Indice

Capitolo 1

Algoritmi di compressione Lossless

La compressione *lossless* è la tecnica che permette di recuperare tutta l'informazione e ricostruire i dati iniziali partendo da quelli compressi attraverso l'utilizzo di appositi algoritmi di compressione.

In particolare saranno mostrati in dettaglio la codifica di HUFFMAN e la codifica di LZW.

1.1 Codifica di Huffman

Per Codifica di Huffman si intende un algoritmo di codifica dell'entropia usato per la compressione di dati, che cerca di trovare il sistema ottimale per codificare stringhe basandosi sulla frequenza relativa di ciascun carattere.

Questa tecnica funziona creando un albero binario di simboli secondo la seguente procedura:

1. Inizializzazione: inserimento in una lista di tutti i simboli in base al numero delle occorrenze;

2. Ripetere il punto 1 finche la lista ha un solo elemento che deve ancora essere inserito;

    - Si prendono dalla lista i 2 simboli che hanno il minor numero di occorrenze, con cui viene creato un sottoalbero di Huffman, avente i due elementi come nodi e un nuovo nodo come padre;
    - Si assegna al padre la somma delle occorrenze dei due figli, in modo tale da mantenere l'ordine del sottoalbero;
    - Si eliminano i nodi figli inseriti nell'albero dalla lista;

3. Assegno una parola chiave ad ogni foglia in base al cammino dalla radice, tenendo conto che i rami di sinistra prendono il valore 0 e quelli di destra 1;

La codifica di Huffman usa un metodo specifico per scegliere la rappresentazione di ciascun simbolo creando un codice senza prefissi (cioè in cui nessuna stringa binaria di nessun simbolo è il prefisso della stringa binaria di un altro simbolo) che esprime il carattere più frequente nella maniera più breve possibile.

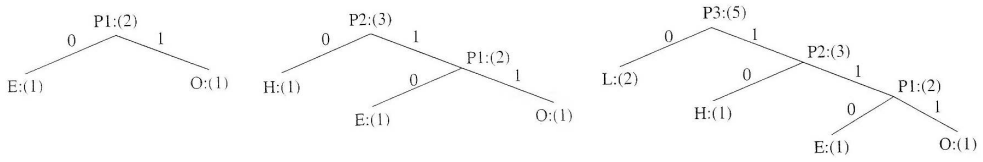

Figura 1.1: Esempio dell'algoritmo di Huffman sulla stringa "hello"

Per esempio il codice 0 assegnato a L non è un prefisso per il codice 10 di H; questa proprieta è importante per ottenere una decodifica efficiente, in quanto rimuove completamente qualsiasi ambiguità.

È stato dimostrato che la codifica di Huffman è il più efficiente sistema di compressione di questo tipo: nessun'altra mappatura di simboli in stringhe binarie può produrre un risultato più breve nel caso in cui le frequenze di simboli effettive corrispondono a quelle usate per creare il codice.

1.2 Algoritmo LZW

Questo algoritmo usa una tecnica di compressione basata su un dizionario di simboli.

La sua particolarità è quella di utilizzare una parola chiave di grandezza prefissata per rappresentare stringhe di lunghezza variabile.

Il dizionario, sia nella codifica che nella decodifica, viene aggiornato dinamicamente durante l'esecuzione dell'algoritmo attraverso l'inserimento di nuove stringhe.

LZW viene usato particolarmente nei sistemi UNIX, nelle immagini GIF e TIFF e nei modem.

La codifica dell' algoritmo viene eseguita nel modo seguente:

1. Preparazione della tabella contenente stringhe formate da un solo simbolo;

2. Inizializzazione: Lettura del primo simbolo, che sarà la nostra stringa prefisso ω;

3. Lettura del simbolo successivo(che chiameremo κ):

   - Se la stringa $\omega\kappa$ appartiene alla tabelle iniziale si pone ω=$\omega\kappa$, e si torna al punto 3;
   - Se la stringa $\omega\kappa$ non appartiene allora viene inserito $\omega\kappa$ in tabella, si pone $\omega = \kappa$, e si torna al punto 3;

La tabella sottostante rappresenta il procedimento di codifica della stringa ABABBABCABABBA.

ω	κ	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	B	6	11	ABBA
A	EOF	1		

Tabella 1.1: Tabella di codifica

Da cui si ottiene la stringa codificata: 124523461
La decodifica dell' algoritmo LZW avviene nenl modo seguente:

1. Preparazione della tabella contenente stringhe formate da un solo simbolo;
2. Lettura del primo simbolo che sarà la nostra stringa ω;
3. Si pone la stringa κ=ω;
4. Si legge il prossimo simbolo e lo si pone ω:
   - Se ω è presente nel dizionario:
     - Si codifica ω;
     - Si pone κ=simbolo precedente;
     - Si pone C= primo carattere di ω;
     - Si aggiunge κC al dizionario;
   - Se ω non e' presente nel dizionario:
     - Si pone κ=simbolo precedente;
     - Si pone C=primo carattere di ω;
     - Si codifica ωC e lo si inserisce nel dizionario;
5. Se ci sono ancora simboli da decodificare si torna al punto 2, altrimenti fine;

La tabella seguente mostra la decodifca della stringa ottenuta dalla codifica precedente: 124523461
Salla quale si ottiene la stringa iniziale: ABABBABCABABBA

ω	κ	entry/output	code	string
			1	A
			2	B
			3	C
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

Tabella 1.2: Tabella di decodifica

Capitolo 2

Algoritmi di compressione Lossy

Le tecniche di compressione dati *lossy* sono metodi di compressione che comportano perdita di dati, e conseguentemente di qualità.

Quindi, al fine di mantenere l'oggetto elaborato ad un discreto livello qualitativo, eliminano soprattutto le informazioni inutili mantenendo quelle più importanti.

Verranno elencati gli algoritmi della DCT (*Discrete Cosine Transform*) e la quantizzazione.

2.1 Discrete Cosine Transform

Data una funzione a due variabili intere $f(x, y)$ questa trasformata permette di ottenere una nuova funzione $F(u, v)$, con u e v due numeri interi.

La DCT viene definita come:

$$F(u,v) = \frac{2}{\sqrt{NM}} C(u)C(v) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cos \frac{(2x+1)u\pi}{2M} \cos \frac{(2x+1)v\pi}{2N}$$

e le costanti $C(u)$ $C(v)$ si ricavano da:

$$C(\epsilon) = \begin{cases} \frac{\sqrt{2}}{2} \text{ se } \epsilon = 0 \\ 1 \text{ altrimenti} \end{cases}$$

Generalmente i valori di N e M sono uguali a 8.

La trasformata inversa, invece, viene definita in questo modo:

$$f(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u,v) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2x+1)v\pi}{16}$$

In definitiva la DCT serve per decomporre i blocchi 8x8 di un'immagine in una somma di frequenze spaziali. Il calcolo della trasformata assegna ad ogni frequenza un coefficiente, che rappresentano il contributo dato dalla frequenza al blocco elaborato. Le ampiezze negative sono rappresentate in nero, le ampiezza positive sono rappresentate in bianco, mentre quelle nulle sono colorate in grigio; inoltre le frequenze crescono da sinistra verso destra e dall'alto verso il basso.

La figura indica i valori di un generico blocco 8x8 dopo la DCT:

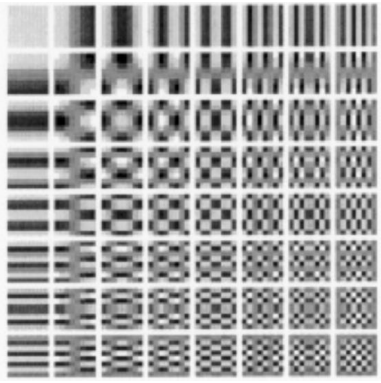

Figura 2.1: Rappresentazione di un blocco 8x8 dopo DCT

2.2 Quantizzazione

La quantizzazione è un processo usato per ridurre la precisione dei coefficienti della DCT, al fine di ottenere un minore bitrate senza un decadimento eccessivo della qualità dell'immagine.

Si tratta di dividere ogni valore risultante dalla DCT per una costante intera, ottenendo un numero intero più una frazione che può essere approssimata in modo diverso: in caso di codifica intra-frame (I-frame), l'arrotondamento è all'intero più vicino, mentre in caso di codifica inter-frame (P o B-frame) e nella codifica JPEG, l'arrotondamento è all'intero inferiore.

La quantizzazione opera attraverso l'uso di alcune matrici, differenti a seconda del grado di compressione risultante che si vuole ottenere.

Le seguenti matrici sono quelle approvate per lo standard MPEG dall'ISO, l'organismo internazionale che si occupa della definizione degli standard, e usate per il processo di quantizzazione:

8	6	5	8	12	20	26	30
6	6	7	10	13	29	30	28
7	7	8	12	20	29	35	28
7	9	11	15	26	44	40	31
9	11	19	28	34	55	52	39
12	18	28	32	41	52	57	46
25	32	39	44	52	61	60	51
36	46	48	49	56	50	52	50

9	9	12	24	50	50	50	50
9	11	13	33	50	50	50	50
12	13	28	50	50	50	50	50
24	33	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	25	26
20	21	22	23	24	25	26	27
21	22	23	24	25	26	27	28
22	23	24	25	26	27	28	29
23	24	25	26	27	28	29	30

16	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Capitolo 3

Conversione RGB in YUV

Uno dei metodi più usati nella codifica cromatica è l'RGB, in cui ogni colore viene espresso mediante le sue componenti di Rosso (R), Verde (G) e Blu (B) in modo additivo, ovvero:

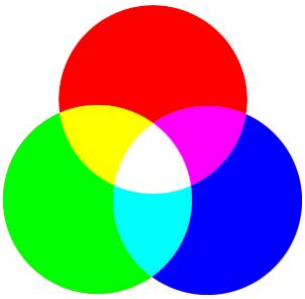

Figura 3.1: Sintesi addittiva di RGB

Uno degli svantaggi di questo metodo è dato dal fatto che l'occhio umano è più sensibile al verde, meno al rosso e ancor meno al blu. Inoltre, la capacità dell'occhio di risolvere i dettagli è maggiore al centro dello spettro visibile e minore ai lati.

Di conseguenza, non c'è ragione di riprodurre accuratamente dettagli che l'occhio non può percepire, conviene quindi utilizzare un altro formato cromatico, come per esempio l'YUV, in cui ogni colore è espresso mediante una componente di luminanza (Y), che ne indica la luminosità, e due di crominanza (U e V), che ne indicano la tinta e la saturazione.

Spesso ci si riferisce alla luminanza col termina "luma" e alla crominanza col termine "chroma".

I valori della crominanza e della luminanza si determinano attraverso le

seguenti formule di conversione:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ U &= 0.492(B - Y) \\ V &= 0.877(R - Y) \end{aligned}$$

oppure nel seguente modo:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ U &= -0.147R - 0.289G + 0.436B \\ V &= 0.615R - 0.515G - 0.100B \end{aligned}$$

Nello standard YUV le due componenti di crominanza possono avere valore negativo: poichè questo può risultare non conveniente, l'MPEG ha adottato lo standard YCbCr, in cui le componenti di crominanza sono scalate, in modo da dare alle tre componenti del colore all'incirca lo stesso intervallo di variazione.

$$\begin{aligned} Cb &= \frac{U}{2} + 0.5 \\ Cr &= \frac{V}{1.6} + 0.5 \end{aligned}$$

Capitolo 4

Codifica MPEG Audio

4.1 Cenni di Psicoacustica

Le nozioni psicoacustiche sono alla base della codifica audio di MPEG.

Si ricorda che l'orecchio umano ha la capacità di recepire suoni che variano da 20Hz a 20kHz di frequenza. I suoni che superano questa soglia vengono definiti *ultrasuoni*. Generalmente la voce ha un range di frequenza variabile tra i 500 Hz e i 4kHz.

Viene inoltre ricordato che l'unità di misura "decibel" rappresenta l'intensità della frequenza nella scala logaritmica.

Il seguente grafico mostra la sensibilità del nostro udito alle varie frequenze:

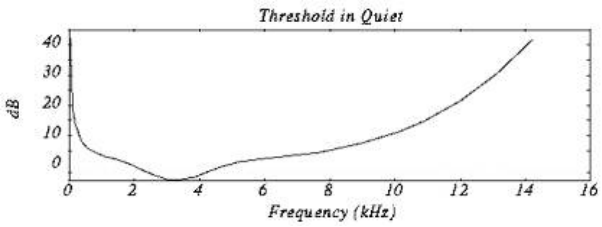

Figura 4.1: Soglia dell'udito unmano

Questo grafico si traccia nel modo seguente: una persona viene posta in una stanza rivestita di materiale fonoassorbente e vengono variate l'ampiezza e la frequenza di un tono a partire da 1 KHz. Quando la persona avverte il tono, si traccia il punto sul grafico corrispondente all'ampiezza per quella specifica frequenza. Otteniamo così l'andamento in frequenza della soglia di udibilità.

Dal grafico emerge che l'orecchio umano è maggiormente sensibile alle frequenze comprese fra 2 e 4 KHz, che richiedono pochissimi dB per essere percepite. Non è un caso che nell'intervallo fra i 2 e i 4 KHz sia compresa la nostra voce.

Grazie a questa particolarità è possibile eliminare quelle componenti spettrali non udibili dall'orecchio medio, in particolare vengono tagliate le alte e le frequenze molto basse.

Un'altra caratteristica del nostro udito consiste nel mascheramento delle frequenze:

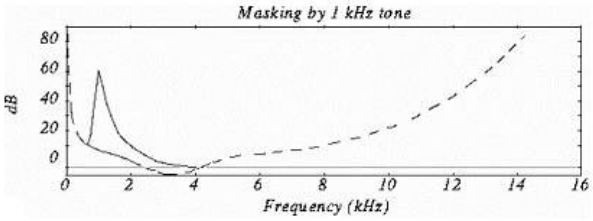

Figura 4.2: Mascheramento di due segnali

In questo caso viene diffuso un tono alla frequenza di 1KHz, detto tono maschera, tenuto fisso a 60dB, e si ripete il discorso precedente sulla soglia di udibilità di un secondo tono, detto tono test.

Quello che emerge è che avvicinandosi sia da sinistra che da destra al tono maschera, si deve alzare il volume del tono test per riuscire a distinguerlo. Oltre i 4 Khz e al di sotto degli 0.5 le cose tornano a posto, però si nota che nell'intorno di 1KHz i due toni sono praticamente indistinguibili a meno che il volume del tono test non venga pesantemente aumentato.

In sostanza, una frequenza debole può essere benissimo mascherata, cioè risultare inudibile.

Questo fenomeno ci permette di eliminare altre componenti spettrali che non risultano udibili all'orecchio umano.

Ultima caratteristica elencata è la capacità del nostro orecchio a recepire un suono subito dopo la fine di un'altro suono di frequenza maggiore che lo mascherava.

Il nostro udito non riesce a sentire subito il nuovo suono, e avrà bisogno di un po di tempo per poter lasciare alla membrana il tempo necessario per riassestarsi.

Come esempio, riportato in figura 5.3 il tono test, di 1KHz e 60dB, viene disattivato all'istante zero. Se il nostro nuovo suono ha un'ampiezza di una quarantina di dB, bastano 5 ms per avvertirlo, mentre se è di soli 20dB, occorrono circa 20 millisecondi perchè risulti avvertibile.

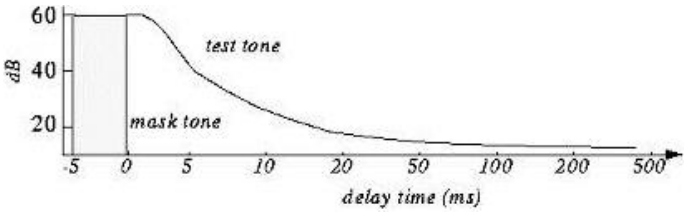

Figura 4.3: Mascheramento temporale

Tenendo conto della sensibilità dell'orecchio e del fenomeno del Masking Audio sarà quindi possibile eliminare dallo spettro del segnale una quantità molto alta di informazioni inutili, poichè non udibili dall'orecchio umano.

4.2 Algoritmo di compressione audio

La compressione di un elemento audio avviene applicando varie trasformazioni al suono in ingresso, facendo sì che il suono che avremo come output risulti compresso. I passaggi che costituiscono la compressione sono i seguenti:

- Applicazione di un banco di filtri al segnale in ingresso, in modo tale da dividerlo in una sequenza di componenti in frequenza;
- Applicazione del modello psicoacustico ai dati in modo parallelo;
- Determinazione del numero di bit necessari alla rappresentazione del coefficiente;
- Composizione del bitstream;

La figura riporta graficamente i vari passaggi dell'algoritmo:

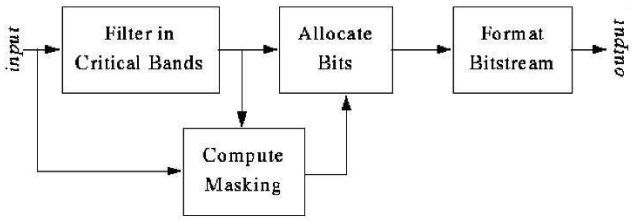

Figura 4.4: Algoritmo codifica mp3

Il segnale in ingresso viene suddiviso in 32 bande attraverso un banco di filtri, producendo come risultato un output di 32 coefficienti, denominati "sottobande". Ogni sottobanda viene suddivisa in frame, ognuno dei quali contenente 384 campioni(12 per ogni sottobanda).

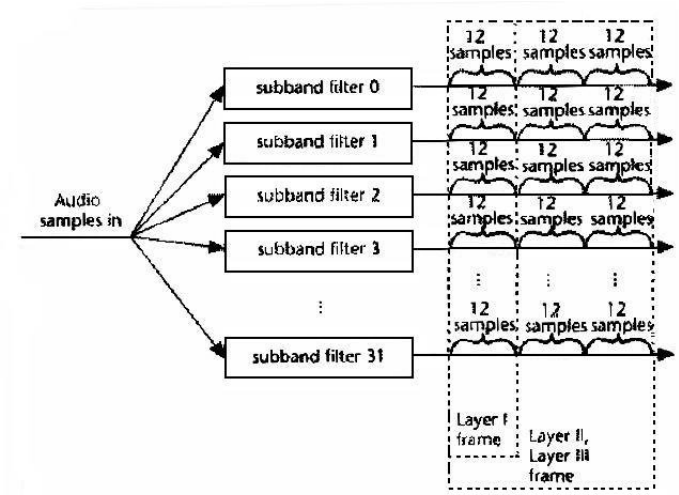

Figura 4.5: Rappresentazione sottobande e frame

Per ogni sottobanda, grazie al modello psicoacustico descritto nel capitolo precedente, viene calcolato il *Signal to Mask Ratio*, definito come il rapporto tra i valori delle frequenze di ogni termine con il valore soglia di ogni sottobanda. Grazie all' SMR e all'SNR *Signal to Noise Ratio*(usato per misurare la qualità del segnale) possiamo ricavare il *Mask to Noise Ratio* come:

$$MNR_{dB} = SNR_{dB} - SMR_{dB}$$

Questo procedimento viene svolto in parallelo con l'applicazione del banco di filtri al nostro segnale d'ingresso su ogni sottobanda.

Se il valore ricavato da questo procedimento è minore del valore della soglia di mascheramento allora questo coefficiente non verrà rappresentato, in modo tale da permettere una rappresentazione finale del nostro output con un numero minore di bit rispetto al segnale d'ingresso.

Il seguente esempio rende più chiari i vari passaggi dell'algoritmo: Supponiamo che i primi 16 valori delle 32 bande siano i seguenti:

Band	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Level (db)	0	8	12	10	6	2	10	60	35	20	15	2	3	5	3	1

- Il valore dell'ottava banda è di 60dB, e ha un mascheramento di 12dB rispetto all settima e di 15dB rispetto alla nona;
- Il valore della settima banda è 10dB, valore minore di 12dB, quindi non viene codificato;
- Il valore della nona banda è 35dB, valore maggiore di 15dB, quindi viene codificato;
- In questo modo invece di usare 6 bits per la codifica, possiamo usarne 4;

La codifica MPEG Layer-3 si diversifica rispetto ai layer 1 e 2 in quanto viene usato un set di filtri particolare, capaci di mantenere molto vicini i valori delle bande critiche.

Inoltre viene usato un modello di DCT modificato, chiamato MDCT(*Modified Discrete Cosine Transform*) capace di rimuovere l'effetto di sovrapposizione che si può avere quando i dati vengono quantizzati e successivamente elaborati con una trasformata inversa, la cui formula è:

$$F(u) = 2\sum_{i=0}^{N-1} f(i)\cos\left[\frac{2\pi}{N}\Big(i + \frac{\frac{N}{2}+1}{2}(u+\frac{1}{2})\Big)\right],\ \ u = 0,\ldots,\frac{N}{2}-1$$

Capitolo 5

Codifica MPEG Video

La codifica MPEG è uno standard video creato dal *Motion Picture Expert Group*.

Un video per appartenere a questo standard deve rispettare alcune direttive dettate dall'*ISO*:

- *Accesso casuale*: si tratta di una funzione essenziale sui dispositivi che permettono l'accesso casuale. Ciò significa che i dati video devono essere accessibili al loro interno e che ogni frame debba essere codificabile in un tempo limitato, quindi viene richiesta la presenza di punti d'accesso.

- *Ricerca veloce avanti/indietro*: deve essere possibile la scansione dei dati compressi e, utilizzando punti di accesso appropriati, mostrare immagini selezionate, per ottenere un effetto di riavvolgimento e di avanzamento veloce.

- *Riproduzione all'indietro*: viene richiesto che il segnale possa essere letto all'indietro pur non mantenedo un' alta qualità.

- *Sincronizzazione audio/video*: il segnale video deve poter essere associato ad un segnale audio.

- *Robustezza*: è necessaria una codifica robusta in modo tale da permettere la correzione di eventuali errori.

Un video può essere interpretato, generalmente, come una sequenza di immagini. È necessario che queste immagini abbiano una codifica individuale e che siano rappresentate ad una velocità prefissata(*frame rate*).

5.1 Divisione dell'immagine in GOP

Nella compressione MPEG i frame della sequenza video vengono divisi in GOP, acronimo di *Group Of Picture*, e codificati attraverso l'uso di tre algoritmi diversi, che verranno spiegati in seguito:

- I-frame (*Intra image*): sono codificati attraverso una tecnica basata su DCT simile alla codifica JPEG ed in modo indipendente, cioè senza alcun riferimento ad altri frame. Inoltre costituiscono gli access point della sequenza data.

- P-frame (*Predicted image*): vengono codificati attraverso la predizione del moto in avanti (forward predictive coding); cioè il frame attuale viene codificato tramite un riferimento ad un frame precedente (I o P frame).

- B-frame (*Bidirectional image*): sono codificati attraverso l'uso di due frame, uno precedente al frame attuale e uno successivo (I o P), dei quali viene fatta la differenza.

Generalmente la sequenza dei frame di un video è composta in questo modo:

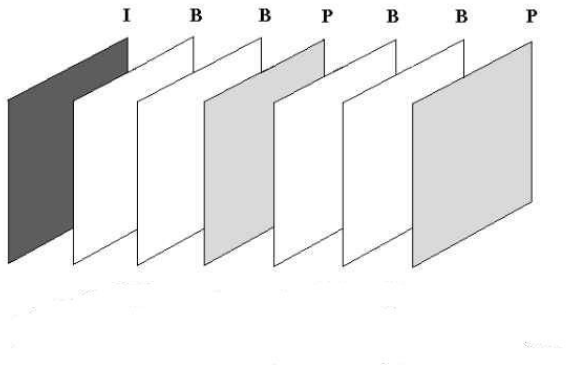

Figura 5.1: Struttura GOP

Quando l'immagine viene suddivisa in GOP si tende a tenere una struttura regolare per tutto il filmato, e questo e' possibile mantenendo costante la distanza tra il frame I e quelli successivi di tipo P.

Diversamente dai frame precedenti i B-frame usano una compensazione del moto bidirezionale.

5.2 Codifica I-frame

I frame di tipo I sono indispensabili per dare la possibilità di avere un video con accesso casuale, per permettere la ricerca veloce avanti/indietro e per determinare una bassa propagazione di eventuali errori.

Questa codifica avviene in modo molto simile alla codifica dello standard JPEG (che verrà spiegata nel capitolo successivo), infatti ne segue pari passo tutti i passaggi, a partire dalla trasformata DCT, fino alla codifica entropica.

Inizialmente viene applicata la DCT ai blocchi dell'immagine ,successivamente avviene la quantizzazione, che applica solitamente la matrice:

$$\begin{pmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 8 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{pmatrix}$$

Come per la codifica JPEG il passaggio successivo consiste nella codifica d'entropia, dove i coefficienti DC vengono codificati prima con l'algoritmo DPCM e poi attraverso HUFFMAN, mentre i coefficienti AC vengono tradotti attraverso la RUNLENGTH encoding e poi attraverso HUFFMAN.

5.3 Codifica P-frame

In questo passaggio ad ogni macroblock del target P-frame viene assegnato un macroblocco del frame I o P precedente con il miglior matching. Questa caratteristiche viene chiamata *predizione*.

La differenza tra il macroblocco ed il macroblocco "matching" viene chiamata *errore di predizione*, che viene successivamente passato alla DCT per la codifica.

Se la predizione avviene a partire dal frame precedente allora si parlerà di *forward prediction*.

Nel caso in cui un movimento inaspettato occluda il nostro macroblocco, non si è in grado di trovare un matching soddisfacente, e quindi questo macroblock può essere ottenuto anche da frame successivi, come mostrato nella figura:

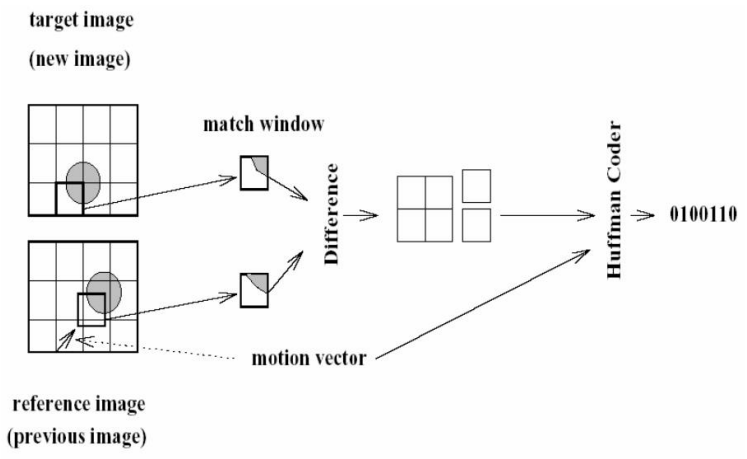

Figura 5.2: Algoritmo di codifica di un P-frame

5.4 Codifica B-frame

Questo tipo di frame adotta la compensazione bidirezionale del moto.

Essenzialmente aggiunge alla forward prediction anche la backward prediction, nella quale il matching macroblock viene cercato tra i frame I o P successivi, e per la codifica ogni B-frame verrà specificato attraverso due motion-vector, uno generato dalla forward e uno dalla backward prediction.

Se il matching in entrambe le direzioni risulta soddisfacente allora i vettori vengono passati al bitstream, e verrà fatta la media dei due blocchi in questione. Nel caso in cui il matching in una direzione non risulti soddisfacente viene usato un solo vettore ed un solo macroblock.

La figura riporta la codifica di un frame di tipo B:

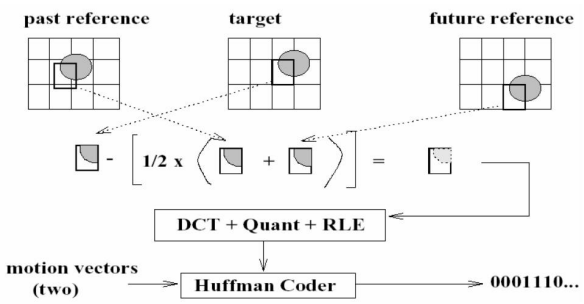

Figura 5.3: Algoritmo di codifica di un P-frame

L'uso in MPEG di questo tipo di frame fa sì che la sequenza di frame

effettiva della figura e l'ordine di trasmissione siano differenti, in quanto il frame B deve essere trasmesso dopo i frame I e P, come rappresentato in figura:

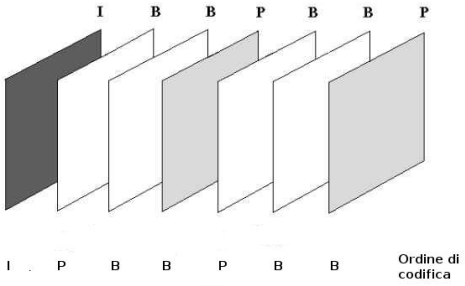

Figura 5.4: Ordine di trasmissione dei frame

5.5 MPEG Layer

La figura riporta graficamente i 6 layer che compongono il bitstream di un video MPEG:

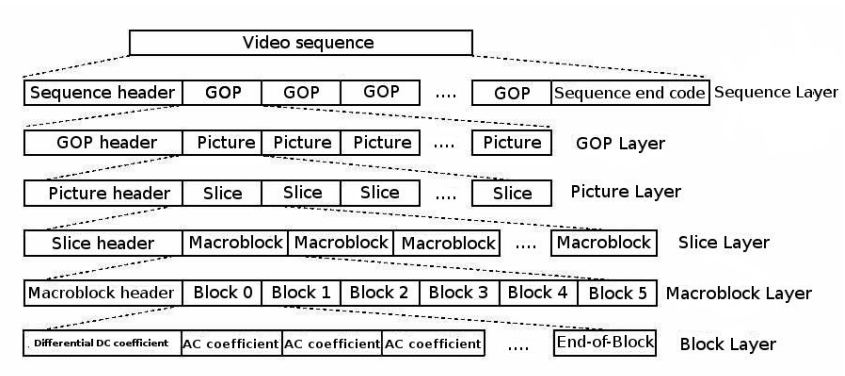

Figura 5.5: Header di un file MPEG

I contenuti dei vari campi sono generalmente i seguenti:

- SEQUENCE LAYER: contiene sempre un sequence header ed un sequence end e include le informazioni sulle figure come le dimensioni orizzontali e verticali, il frame rate e la matrice di quantizzazione;

- GROUP OF PICTURES LAYER: contiene una o più figure, una delle quali deve essere di tipo I, e le informazioni temporali, quali ore-minuti-secondi dall'inizio della sequenza;
- PICTURE LAYER: corrisponde ad un frame.
- SLICE LAYER: MPEG introduce il concetto di Slice costituiti da una sequenza di macroblock. Questo layer è importante perchè permette una nuova sincronizzazione in seguito alla lettura di bit corrotti. Gli Slices possono avere lunghezze variabili, che devono essere dichiarate all'interno dell'header
- MACROBLOCK LAYER: ogni macroblocco consiste in 4 blocchi per la luminanza Y, uno per il valore Cb ed uno per il valore Cr. Ogni blocco appena menzionato è di dimension 8x8.
- BLOCK LAYER: in questo layer l'informazione cambia a seconda del tipo di blocco si sta analizzando. Se il blocco è di tipo intra, cioè un I-frame allora il coefficiente DC (codificato con DPCM) viene letto per primo seguito dai coefficienti AC (codificati con VLC). Se, invece, il blocco esaminato è di tipo inter (P o B-frame) allora i coefficienti AC e DC vengono entrambi codificati con VLC.

Capitolo 6

Codifica JPEG

JPEG è uno standard per la compressioni delle immagini sviluppato dal Joint Photographic Expert Group.

Questo standard fornisce quattro modi di operare:

- *Codifica sequenziale*: è basata su DCT, dove ogni componente del'immagine viene codificato in uno scan singolo. È un tipo di compressione lossy;
- *Codifica progressiva*: è basata su DCT, ed è codificata in più scansioni, in modo tale da permettere una decodifica veloce anche in caso di tempi di trasmissione elevati. Per questo motivo è molto usata nella applicazioni web. Questa codifica è di tipo lossy.
- *Codifica lossless*: l'immagine viene codificata in modo tale da averne una riproduzione esatta. Viene codificata attraverso l'algoritmo di Huffman.
- *Codifica gerarchica*: l'immagine viene codificata con risoluzioni multiple; questa codifica è di tipo lossy.

Di seguito verrà illustrata la codifica JPEG di tipo sequenziale.

Essenzialmente l'algoritmo di compressione JPEG è suddiviso in quattro fasi principali, ciascuna delle quali contribuisce alla compressione finale dell'immagine elaborata. Le fasi dell'algoritmo sono le seguenti:

- Conversione dal formato RGB a YUV;
- FDCT
- Quantizzazione;

- Codifica d'entropia;

La seguente figura rappresenta graficamente le varie fasi della codifica:

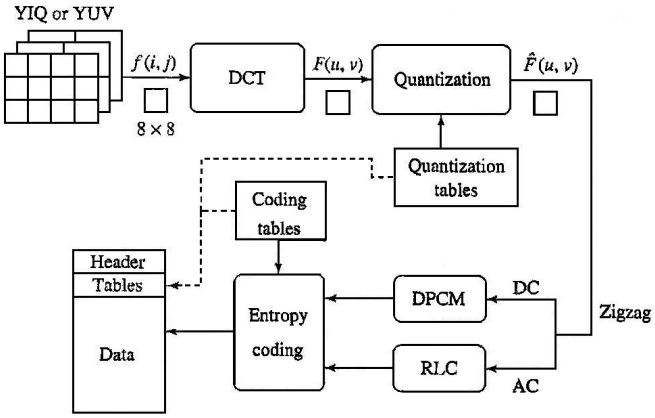

Figura 6.1: Algorimto JPEG

6.1 DCT

FDCT (Forward Discrete Cosine Transform) comprime l'immagine concentrando la maggior parte del segnale nelle componenti di bassa frequenza, in quanto le componenti presenti nelle alte frequenze sono meno distinguibili. Per un tipico blocco, buona parte delle frequenze spaziali o sono nulle o sono prossime allo zero e non necessitano quindi di codifica. Questa compressione è di tipo lossless.

La FDCT opera nel seguente modo:

- L'immagine originale viene completamente suddivisa in blocchetti contenenti 8x8 elementi (facilmente visibili se si usa un alto grado di compressione, dove si nota un'immagine molto "spigolosa");

- Ogni elemento di ciascun blocchetto viene shiftato da tipo intero senza segno compresi in [0, 2p-1] a intero con segno compresi invece tra [-2p-1 , 2p-1 -1]. P rappresenta la precisione dell'immagine.

- Ogni blocchetto viene dato in input alla FDCT, che lo considera come una matrice di 64 valori e verrà elaborato attraverso la seguente formula:

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2x+1)v\pi}{16}$$

con

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} \text{ se u,v=0} \\ 1 \text{ altrimenti} \end{cases}$$

che restituisce come output un insieme di 64 valori rappresentanti i valori in cui il segnale originale è stato scomposto.

Sarà presente un coefficiente con entrambi i valori nulli F(0,0) che verrà chiamata DC (Direct Coefficient), mentre tutti gli altri valori saranno chiamati AC (Alternate Coefficient).

6.2 Quantizzazione

La quantizzazione è il passo della codifica JPEG che permette di ridurre il numero totale di bit necessari per rappresentare l'immagine compressa riducendo l'ampiezza dei coefficienti il cui contributo è molto basso o nullo per la qualità dell'immagine.

Questa caratteristica rende le modifiche apportate irreversibili, ed è la causa per cui la codifica sequenziale risulta lossy.

Questo passaggio consiste essenzialmente tra il rapporto dei valori di ogni blocchetto e tra i rispettivi valori della tabella di quantizzazione scelta, il tutto arrotondato per difetto.

In particolare il divisore F(u,v) rappresenta i valori risultanti dalla DCT, Q(u,v) rappresenta i valori della tabella di quantizzazione.

$$F_q(u,v) = \lfloor \frac{F(u,v)}{Q(u,v)} \rfloor$$

A seconda del grado di compressione che si vuole ottenere si può scegliere la matrice adatta tra quelle inserite nello standard MPEG dall'ISO.

6.3 Codifica d'entropia

Questa parte della codifica è caratterizzata dalla diversa considerazione tra il coefficiente DC e gli altri coefficienti AC.

I valori DC vengono codificati in questo modo:

- DPCM (Differetial Pulse Code Modulation);
- Codifica di Huffman;

mentre i valori AC attraverso:

- RLC (Run Length Coding);
- Codifica di Huffman;

La codifica RLC usa uno schema di lettura dei valori tale da avere lunghe sequenze di zeri, la lettura a Zig-Zag. Questa particolare lettura tramuta il blocco 8x8 in questione in un vettore da 64 valori.

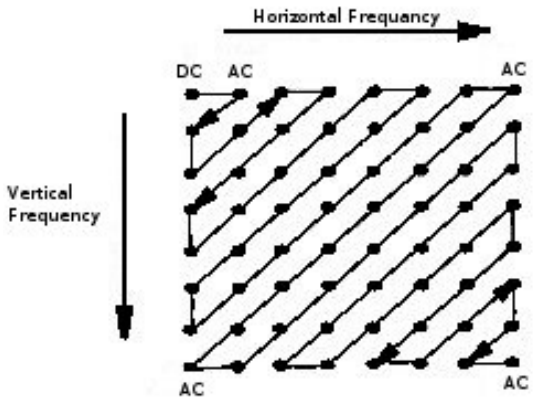

Figura 6.2: Lettura a Zig-Zag

Per esempio la lettura del seguente blocco F(u,v):
porta ad avere il seguente vettore:

$$(32, 6, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 1, 0, 0, ..., 0)$$

I valori ottenuti vengono poi sostituiti dalla coppia (RUNLENGTH, VALUE).

La variabile RUNLENGTH rappresenta il numero di zeri della lettura, mentre il VALUE rappresenta il prossimo valore diverso da zero.

Prendendo in considerazione l'esempio precedente avremo i seguenti valori

$$(0, 6)(0, -1)(0, -1)(1, -1)(3, -1)(2, 1)(0, 0)$$

32	6	−1	0	0	0	0	0
−1	0	0	0	0	0	0	0
−1	0	1	0	0	0	0	0
−1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 6.3: Esempio di matrice F(u,v)

tenendo in considerazione il fatto che il primo valore del blocco è il valore DC. La codifica RLC viene eseguita per ogni blocco 8x8 dell'immagine.

La codifica dei valori DC spetta alla DPCM, la quale tiene conto del fatto che i coefficienti DC di blocchi diversi possono essere dei valori grandi e differenti.

Come esempio si consideri che i coefficienti DC di alcuni blocchi diversi siano rispettivamente i valori:

$$(150, 155, 149, 152, 144)$$

DPCM restituirà i valori:

$$(150, 5 - 63 - 8)$$

Questi valori vengono trovati assumendo che l'i-esimo blocco predetto sia ricavabile attraverso

$$\begin{aligned} d_i &= DC_(i+1) - DC_i \\ d_0 &= DC_0 \end{aligned}$$

La codifica DPCM viene eseguita una sola volta per tutti i blocchi che costituiscono l'immagine.

Il passaggio successivo a RLC e a DPCM è la codifica d'entropia, che viene ottenuta attraverso la codifica di Huffman.

La codifica per i blocchi AC avviene tenendo conto della coppia (RUN-LENGTH ,VALUE) ottenuta mediante RLC.

La variabile VALUE, a questo punto, viene rimpiazzata da SIZE e AMPLITUDE, le quali possono usare al massimo 4 bit ognuna e insieme occupano un byte. Questa coppia verrà considerata come Simbolo 1.

AMPLITUDE sarà invece considerato come Simbolo 2 e il suo numero di bit indicato dal valore SIZE.

$$\begin{aligned} Simbolo1 &= (RUNLENGTH, SIZE) \\ Simbolo2 &= (AMPLITUDE) \end{aligned}$$

La codifica di Huffman per i coefficienti DC tiene conto che ogni valore codificato attraverso DPCM è rappresentato da una coppia da una coppia di simboli (SIZE , AMPLITUDE). La variabile SIZE serve per tenere traccia del numero di bit che sono necessari alla rappresentazione del coefficiente, mentre la variabile AMPLITUDE serve per tenere conto dei bit come codice.

Come esempio, considerando i valori ottenuti attraverso la DPCM

$$(150, 5, -6, -3, -8)$$

si ottengono le seguenti coppie di valori:

$$(8, 10010110)(3, 101)(3, 001)(2, 11)(4, 0111)$$

Tra i due valori della coppia (SIZE , AMPLITUDE) solo il valore SIZE viene codificato attraverso Huffman, in quanto questa codifica su AMPLITUDE (che contiene valori che possono cambiare ampiamente) non comporta risultati evidenti. Dopo aver superato tutti questi passaggi l'immagine risulta compressa e rappresentata in modo più o meno soddisfacente a seconda del livello di compressione scelto. Sarà inoltre possibile misurare il rapporto di compressione dell'immagine finale in due modi differenti, attraverso le seguenti formule:

$$Rc = \frac{Dim.file\ \ compresso}{Dim.file\ \ originale}$$

oppure

$$Rc = \frac{bit\ \ codificati}{pixel}$$

L' immagine finale è composta in questo modo:

dove il campo FRAME indica la figura, il campo SCAN indica un passaggio attraverso i pixel dell'immagine, il SEGMENT rappresenta un gruppo di blocchi, mentre BLOCK rappresenta nel blocchetto 8x8 . Un esempio di informazioni contenute nell'header puo essere il seguente:

FRAME contiene:

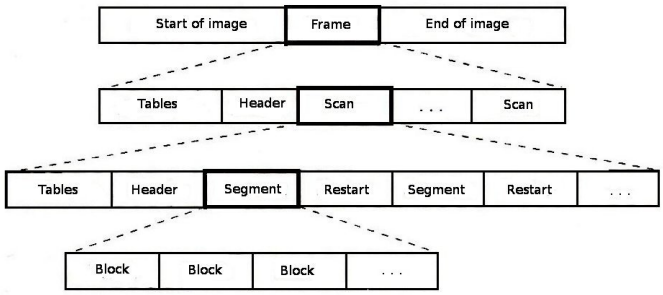

Figura 6.4: Composizione finale dell'immagine

- I bit per pixel;
- Larghezza e altezza dell'immagine;
- Numero di componenti;
- ID univoco;
- Tabella di quantizzazione usata;

SCAN contiene:

- Il numero di componenti nello scan;
- ID dei componenti;
- Tabella della codifica di Huffman.

6.4 Decodifica

La decodifica JPEG esegue il processo inverso dell' encoder, e ne inverte l'ordine di esecuzione in questo modo:

- decodifica d'entropia;
- dequantizzazione;
- Inverse DCT;

La Decodifica dell'entropia prende la sequenza binaria e la trasforma prima in sequenza intermedia di simboli, poi nella sequenza a Zig-Zag rappresentante i 64 coefficienti DCT quantizzati

La Dequantizzazione prende i 64 coefficienti quantizzati DCT e li trasforma in 64 coefficienti DCT semplici seguendo lespressione:

$$F_q^{'}(u,v) = F_q(u,v)Q(u,v)$$

L'Inverse DCT (IDCT) prende in input i 64 coefficienti DCT e ricostruisce in output il segnale discreto di 64 punti rappresentante un blocco 8x8 dellimmagine originale

$$F(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)f(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2x+1)v\pi}{16}$$

con

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} \text{ se u,v=0} \\ 1 \text{ altrimenti} \end{cases}$$