Algorithms and Architectures", Springer International Publishing, 2014, ISBN: 9783319068954, https://books.google.it/books?id=3WhYBAAAQBAJ

[19] HEVC licences: A crisis, the causes and a solution, https://blog.chiariglione.org/a-crisis-the-causes-and-a-solution

[20] S. Zimmerman, "Google's Royalty-Free Answer to HEVC: A Look at AV1 and the Future of Video Codecs", 15 Maggio 2017, https://web.archive.org/web/20170614042710/https://www.xda-developers.com/av1-future-video-codecs-google-hevc

[21] AV1, from Wikipedia, the free encyclopedia, ultima visita il 04-06-2021, https://en.wikipedia.org/wiki/AV1

[22] P. de Rivaz, J. Haughton, "AV1 Bitstream & Decoding Process Specification", ultima modifica il 08-01-2019, https://aomediacodec.github.io/av1-spec/av1-spec.pdf

[23] Facebook, AV1 Test Page, https://www.facebook.com/AV1-Test-Page-330716120785217

[24] Netflix Technology Blog, "Netflix Now Streaming AV1 on Android", 05 Feb 2020, https://netflixtechblog.com/netflix-now-streaming-av1-on-android-d5264a515202

[25] Alliance for Open Media, Membership, http://aomedia.org/membership/members

[26] Alliance for Open Media, "IBC 2020: Panel Discussion with Amazon, Facebook, Google, Intel, Netflix, and Tencent on AV1 Commercial Readiness", https://aomedia.org/in%20the%20news/aomedia-in-the-news-ibc-2020-panel-discussion
[27] VMAF - Video Multi-Method Assessment Fusion, https://github.com/Netflix/vmaf
[28] Video Multimethod Assessment Fusion, from Wikipedia, the free encyclopedia, ultima visita il 15-05-2021, https://en.wikipedia.org/wiki/Video_Multimethod_Assessment_Fusion
[29] Netflix Technology Blog, "VMAF: The Journey Continues", 25 Oct 2018, https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12
[30] AOM, AV1 Codec Library, https://aomedia.googlesource.com/aom
[31] Scalable Video Technology for AV1, SVT-AV1 Encoder and Decoder, https://github.com/AOMediaCodec/SVT-AV1
[32] RAV1E, Rust AV1 Encoder, https://github.com/xiph/rav1e
[33] Aurora1, an AV1 encoder, https://visionular.com/products/aurora1-av1-encoder
[34] MSU codec study report, encoders performance analysis, 2019, https://visionular.cn/Works/2019-msu-study-aurora1-top-av1-encoder
[35] EVE-AV1, making the new standard for video compression a reality, https://www.twoorioles.com/eve-av1
[36] Cisco's AV1, AV1 comes to Webex, https://blog.webex.com/engineering/the-av1-video-codec-comes-to-webex
[37] DAV1D, an AV1 cross-platform decoder, https://code.videolan.org/videolan/dav1d
[38] GAV1, an AV1 decoder, https://chromium.googlesource.com/codecs/libgav1
[39] AV1 Video Extension, Microsoft Corporation extension to play videos encoded using the AV1 standard, https://www.microsoft.com/en-us/p/av1-video-extension/9mvzqvxjbq9v
[40] Xiph.org Video Test Media, https://media.xiph.org/video/derf
[41] SJTU Video Sequences, http://medialab.sjtu.edu.cn/web4k
[42] FFmpeg, A cross-platform solution to record, convert and stream audio and video, versione N-99801-gfbb44bc, https://www.ffmpeg.org
[43] Script utilizzati in ambiente desktop per l'esecuzione di codifiche, decodifiche e raccolta dei dati, repository GitHub, https://gith

ub.com/federicoserraino/AV1-performance-analysis/tree/main/desktop
[44] Fogli di calcolo performance in ambiente desktop, repository GitHub, https://github.com/federicoserraino/AV1-performance-analysis/tree/main/desktop/spreadsheets
[45] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP – Design Principles and Standards", Febbraio 2011, https://pdfs.semanticscholar.org/03c4/2879f25ca66a9ae906672a6e3649bb07ca03.pdf
[46] ExoPlayer, release 2.13.3 (2021-04-14), https://github.com/google/ExoPlayer
[47] Apache, The Number One HTTP Server On The Internet, https://httpd.apache.org
[48] pAV0NE, an open source multimedia player for Android, https://github.com/federicoserraino/pAV0NE
[49] MP4Box, The multimedia packager available in GPAC, release 1.1.0-DEV-rev802-gc0ea96c7b-master, https://github.com/gpac/gpac/wiki/MP4Box
[50] VLC for Android, release 1.1.0-DEV-rev802-gc0ea96c7b-master, https://www.videolan.org/vlc/download-android.html
[51] ADB, Android Debug Bridge, https://developer.android.com/studio/command-line/adb
[52] Fogli di calcolo performance in ambiente mobile, repository GitHub, https://github.com/federicoserraino/AV1-performance-analysis/tree/main/mobile/spreadsheets