Università degli Studi di Firenze

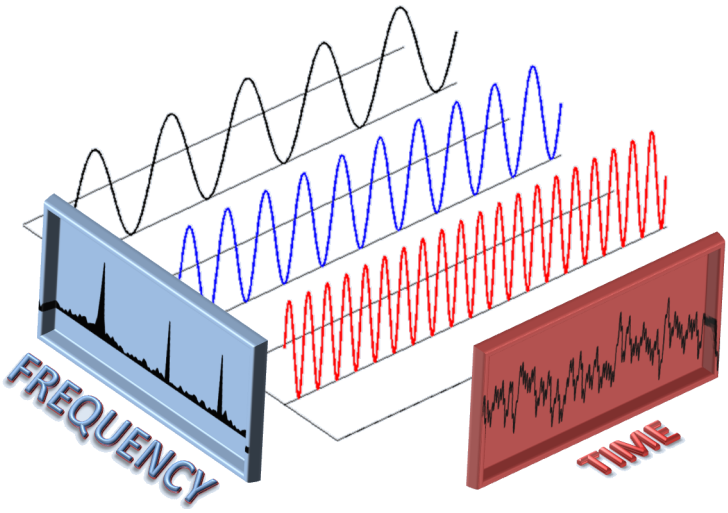
Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Magistrale in Informatica (classe LM-18)

Anno accademico 2012/2013

La teoria dell'MP3

Autore:
Tommaso Papini
tommy39@gmail.com



settembre 2013

Indice

Elenco delle figure

Elenco delle tabelle

Abstract

In questa relazione ci occuperemo di analizzare il formato ***MP3*** (*MPEG-I Layer III*), ormai ampiamente consolidato in tutto il mondo come formato audio ed efficiente metodo di compressione.
Verrà innanzitutto definito il contesto in cui si viene a trovare l'*MP3*, esponendo brevemente alcuni fondamenti di teoria del suono e raccontando la storia che ha portato all'evoluzione dei formati audio precedenti all'*MP3*.
Quindi verrà analizzato il formato *MP3* stesso, sia come formato di file audio che come metodo di compressione (e decompressione) audio, esponendone le idee di base e le principali tecniche utilizzate.
Infine, verranno proposte alcune note conclusive, non volte ad esporre alcun risultato particolare, ma intese come commentario finale e riassuntivo dell'argomento trattato e della relazione stessa.

Parte I

Breve storia e introduzione

Capitolo 1

Introduzione

CHIUNQUE abbia la seppur minima passione per la musica avrà sicuramente sentito parlare, al giorno d'oggi, del formato (o più in generale, della tecnologia) *MP3*. Sicuramente i più sapranno che l'MP3 è un formato audio digitale che permette di memorizzare file audio, come canzoni, utilizzando molto meno spazio rispetto ai precedenti formati, pur mantenendo una buona qualità sonora: molti ricorderanno l'avvento dei primi lettori CD MP3, che permettevano di registrare e riprodurre centinaia di canzoni in un unico CD da 700 MB, al contrario delle massimo 20 canzoni che si potevano masterizzare in qualità CD (di queste differenze di formati e qualità audio parleremo ampiamente più avanti).
La parola MP3 è spesso associata ai concetti di "Internet", "frode" o "pirateria informatica", dal momento che la sua venuta ha dato il via, o comunque incentivato, il download di enormi quantità di dati audio, spesso senza possedere alcuna licenza ed in modo certamente illegale.

Ma a parte questi aspetti di conoscenza comune, pochi sanno veramente cos'è l'MP3 e come funzioni ed ancor meno sono quelli che si prendono la briga di spiegarlo. Con questo elaborato ci prefiggiamo quindi l'obiettivo di descrivere come e perché è nato l'MP3 e come esso funzioni, senza perdersi troppo nei dettagli tecnici ma fornendo comunque un'idea generale della tecnologia MP3.

1.1 Definizioni

Senza dilungarci troppo nei dettagli, diamo alcune definizioni iniziali che faciliteranno la comprensione di questa relazione.

Definizione 1.1 *L'***MP3** *(ovvero* **M**oving **P**icture *Expert Group-1/2 Layer* ***III****), detto anche* MPEG-1/2 Layer III*, è un algoritmo di compressione audio di tipo* lossy*, sviluppato dal gruppo* MPEG*.*

Avendo introdotto, nella definizione precedente, il concetto di algoritmo *lossy*, definiamo adesso cos'è un algoritmo di compressione *lossy* e cos'è invece uno *loosless*:

Definizione 1.2 *Un algoritmo di compressione* **lossy** *è un metodo di codifica che comprime i dati scartandone alcuni. Tramite il processo di decodifica, quindi, non sarà possibile riottenere i dati originali.*

Definizione 1.3 *Un algoritmo di compressione* **loosless** *è un metodo di codifica che comprime i dati senza perderne alcuno. Tramite opportuna decodifica sarà quindi possibile ottenere nuovamente i dati originali.*

Già da queste definizioni possiamo dedurre che l'MP3 è un algoritmo di compressione audio che, comprimendo i dati, scarta alcune informazioni, al fine di rendere il file finale molto più leggero e maneggevole.

1.2 Cenni di compressione di dati

Nel 1949 Claude E. Shannon provò, all'interno del suo articolo "A Mathematical Theory of Communication", che esiste un limite teorico alla compressione dei dati senza perdere informazione, ovvero comprimendo i dati con algoritmi loosless. Questo limite, detto *tasso d'entropia*, dipende dalla probabilità di trovare determinate sequenze di bit: è possibile comprimere i dati con un tasso di compressione vicino al tasso d'entropia ed è matematicamente impossibile fare meglio.
Per ottenere una maggior compressione dei dati è necessario utilizzare algoritmi lossy e quindi accettare di perdere parte dei dati.

Di seguito esporremo tre codifiche di tipo loosless molto semplici, una delle quali, come vedremo, utilizzata anche all'interno dell'algoritmo di compressione e decompressione MP3.

1.2.1 Run-Length Encoding

L'algoritmo di compressione loosless *Run-Length Encoding* (o semplicemente *RLE*) consiste nel codificare le varie sequenze di bit consecutivi aventi lo stesso valore come coppie, dove il primo elemento rappresenta il valore di quei bit e il secondo elemento indica il numero di bit che compongono la sequenza, ovvero la lunghezza della sequenza. In Figura 1.1 possiamo vedere un esempio della codifica RLE. Questa codifica è ideale per dati con lunghe sequenze di bit identici (quindi sconsigliato per dati casuali).

Figura 1.1: Algoritmo di compressione Run-Length Encoding.

1.2.2 Move-To-Front

La codifica *Move-To-Front* (o *MTF*) si basa sul concetto di entropia ed infatti è ottimizzata quando la lettura di un carattere aumenta le probabilità di trovare lo stesso carattere subito dopo.
L'algoritmo inizia codificando le lettere dell'alfabeto secondo l'ordine usuale (da 0 a 25). Quindi ogni carattere che viene incontrato viene spostato in cima alla lista. In generale, elementi in cima alla lista vengono codificati con meno bit, mentre quelli verso il fondo richiedono più bit.

Se ad esempio prendiamo la parola "BANANAAA", la codifica MTF di questa parola sarà quella in Tabella 1.1.

Sequenza	**Codifica**	**Lista**
b	1	abcdefghijklmnopqrstuvwxyz
ba	1, 1	bacdefghijklmnopqrstuvwxyz
ban	1, 1, 13	abcdefghijklmnopqrstuvwxyz
bana	1, 1, 13, 1	nabcdefghijklmopqrstuvwxyz
banan	1, 1, 13, 1, 1	anbcdefghijklmopqrstuvwxyz
banana	1, 1, 13, 1, 1, 1	nabcdefghijklmopqrstuvwxyz
bananaa	1, 1, 13, 1, 1, 1, 0	anbcdefghijklmopqrstuvwxyz
bananaaa	1, 1, 13, 1, 1, 1, 0, 0	anbcdefghijklmopqrstuvwxyz

Tabella 1.1: Algoritmo di compressione Move-To-Front.

1.2.3 Codifica di Huffman

Il concetto di entropia viene ampiamente applicato anche alla codifica di Huffman che, come vedremo più avanti, viene utilizzata all'interno dell'algoritmo di compressione MP3. L'idea che sta alla base della codifica di Huffman è quella di codificare con meno bit i caratteri più frequenti. La probabilità di incontrare determinati caratteri dev'essere determinata a priori (ad esempio analizzando i dati che si vogliono comprimere). Successivamente, in base alle probabilità calcolate, si assegnano codifiche più corte ai caratteri più frequenti e si memorizzano le associazioni carattere-codifica in una tabella, detta *tabella di Huffman*, necessaria per la successiva decodifica dei dati. Come possiamo vedere in Figura 1.2, la tabella di Huffman può essere rappresentata anche come un albero binario.

Simbolo	Probabilità
A	0.13
B	0.05
C	0.33
D	0.08
E	0.18
F	0.23

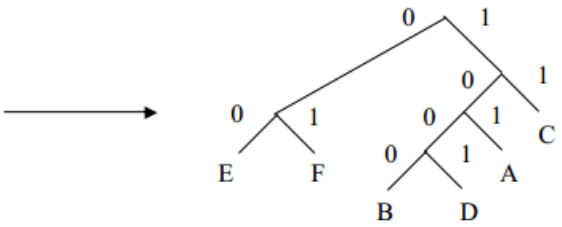


Figura 1.2: Codifica di Huffman.

Capitolo 2

Cenni di teoria del suono

Intuitivamente, prima di poter parlare di algoritmi di compressione audio, e quindi di MP3, sarà necessario innanzitutto definire cos'è il suono e come si comporta. In questo capitolo verranno quindi presentati alcuni concetti fondamentali di teoria del suono, sia dal punto di vista fisico (onde e tutto ciò che ne concerne) che dal punto di vista informatico (campionamento e sintesi).

2.1 Il suono come onda

Definizione 2.1 *Il* **suono** *è un'*onda meccanica*, ovvero un'*onda di pressione *che si propaga attraverso un mezzo.*

Questa definizione che abbiamo appena dato sul suono è molto chiara e concisa: il suono è un'onda. Ma non una qualsiasi onda. Il suono è un'onda di pressione, ovvero non è nient'altro se non l'oscillazione (movimento "avanti-indietro", in parole povere) del mezzo in cui si propaga, che può essere solido o fluido. Il timpano umano è infatti una membrana sottilissima volta a captare queste oscillazioni presenti attorno a noi; sarà poi il cervello che tradurrà i movimenti della membrana in informazioni utili, interpretandole come quello che noi chiamiamo suono.
Come tutte le onde, anche il suono avrà quindi tutta una serie di attributi caratteristici, quali:

- **periodo**: minimo intervallo di tempo dopo il quale i valori dell'onda si ripetono (s, secondi);
- **frequenza**: numero di oscillazioni per unità di tempo (Hz, hertz);
- **lunghezza d'onda**: minima distanza dopo la quale i valori dell'onda si ripetono(m, metri);
- **numero d'onda**: numero di oscillazioni per unità di lunghezza (m^{-1}, metri alla meno uno);

- **ampiezza**: misura i cambiamenti dell'onda all'interno di uno stesso periodo (varia);
  - *ampiezza picco-a-picco*: massimo scarto tra una cresta (punto di massimo dell'onda) e la valle (punto di minimo) successiva;
  - *ampiezza a picco*: massimo valore assoluto dell'onda;
  - *ampiezza RMS* (root mean square): scarto quadratico medio dell'ampiezza.
- **velocità**: velocità con cui l'onda si propaga nel mezzo ($^m/_s$, metri al secondo).

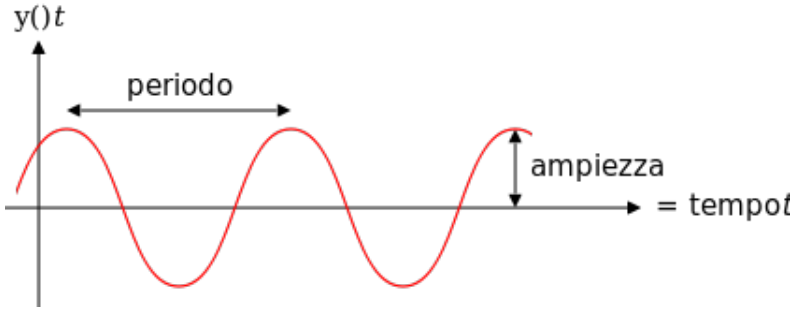


Figura 2.1: Periodo e ampiezza di un'onda periodica.

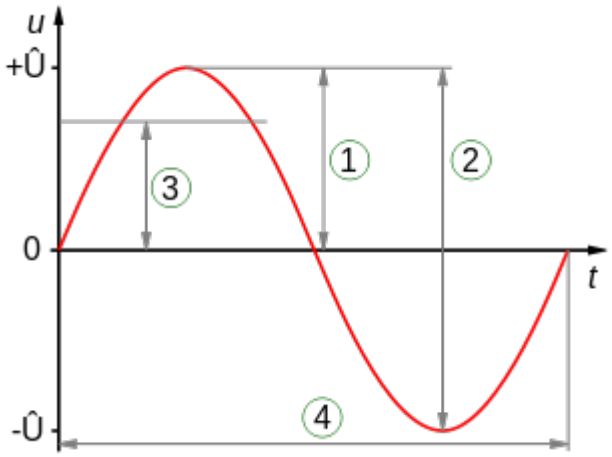


Figura 2.2: Diversi tipi di ampiezze: ampiezza a picco (1), ampiezza picco-a-picco (2) e ampiezza RMS (3).

Come è facile dedurre, *periodo* e *frequenza* sono uno l'inverso dell'altro, come anche *lunghezza d'onda* e *numero d'onda*. L'unità di misura dell'ampiezza dipende dal tipo d'onda che si sta analizzando: se si tratta di corrente elettrica può rappresentare un voltaggio (V, volt), una corrente (A, ampere) o la potenza (W, watt), mentre se parliamo di onde sonore l'ampiezza può rappresentare la forza dell'onda di pressione (dB, decibel), oppure il movimento oscillatorio stesso, con 0 punto di quiete e -1 ed 1 limiti, rispettivamente,

inferiore e superiore. La velocità d'onda è invece calcolata come $v = {}^{\lambda}/_{p} = \lambda \cdot f$, dove λ è la lunghezza d'onda, p il suo periodo ed f la sua frequenza. La velocità del suono a 20 $^{\circ}C$ nell'aria al livello del mare è circa 343 ${}^{m}/_{s}$, ma come per molte altri tipi di onda la sua velocità varia a seconda di molti fattori, come natura del mezzo, temperatura, densità, ecc....

2.2 Percezione del suono

Avendo definito il suono come un'onda, possiamo affermare che in natura esistono un'infinità di suoni diversi, generati da altrettante frequenze caratteristiche. L'orecchio umano, tuttavia, non è in grado di percepire tutte le frequenze esistenti in natura: in generale, l'uomo riesce a percepire tutte le frequenze che vanno dai 20 *Hz* ai 20 *kHz* (20mila *Hz*). Ovviamente, ci sono eccezioni e casi particolari di persone che affermano di riuscire a percepire e distinguere anche frequenze al di fuori di questo range. Inoltre, l'uomo non percepisce tutte le frequenze allo stesso modo: più ci si avvicina ai limiti del range di udibilità, più sarà difficile percepire tale suono. Infatti le frequenze che meglio possiamo sentire sono quelle tra i 2 e i 4 *kHz* [14].
A tal proposito, la Figura 2.3 rappresenta la *soglia assoluta di udibilità* (in inglese, *absolute threshold of hearing*) per un uomo medio a 20, 40 e 60 anni, ovvero qual'è la minima intensità che ogni frequenza deve avere per essere percepita dall'uomo. In altre parole, tutto ciò che l'uomo non sente è rappresentato dall'area sottesa dal grafico in Figura 2.3.

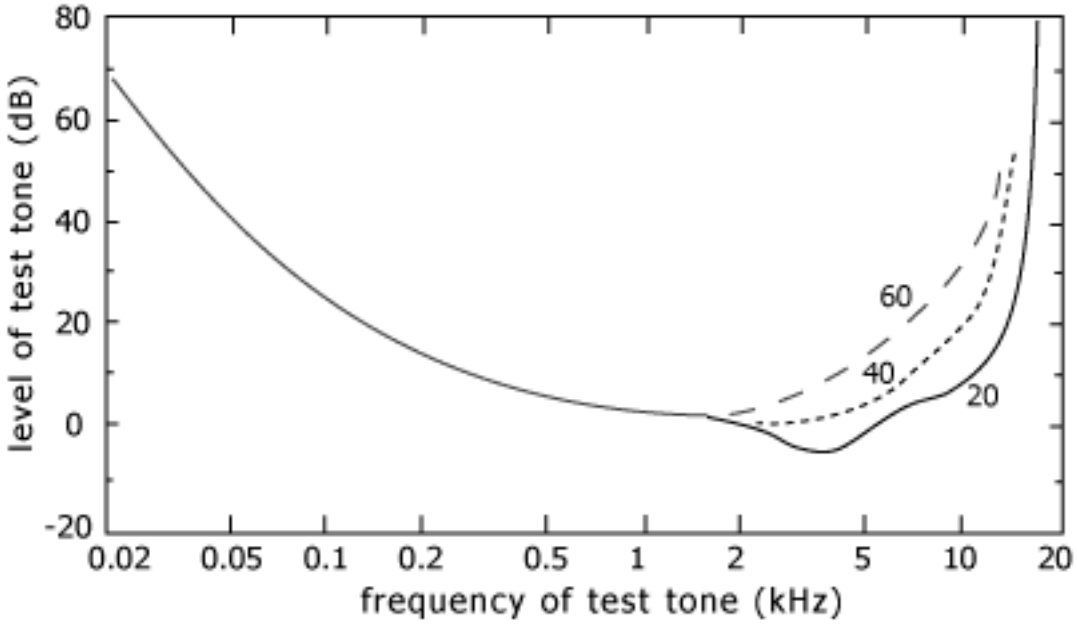


Figura 2.3: Soglia assoluta di udibilità per un uomo medio a 20, 40 e 60 anni.

L'uomo, quindi, non è una macchina, in quanto il suo orecchio e il suo cervello non sono strumenti perfetti. È nata infatti negli anni la disciplina detta *psicoacustica*, che si occupa di studiare non tanto il suono in sé dal punto di vista fisico, ma come esso viene percepito dall'uomo. Della psicoacustica sentiremo parlare più avanti quando entreremo nel merito dell'MP3, in quanto essa costituisce uno dei principi cardine che stanno alla base della tecnologia MP3.

2.3 Sintesi e Campionamento

Abbiamo detto che il suono è un'onda di pressione, ovvero il movimento oscillatorio ("avanti-indietro") del mezzo nel quale si propaga. Infatti tutti i suoni sono prodotti dal movimento, più o meno rapido, di uno o più corpi, il quale genera questa catena di oscillazioni, producendo il suono stesso.

2.3.1 Sintesi

Come viene riprodotto, allora, artificialmente un suono? Secondo quanto detto, abbiamo bisogno di un corpo che faccia a comando questo movimento oscillatorio. Non solo, è necessario anche che il movimento "avanti-indietro" che questo corpo esegue abbia la stessa frequenza del suono che vogliamo riprodurre, ovvero che esegua lo stesso numero di oscillazioni nell'unità di tempo.

L'oggetto di cui stiamo parlando è detto *speaker*, o più comunemente *altoparlante*. Uno speaker è formato da un elettromagnete che, a seconda dell'intensità della corrente elettrica che lo attraversa, si sposterà avanti o indietro rispetto alla posizione di quiete. La gestione dello speaker avviene tramite una *scheda audio* (o *soundcard* in inglese), ovvero un DAC (*Digital-to-Analog Converter*) che trasforma un numero digitale in input in un segnale elettrico in output. Se si utilizza la convenzione prima esposta (0 = stato di quiete, -1 = tutto indietro, 1 = tutto avanti) possiamo allora fornire alla scheda audio una serie di valori nel tempo, in modo da far muovere lo speaker avanti e indietro. Se rappresentassimo questo movimento in funzione del tempo, otterremmo proprio i ben noti grafici delle onde sonore, come quelli nelle Figure 2.1 e 2.2. Ad esempio per riprodurre la nota La, che ha una frequenza di 440 *Hz* [12], lo speaker dovrà eseguire 440 oscillazioni "avanti-indietro" ogni secondo.

2.3.2 Campionamento

Si osservi che, per quanto detto prima, per oscillazione si intende un intero movimento "avanti-indietro", ovvero un intero periodo, quindi affinché il suono della nota La venga prodotto correttamente, lo speaker dovrà effettuare 440 movimenti in avanti e 440 movimenti indietro e sarà quindi necessario fornire alla scheda audio l'informazione su questi 880 movimenti totali. Il fatto di richiedere il doppio di informazione rispetto al numero di oscillazioni al secondo è noto come *Teorema del campionamento di Shannon-Nyquist.*

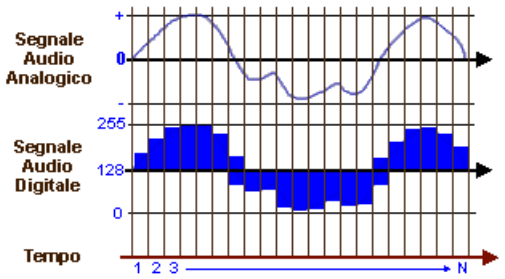


Figura 2.4: Campionamento di un segnale audio analogico.

Prima di parlare di questo importante teorema, introduciamo due concetti chiave del campionamento:

Definizione 2.2 *La* **sampling frequency** *(o* frequenza di campionamento*) è il numero di volte in cui viene misurato (o campionato) un segnale audio esterno in un determinato intervallo di tempo.*

Definizione 2.3 *La* **bit depth** *(o* profondità di bit*) indica il numero di bit riservati alla memorizzazione di ogni campione (o sample).*

Un microfono, in linea di massima, è sostanzialmente un altoparlante al contrario: una membrana collegata ad un elettromagnete si muove secondo i suoni esterni, assecondandone i movimenti. Ogni $1/f$ secondi (con f frequenza di campionamento) viene misurata la posizione di questa membrana rispetto alla posizione di quiete. Questa misura prende il nome di *campione* (o *sample*) e ognuno di questi campioni verrà registrato in memoria utilizzando un numero di bit definito dalla bit depth. Una volta registrati abbastanza campioni, effettuare la sintesi tramite scheda audio e speaker sarà semplice: verrà effettuata una semplice interpolazione tra i sample registrati e i valori della funzione risultante verrano dati in pasto alla scheda audio, che azionerà lo speaker di conseguenza. Ovviamente, una maggiore sampling frequency permette di avere campionamenti più fitti e quindi una riproduzione più accurata del suono registrato. Analogamente, più bit si riservano per i campioni, minore sarà l'errore commesso su ogni misura (detto *errore di quantizzazione*), rendendo ancora una volta la registrazione più accurata. Tuttavia, avere più sample e/o avere sample più voluminosi (come quantità di bit) andrà a incidere negativamente sulle dimensioni totali della registrazione risultante, rendendo, in casi estremi, il file finale eccessivamente grande.
Possiamo quindi trovare un compromesso tra dimensioni del file e qualità audio? Come vedremo più avanti, il punto di forza dell'MP3 sta proprio nell'aver definito un modo di memorizzare i file audio che permetta sia di godere di un'ottima qualità audio che, allo stesso tempo, di mantenere le dimensioni del file ridotte.

Il Teorema del campionamento di Shannon-Nyquist serve a mettere in relazione la frequenza di campionamento con la frequenza di output ottenuta dalla sintesi:

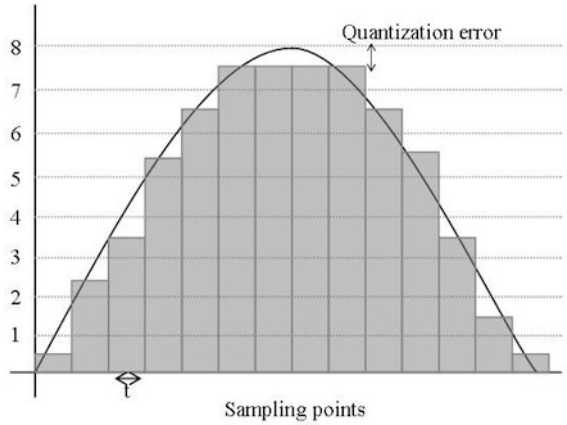


Figura 2.5: Campionamento ed errore di quantizzazione.

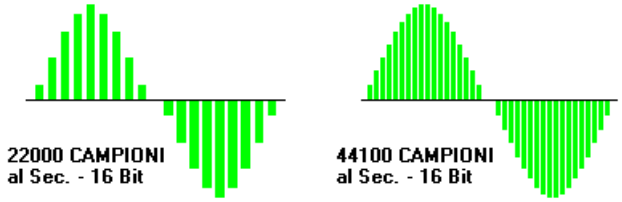


Figura 2.6: Uso di differenti sampling frequency: all'aumentare della sampling frequency aumenta la fedeltà del file audio ma anche le dimensioni complessive.

Teorema 2.1 (Shannon-Nyquist) *Per poter correttamente riprodurre un segnale audio, come minimo devono essere misurati due campioni per ogni periodo della sua onda caratteristica.*

Quindi come dicevamo prima, se vogliamo correttamente riprodurre una nota La (440 *Hz*), avremmo bisogno di una frequenza di campionamento di almeno 880 *Hz*. Viceversa, campionando con una frequenza fissata, non sarà possibile riprodurre tutte le frequenze maggiori della metà della sampling frequency (noto come *limite di Nyquist*).
La "qualità CD", infatti, è caratterizzata da una sampling frequency di 44.1 *kHz*, il che significa che verranno scartate tutte le frequenze al di sopra dei 22.05 *kHz*. Questo valore non è casuale: 22.05 *kHz* è poco sopra il limite superiore di udibilità dell'uomo (20 *kHz*). In parole povere, registrare frequenze più alte sarebbe soltanto uno spreco di spazio, in quanto non verrebbero comunque udite dall'uomo.

Se la frequenza di campionamento non è abbastanza elevata, alcuni campioni si potrebbero sovrapporre in frequenza, dando origine al fenomeno (negativo) detto *aliasing*: l'aliasing rende impossibile ricostruire parte del segnale originale a partire dai suoi campioni (come formalizzato dal Teorema di Shannon-Nyquist). L'aliasing si elimina aumentando la frequenza di campionamento. Se il segnale di input non è a banda limitata, allora si

filtra prima attraverso un filtro passa-basso, detto *filtro anti-aliasing*.

Capitolo 3

Storia: dal PCM all'MP3

Avevamo detto, nel capitolo introduttivo, che l'MP3 è sì un formato di file audio, ma anche che con MP3 si intende principalmente un algoritmo di compressione audio (di tipo lossy). Viene allora spontaneo chiedersi a che tipo di formato audio si applichi la compressione MP3, ovvero che formato audio era predominante prima dell'arrivo dell'MP3.

3.1 PCM

La risposta a queste domande è racchiusa nella sigla *PCM*. *PCM*, che sta per ***P**ulse **C**ode **M**odulation* (ovvero *Modulazione a Codice di Impulsi*), è sostanzialmente il metodo di campionamento descritto nel Capitolo 2: per registrare e memorizzare un segnale audio esterno si eseguono delle misure a intervalli di tempo regolari (dettati dalla sampling frequency), quindi si quantizzano i sample ottenuti (ovvero si mappano in valori discreti) ed infine si digitalizzano (ovvero si codifica ogni campione in codice binario).
In sostanza, il metodo PCM tende a catturare un segnale audio esterno registrandone e memorizzandone una serie di valori, in modo da poter approssimare e ricostruire l'onda caratteristica ogni qual volta si voglia riprodurre quel suono.

3.1.1 Il problema delle dimensioni

Ovviamente la PCM può operare a diversi valori di sampling frequency e di bit depth, anche se l'accoppiata più famosa è quella relativa alla qualità CD, ovvero 44.1 *kHz* di sampling frequency e 16 bit come bit depth.
Questo significa che ogni secondo vengono registrati 44100 campioni, ognuno dei quali grande 16 bit, ovvero 88200 byte al secondo [1]. Questo valore raddoppia se l'audio registrato è stereo (una registrazione per il canale sinistro ed una per il canale destro). Con queste premesse, è facile vedere che un minuto di audio stereo in PCM occupa poco più

[1] Calcolato come: $(44100 \cdot 16)/8 = 88200$ byte .

di 10 MB. Quindi un normale CD da 700 MB può contenere soltanto 70 minuti di audio in questo formato, che corrispondono a circa 20 canzoni [1]

Fintanto che si parla di CD audio, queste limitazioni non presentano un vero problema. Tuttavia quando si parla di applicazioni, che registrano su computer grandi quantità di audio, e specialmente di applicazioni su Web, per le quali è necessario scambiare file audio a grande velocità, le dimensioni dei file iniziano a pesare e a diventare un problema non indifferente.

Un primo approccio per diminuire le dimensioni dei file audio è quello di diminuire la sampling frequency: meno campioni significa meno spazio necessario. Se dimezziamo la frequenza di campionamento, allora avremo esattamente la metà dei campioni e quindi un file grande la metà rispetto all'originale. Tuttavia, per il Teorema di Shannon-Nyquist (Teorema 2.1 a pagina 14), sappiamo che il limite superiore delle frequenze riproducibili è dato dalla metà della sampling frequency, quindi dimezzando quest'ultima dimezzeremo di fatto anche il limite di Nyquist, scartando metà frequenze di quelle che prima venivano registrate. Se ad esempio dimezziamo la sampling frequency della qualità CD, da 44.1 *kHz* a 22.05 *kHz*, avremo che verranno scartate non più le frequenze maggiori di 22.05 *kHz*, ma tutte quelle maggiori di 11.025 *kHz*. Mentre quelle maggiori di 22 *kHz* non ci interessano più di tanto (per via del range di udibilità umano 20 *Hz* - 20 *kHz*) quelle comprese tra 11.025 e 22.05 *kHz* invece sono frequenze importanti, in quanto udibili dall'uomo. Perdere queste frequenze risulterebbe, inevitabilmente in una perdita di qualità del segnale audio.

Un secondo approccio è invece quello di diminuire la bit depth, ovvero di assegnare meno bit ad ogni campione registrato. Diminuendo i bit per ogni sample ovviamente diminuiscono anche le dimensioni totali, ma aumenta anche l'errore di quantizzazione, in quanto meno bit devono essere usati per approssimare una grandezza continua. In questo frangente, tuttavia, le perdite relative alla diminuzione di bit depth sono molto più gravi rispetto alla diminuzione della sampling frequency. Infatti, dimezzando la bit depth di ogni sample le dimensioni totali saranno sì dimezzate, tuttavia la qualità del suono risultante sarà più che dimezzata. Se pensiamo ad un campionamento a 16 bit, ricaviamo che ogni sample può assumere uno di 65536 (2^{16}) diversi valori, che è come dire che la misura ottenuta può essere rappresentata utilizzando 65536 "sfumature" diverse. Tuttavia utilizzando 8 bit per ogni sample, abbiamo che ogni campione potrà assumere uno di 256 (2^8) possibili valori, che è molto meno della metà.
Diminuire la bit depth porta quindi ad un aumento esponenziale degli errori di quantizzazione, diminuendo il cosiddetto *rapporto segnale/rumore*.

3.2 MP3

Con queste premesse, la *ISO* (*International Standards Organization*) e la *IEC* (*International Electrothecnical Commision*) formarono il gruppo *MPEG* (ovvero *Moving Picture*

[1] Considerando una durata media per canzone di 3.5 minuti.

Experts Group), nella speranza di definire uno standard mondiale per la codifica di video e audio di alta qualità. In particolare, si erano proposti come obiettivo quello di definire un metodo di codifica audio e video che permettesse la scrittura e la lettura da un generico dispositivo capace di fornire 1.5 Mbit al secondo (come i ben noti CD).
Un punto cruciale divenne allora la compressione di questi dati, in quanto la lettura e scrittura di dati audio e video non compressi richiederebbe molto di più di 1.5 Mbit al secondo, infatti il solo audio codificato tramite PCM richiede un bitrate di 1.4 *Mbps* per la sua riproduzione [1].

3.2.1 Lo standard MPEG 1

Lo standard *MPEG 1*, riguardante appunto la compressione di video e dell'audio associato, fu pubblicato dal gruppo MPEG nel 1993, sotto il nome di *ISO/IEC 11172*. La parte relativa alla compressione audio dell'MPEG 1 fu suddivisa in tre *Layer*, o *livelli*: questi livelli aggiungono, ognuno, nuovi meccanismi di compressione audio, rendendo la compressione stessa più complessa ed efficiente ad ogni livello. Inoltre questi layer sono retrocompatibili, ovvero un decoder costruito per il Layer 3 riesce a decodificare anche file codificati con il Layer 2 e con il Layer 1. Quello che noi oggi chiamiamo MP3 è, di fatto, tutto l'algoritmo di compressione dell'MPEG 1 Layer 3.
Successivamente, nel 1995, una seconda versione dello standard MPEG venne pubblicata, chiamata comunemente *MPEG 2* e formalmente *ISO/IEC 13818*. Ai fini delle tecniche caratteristiche dell'MP3 è indifferente parlare di MPEG 1 o 2, infatti spesso ci si riferisce all'MP3 come *MPEG 1/2 Layer 3*.

Ogni layer dello standard MP3 aggiunge complessità all'algoritmo, riuscendo ad ottenere una compressione sempre più efficace. In Tabella 3.1 possiamo vedere i fattori di compressione MPEG 1 rispetto alla codifica PCM e il bitrate richiesto per la riproduzione di un file audio all'aumentare della complessità di compressione.

Codifica	**Rapporto**	**Bitrate richiesto**
PCM (qualità CD)	1:1	1.4 *Mbps*
Layer 1	4:1	384 *kbps*
Layer 2	8:1	192 *kbps*
Layer 3	12:1	128 *kbps*

Tabella 3.1: Bitrate richiesti per le codifiche MPEG 1.

Si ha quindi che la codifica MP3 riesce a diminuire le dimensioni di un file audio, codificato con la PCM, di un fattore 12, diminuendo notevolmente le dimensioni originali, senza tuttavia intaccarne notevolmente la qualità, grazie ai principi della psicoacustica che vedremo più avanti.

[1] Calcolato come $44.1\ kHz \cdot 16\ bit \approx 0.7\ Mbps$ per audio mono, 1.4 *Mbps* per audio stereo (il doppio).

3.2.2 Libertà d'implementazione

Le specifiche dello standard ISO 11172-3 (ovvero dell'MP3) definiscono come dev'essere strutturato e interpretato un file `.mp3` e come deve avvenire la decodifica di quest'ultimo. Questo è ciò che rende l'MP3 uno standard internazionale: un file `.mp3`, infatti, potrà essere decodificato da qualsiasi decoder MP3.
Tuttavia, molte fasi di codifica e dettagli implementativi non sono specificati nello standard ISO, lasciando agli sviluppatori una discreta libertà di scelta per quanto riguarda le varie parti dell'encoder. In altre parole, lo standard definisce soltanto cosa si deve ottenere dalla codifica, ma non come eseguire la codifica stessa: diversi sviluppatori possono utilizzare tecniche diverse per ottenere lo stesso risultato, o addirittura impiegare nuove tecniche che migliorano/velocizzano il processo di codifica.
Il processo di decodifica è invece già più standardizzato, anche se molti dettagli implementativi vengono lasciati allo sviluppatore finale.

3.2.3 Alcuni dettagli tecnici

Concludiamo esponendo alcuni dettagli tecnici dello standard MP3.

Innanzitutto parliamo di *bitrate*. Il *bitrate* (letteralmente *tasso di bit*) indica quanti bit di memoria sono dedicati ad ogni secondo del segnale audio, ovvero quanti bit al secondo devono essere letti per la sua riproduzione. Ovviamente più bit al secondo implica una maggior qualità del suono ma anche maggiori dimensioni del file. Lo standard MP3 prevede bitrate dagli 8 *kbps* fino ai 320 *kbps*, anche se il valore più utilizzato è 128 *kbps*.
In linea di massima, lo standard MPEG 1 Layer 3 definisce due tipologie di bitrate: il *bitrate costante* (*Constant BitRate*, *CBR*) e il *bitrate variabile* (*Variable BitRate*, *VBR*).
Con il **bitrate costante** si assegna lo stesso numero di bit a ogni secondo del file audio. Tuttavia molte registrazioni (ad esempio canzoni) variano spesso di complessità, passando da momenti in cui suonano molti strumenti (e quindi più bit sono richiesti per avere una buona fedeltà) ad altri più semplici (in cui basterebbero pochi bit al secondo). Con il **bitrate variabile** si possono invece assegnare a diverse parti della canzone (dette *frame*, come vedremo più avanti) una quantità diversa di bit, a seconda della complessità del segnale.

La qualità finale del file `.mp3` è anche data dalla sampling frequency (come già avevamo visto per la PCM): più campioni al secondo significa una più alta qualità audio, ma anche un file più voluminoso. Lo standard MP3 prevede la codifica a 32, 44.1 o 48 *kHz*.

Infine, un file `.mp3` avrà una delle seguenti *modalità di canale* (*channel mode*):

- canale singolo (mono);
- canale doppio;
- stereo;

- joint stereo.

Il *canale singolo* (o *mono*) è il più semplice di tutti e prevede un singolo flusso di dati audio. Il *canale doppio* prevede invece due canali singoli indipendenti (uno destro ed uno sinistro). La modalità *stereo* prevede invece due canali non indipendenti, in quanto i bit per ogni canale posso essere ripartiti tra i due (se ad esempio in un certo momento il segnale sinistro è più complesso di quello destro, si possono prendere alcuni bit di quest'ultimo). Con *joint stereo* si intende invece una modalità che tenta di ottimizzare la codifica dei due canali eliminando le ridondanze tra questi. Esistono due tecniche per il joint stereo: *mid/side stereo* (o *MS stereo*) e *intensity stereo*. Mid/side stereo è composto da due canali: *mid*, che rappresenta le parti in comune tra canale destro e sinistro, e *side*, che rappresenta le differenze tra canale destro e sinistro. In intensity stereo invece le frequenze più alte vengono sommate tra loro, in modo da riuscire a trasmettere tutto in un unico canale. MS stereo è una tecnica loosless, mentre intensity stereo è una tecnica lossy, anche se le inconsistenze introdotte sono spesso non percepibili all'orecchio umano.

Parte II

MP3

Capitolo 4

Introduzione all'MP3

Abbiamo visto, nei capitoli precedenti, che il metodo di campionamento della PCM aveva come obiettivo quello di riprodurre, il più fedelmente possibile, l'onda caratteristica di un segnale audio esterno. Tuttavia questo approccio racchiude in sé un'assunzione implicita, ovvero che sia necessario disporre dell'onda caratteristica per poter riprodurre correttamente un segnale audio. Quest'assunzione nasce sostanzialmente da una scarsa conoscenza della percezione del suono da parte dell'orecchio e del cervello umano.

4.1 Codifica percettiva

Il problema di fondo è che l'orecchio ed il cervello umano sono "strumenti" imperfetti, che non captano il suono in maniera perfetta e assoluta per come è, ma lo interpretano secondo una serie di fattori. Un esempio è rappresentato dalla soglia assoluta di udibilità umana (in Figure 2.3). È stato dimostrato, inoltre, che raddoppiare l'ampiezza di un segnale audio non corrisponde ad un raddoppiamento dell'effettivo volume percepito. In generale, esistono molti aspetti del suono che l'orecchio ed il cervello umano semplicemente trascurano, accentuando alcune proprietà dei suoni esterni anziché percepire tutto in modo assoluto.

Come già accennato, la disciplina che si occupa dello studio della percezione del suono prende il nome di *psicoacustica* ed assume un ruolo chiave nella tecnologia MP3.
L'idea di base dell'MP3 è la seguente: se alcune caratteristiche del suono non vengono comunque percepite dall'uomo, perché sprecare spazio inutilmente cercando di riprodurre tutta l'onda caratteristica? Sarebbe molto più intelligente cercare di memorizzare e riprodurre soltanto quelle parti del suono che effettivamente verrano percepite.
Infatti quello che fa l'MP3 è proprio di selezionare le caratteristiche più rilevanti di un certo suono ed assegnargli una maggiore priorità, in modo da riprodurle più fedelmente a scapito di caratteristiche meno rilevanti o superflue. Si potrebbe dire che, mentre la

PCM tenta di catturare un segnale audio "per come è", l'MP3 tenta di catturarlo "per come suona".

Risulta allora necessario definire cosa è rilevante per l'orecchio umano e cosa non lo è: queste informazioni costituiscono il cosiddetto **modello psicoacustico**, che definisce quanto e quali parti del suono sono rilevanti.
Prima di entrare nel dettaglio del modello psicoacustico, è necessario definire due concetti chiave: *ridondanza* e *irrilevanza*, che suddividono in due categorie distinte tutte quelle informazioni del suono considerate non necessarie per l'uomo, o comunque eliminabili senza perdere troppo in termini di qualità audio.
Abbiamo già parlato di **ridondanza** quando abbiamo visto il metodo PCM ed abbiamo parlato del limite di Nyquist: nella qualità CD (frequenza di campionamento di 44.1 *kHz*) tutte le frequenze maggiori di 22.05 *kHz* vengono automaticamente considerate ridondanti, e quindi scartate. Si potrebbe aumentare la sampling frequency, per catturare più alte frequenze, ma il limite in questione non verrebbe eliminato, ma soltanto spostato più in alto. In altre parole, la ridondanza è un concetto onnipresente nel mondo digitale.
L'**irrilevanza**, invece, è un concetto molto più complesso, che descrive tutte quelle caratteristiche di un'onda sonora che sono prive di senso in termini di percezione umana. Tutte queste caratteristiche, irrilevanti per l'uomo, verranno descritte in un opportuno modello psicoacustico e quindi eliminate (o ridotte di dimensioni) dal file audio finale, diminuendo le dimensioni di quest'ultimo, senza tuttavia influenzarne troppo la qualità, in quanto le informazioni scartate verrebbero comunque ignorate dal cervello umano.

4.2 Effetto di mascheramento

Il modello psicoacustico utilizzato nella codifica MP3 è basato su un'importante fenomeno, caratteristico dell'orecchio umano, detto *mascheramento* (o *masking* in inglese). Prima di definire cos'è il mascheramento, ricordiamo di seguito il *principio di sovrapposizione* per onde sonore:

In un punto dello spazio in cui giungono due o più suoni simultanei, il suono risultante è dato dalla somma (algebrica) dei due (o più) suoni incidenti.

Quindi sappiamo che quando due (o più) suoni giungono al nostro orecchio contemporaneamente, quello che noi percepiamo di fatto è la sovrapposizione di questi due (o più) suoni. Questo principio può essere formulato anche al contrario, affermando che ogni suono percepito può essere visto come somma (algebrica) di due o più suoni.
Il nostro cervello e orecchio sono in grado di eseguire, ogni volta che percepiscono un suono esterno, un'analisi spettrografica su di esso, per cercare di ricavarne i suoni generatori. È sempre possibile, per il nostro orecchio, ricavare e distinguere le componenti di un suono composto? La risposta è no, in quanto l'orecchio umano è adattivo e, a seconda della situazione, percepisce stessi suoni in modo diverso. Generalmente, l'orecchio riesce a ricavare le componenti dei suoni che percepisce, distinguendone le varie frequenze, ad eccezione dei seguenti quattro casi:

- quando due suoni simultanei hanno altezze molto simili (*battimenti*);
- quando uno dei due suoni è molto più forte dell'altro (*mascheramento simultaneo*);
- quando un suono molto forte precede di poco un suono più debole (*mascheramento temporale in avanti*);
- quando un suono molto forte segue di poco un suono più debole (*mascheramento temporale all'indietro*).

Quindi l'effetto di **mascheramento** è appunto quando la somma di più componenti sonore fa scomparire una delle componenti (mascheramento simultaneo e temporale) o produce un suono totalmente nuovo (battimenti).

Il **mascheramento simultaneo** (o *mascheramento a dominio di frequenza*) si ha quando percepiamo un suono predominante assieme ad uno più debole. Un classico esempio è quando proviamo a parlare mentre passa un treno vicino, o c'è un altro rumore forte di sottofondo: il rumore del treno è predominante e verrà percepito più chiaramente, mentre la voce, più debole, verrà percepita con più difficoltà. Più formalmente, si è dimostrato che i suoni percepiti dall'orecchio umano sono suddivisi in 24 *bande critiche*: se più frequenze della stessa banda critica arrivano all'orecchio, quest'ultimo non sarà in grado di distinguere le frequenze originali o comunque verrà percepita soltanto la componente dominante. In Figura 4.1 possiamo vedere come un suono più forte modifichi la soglia di udibilità, rendendo muti suoni prima percepibili. Quando si verifica il mascheramento simultaneo, la soglia di udibilità prende il nome di *soglia di mascheramento* (o *masking threshold* in inglese).
Il **mascheramento temporale** (o *mascheramento a dominio di tempo*) invece si ha quando un suono predominante precede o segue, in un breve lasso di tempo, un suono più debole. La soglia di tempo necessaria affinché si verifichi il mascheramento temporale è circa 50 *ms* per il mascheramento all'indietro e dai 50 ai 300 *ms* per il mascheramento in avanti [14] (si veda la Figura 4.2). Possiamo immaginarci il mascheramento temporale pensando ad un ambiente silenzioso in cui improvvisamente si battono le mani: inizialmente il suono delle mani verrà percepito come più forte. Analogamente, se subito prima del battito di mani venisse sparato un colpo di pistola, il battito di mani stesso apparirebbe più attenuato.

L'effetto di mascheramento può essere visto come un'imperfezione dell'orecchio umano, tuttavia rappresenta il punto di forza dell'MP3 e della codifica percettiva: se in un certo lasso di tempo alcune parti del suono vengono percepite con più difficoltà rispetto ad altre, allora possiamo pensare di assegnare più bit per l'informazione del suono predominante e meno per le parti nascoste dal mascheramento. In questo modo l'errore (o distorsione) relativo alle parti mascherate verrà comunque percepito poco, senza degradare più di tanto la qualità complessiva.

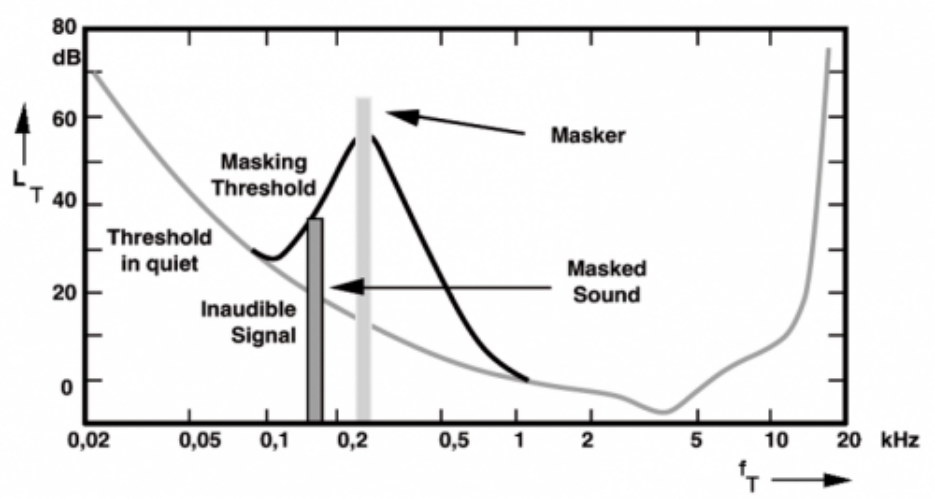


Figura 4.1: Soglia di mascheramento.

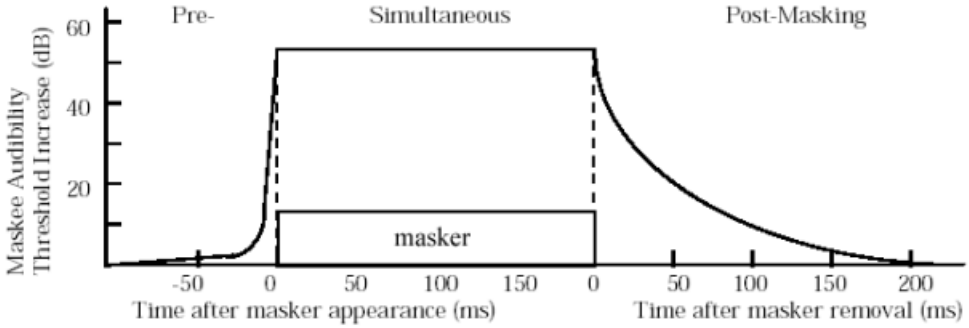


Figura 4.2: Mascheramento temporale.

Capitolo 5

Struttura di un file `.mp3`

Prima di andare ad analizzare come funziona la codifica (e decodifica) MP3, studiamo com'è fatto un file `.mp3`, ovvero cosa ci si aspetta di ottenere dalla codifica.

Un file `.mp3` è suddiviso in parti dette *frame*. Ogni frame contiene 1152 campioni e dura circa 26 *ms*, il che significa che verranno riprodotte circa 38 *fps* (*frame per secondo*). Inoltre, ogni frame è suddiviso in due *granuli* di 576 sample l'uno.
La dimensione in *byte* di ogni frame dipende sia dal bit rate che dalla sampling frequency scelti, secondo la seguente formula:

$$\text{dimensione frame} = \frac{144 \cdot \text{bitRate}}{\text{samplingRate}} + \text{padding}. \tag{5.1}$$

Il bit di *padding* (allocato all'inizio di ogni frame) è utilizzato per rispettare l'esatto bitrate per tutti i frame (dato che dividendo il bitrate per il numero di frame per secondo si potrebbe non ottenere un intero). Se è impostato ad uno allora la dimensione del frame corrispondente viene aumentata di un byte.
Si tenga ben presente che la dimensione di ogni frame dev'essere sempre un intero, in caso contrario si effettuerà un arrotondamento.

5.1 Struttura dei frame

La struttura generale di ogni frame è mostrata in Figura 5.1.

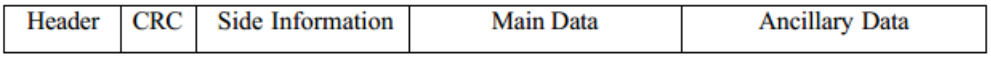


Figura 5.1: Struttura di un frame.

Come possiamo vedere, ogni frame è suddiviso in:

- *Header*: detto anche *Intestazione*, è la parte iniziale di ogni frame;
- *CRC*: *Cyclic Redundancy Check*, controlla che non ci siano errori nel frame;
- *Side Information*: informazione necessaria per decodificare la parte audio del frame;
- *Main Data*: gli effettivi dati audio da decodificare;
- *Ancillary Data*: detto anche *Dati Ausiliari*, sono dati opzionali destinati a funzioni accessorie.

5.1.1 Header

L'header del frame MP3 ha una lunghezza fissa di 32 bit. La Figura 5.2 mostra la struttura di un generico frame header.

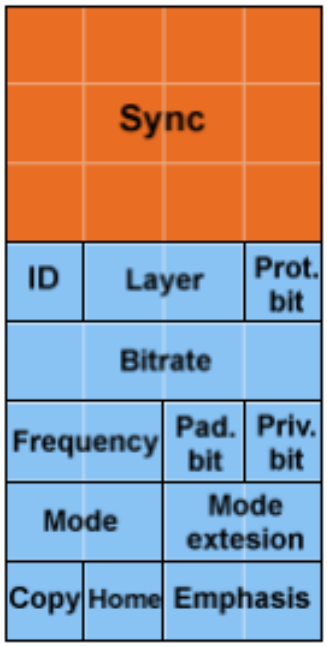


Figura 5.2: Struttura dell'header del frame.

Analizziamo di seguito i vari campi che compongono l'header:

- *Sync* (12 bit): questi bit di sincronizzazione sono sempre impostati tutti a 1 e servono a determinare dove inizia un nuovo frame.
- *ID* (1 bit): specifica la versione MPEG. Se il bit è asserito significa che il frame è stato codificato con MPEG-1, altrimenti con MPEG-2. Alcuni encoder MP3,

inoltre, allocano soltanto 11 bit per la parte Sync per poterne utilizzare 2 per l'ID, secondo la Tabella 5.1.

bit	**Versione MPEG**
00	MPEG-2.5 (aggiornamento dell'MPEG-2)
01	Riservato
10	MPEG-2
11	MPEG-1

Tabella 5.1: Valori del campo ID a 2 bit.

- *Layer* (2 bit): come per il campo ID, i 2 bit del campo Layer indicano quale Layer dello standard MPEG si sta utilizzando, secondo la Tabella 5.2.

bit	**Layer MPEG**
00	Riservato
01	Layer III
10	Layer II
11	Layer I

Tabella 5.2: Valori del campo Layer.

- *Protection bit* (1 bit): se asserito, allora viene utilizzato il campo CRC per la rilevazione di errori nel frame.
- *Bitrate* (4 bit): questi quattro bit, assieme ai campi ID e Layer, informano il decoder sul bitrate richiesto, secondo la Tabella 5.3. Ovviamente se il file `.mp3` è codificato con CBR (Constant BitRate), allora tutti i frame avranno lo stesso valore nel campo Bitrate.
- *Frequency* (2 bit): questi 2 bit indicano la sampling frequency utilizzata, come indicato in Tabella 5.4.
- *Padding bit* (1 bit): come già accennato riguardo alla dimensione dei frame, se con la (5.1) si ottiene un numero decimale, per alcuni frame si arritonderà per eccesso, mentre per altri per difetto. Se ad esempio si utilizza un bitrate di 128 *bps* ed una sampling frequency di 44.1 *kHz*, si avrà la dimensione dei frame pari a circa 417.95 byte, ovvero si avranno frame da 417 e frame da 418, per rispettare il bitrate indicato. Se il bit di padding è settato, allora il frame attuale è stato arrotondato per eccesso.
- *Private bit* (1 bit): bit a uso specifico delle applicazioni.
- *Mode* (2 bit): riprendendo quanto detto riguardo alle modalità di canale, i bit Mode indicheranno la modalità del frame attuale secondo la Tabella 5.5.

bit	MPEG-1			MPEG-2		
	Layer I	**Layer II**	**Layer III**	**Layer I**	**Layer II**	**Layer III**
0000						
0001	32	32	32	32	32	8
0010	64	48	40	64	48	16
0011	96	56	48	96	56	24
0100	128	64	56	128	64	32
0101	169	80	64	160	80	64
0110	192	96	80	192	96	80
0111	224	112	96	224	112	56
1000	256	128	112	256	128	64
1001	288	160	128	288	160	128
1010	320	192	160	320	192	160
1011	352	224	192	352	224	112
1100	384	256	224	384	256	128
1101	416	320	256	416	320	256
1110	448	384	320	448	384	320
1111						

Tabella 5.3: Valori del campo Bitrate (i valori in rosso rappresentano la scelta più frequente).

bit	MPEG-1	MPEG-2	MPEG-2.5
00	44.1 *kHz*	22.05 *kHz*	11.025 *kHz*
01	48 *kHz*	24 *kHz*	12 *kHz*
10	32 *kHz*	16 *kHz*	8 *kHz*
11	Riservato		

Tabella 5.4: Valori del campo Frequency.

bit	Channel Mode
00	Stereo
01	Joint Stereo
10	Canale Doppio
11	Canale Singolo

Tabella 5.5: Valori del campo Mode.

- *Mode Extension* (2 bit): bit utilizzati soltanto nel caso in cui la modalità di canale sia impostata su joint stereo, nel qual caso il campo Mode Extension indicherà se viene utilizzata la tecnica MS stereo, intensity stereo o una combinazione delle due (si veda la Tabella 5.6).
- *Copy* (1 bit): se questo bit è asserito, allora significa che il contenuto è coperto da copyright ed è illegale copiarlo.

bit	MS stereo	Intensity stereo
00	off	off
01	off	on
10	on	off
11	on	on

Tabella 5.6: Valori del campo Mode Extension.

- *Home* (1 bit): se asserito, indica che il frame si trova nel file originale.
- *Emphasis* (2 bit): bit di enfasi che indicano come ri-equalizzare il suono dopo un'eliminazione del rumore di tipo Dolby (raramente usati). Per conoscenza, si veda la Tabella 5.7.

bit	Modello di soppressione del rumore
00	Nessuno
01	50/15 *ms*
10	Riservato
11	CCITT J.17

Tabella 5.7: Valori del campo Emphasis.

5.1.2 CRC

Il campo CRC (di 0 o 16 bit) è una Cyclic Redundancy Check sui dati sensibili del frame, per evitare che contengano errori. Se il bit Protection nell'header è asserito, allora il campo CRC conterrà una checksum dei bit dal 16 al 32 dell'header e della side information (che secondo lo standard contengono i dati più sensibili del frame). Se un frame risulta danneggiato, può essere mutato o rimpiazzato dal frame precedente.

5.1.3 Side Information

Il campo Side Information contiene tutte quelle informazioni necessarie al decoder per decodificare i dati audio di ogni frame. Se si utilizza un canale singolo, allora questo campo sarà lungo 17 byte, altrimenti 32. La struttura del campo Side Information è mostrato in Figura 5.3.

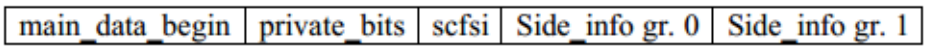


Figura 5.3: Struttura del campo Side Information.

Analizziamo di seguito i campi della Side Information. Dal momento che le dimensioni dei vari campi possono variare a seconda della modalità di canale scelta, tra parentesi

verranno indicate le dimensioni per la modalità mono (primo valore) e le dimensioni per tutte le altre modalità (secondo valore). Se invece è indicato un solo valore, la dimensione di quel campo è costante. Inoltre, tutte le tabelle si riferiscono alla modalità mono.

- *main_ data_ begin* (9 bit): utilizzando il formato del Layer III, viene impiegata una tecnica detta *bit reservoir* (letteralmente, *deposito di bit*), che consente di sfruttare lo spazio inutilizzato nel campo Main Data di un frame da frame consecutivi. Il campo main_data_begin è un numero negativo che indica quanti byte prima (rispetto al primo bit della synchronization word) iniziano i dati principali, escludendo dal conteggio le parti statiche di un frame, come l'header. Essendo un campo a 9 bit, i dati principali possono iniziare fino a $(2^9 - 1) \cdot 8 = 4088$ bit prima, che corrisponde a svariati frame. Se il campo è impostato a 0, allora i dati principali iniziano subito dopo la Side Information.

- *private_ bits* (5 bit, 3 bit): bit privati destinati all'utilizzo specifico da parte delle applicazioni.

- *scfsi* (4 bit, 8 bit): la *ScaleFactor Selection Information* (o *Informazione sulla Selezione dei Fattori di Scala*) indica se dei certi *scalefactor* (*fattori di scala*, vedremo più avanti cosa sono) sono condivisi oppure no tra i due granuli di un frame. Le bande di fattori di scala sono suddivise in 4 gruppi, secondo la Tabella 5.8. In questo campo, vengono trasmessi 4 bit per ogni canale: se il bit del gruppo è asserito, allora quelle scalefactor band valgono sia per il granulo 0 che per il granulo 1, così da poterle trasmettere una sola volta.

Gruppo	**Scalefactor Band**
0	0, 1, 2, 3, 4, 5
1	6, 7, 8, 9, 10
2	11, 12, 13, 14, 15
3	16, 17, 18, 19, 20

Tabella 5.8: Gruppi delle scalefactor band

- *Side_ info*: le ultime due parti della Side Information ("Side_info gr.0" e "Side_info gr.1") contengono le informazioni per la decodifica dei due granuli del frame e sono strutturalmente identiche (si veda la Figura 5.4).

part2_3_length	big_values	global_gain	scalefac_compress
windows_switching_flag	block_type	mixed_block_flag	table_select
subblock_gain	region0_count	region1_count	preflag
scalefac_scale	count1table_select		

Figura 5.4: Struttura della Side Information per ogni granulo.

Di seguito, l'analisi dei sottocampi che compongono la Side Information di ogni granulo:

- *part2_3_length* (12 bit, 24 bit): indica quanti bit allocare nel campo Main Data per i fattori i scala (part2) e per i dati codificati con Huffman (part3) (più avanti vedremo com'è composto il campo Main Data).
- *big_values* (9 bit, 18 bit): i 576 sample contenuti in ogni granulo non sono necessariamente codificati tutti con la stessa tabella di Huffman. In particolare, le frequenze di questi sample, che spaziano da 0 al limite di Nyquist, verranno suddivise in 5 regioni (Figura 5.5), ognuna della quali codificata con una diversa tabella di Huffman. L'obiettivo è quello di rendere la codifica di Huffman ancora più efficiente codificando in modo diverso diverse parti dello spettro.

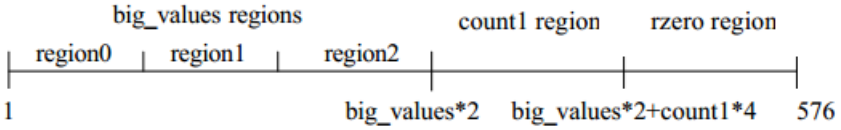


Figura 5.5: Regioni di frequenza.

Nella regione rzero si trovano le frequenze più alte, divise a coppie. Nella regione count1 si trovano frequenze più basse di quelle in rzero, suddivise in gruppi di quattro. Infine, nelle regioni big values si trovano tutte le altre frequenze che arrivano fino a zero. Il massimo valore assoluto di ampiezza per i sample nelle regioni big values è limitato a 8191. Dal momento che il campo big_values determina la grandezza delle regioni big values, il massimo valore possibile per questo campo è 288 [1].

- *global_gain* (8 bit, 16 bit): specifica la grandezza dello step di quantizzazione.
- *scalefac_compress* (4 bit, 8 bit): determina il numero di bit utilizzati per i fattori di scala. Ogni granulo può essere diviso in 12 (per finestre corte) o 21 (per finestre lunghe) bande di fattori di scala. Non spiegheremo qui cosa si intende con finestre, in quanto verrà spiegato più ampiamente all'interno del Capitolo 6. Le bande di fattori di scala sono successivamente suddivise in due parti (0-6, 7-11 per finestre corte e 0-10, 11-20 per finestre lunghe), rispettivamente lunghe *slen1* e *slen2*. A seconda del valore del campo scalefac_compress, verranno assegnati più o meno bit alle due parti, come mostrato in Tabella 5.9.
- *windows_switching_flag* (1 bit, 2 bit): se asserito, indica che sarà utilizzata una finestra diversa da quella di default (si veda Capitolo 6). Inoltre quando

[1] Calcolato come 576/2, in quanto la dimensione della regione di big values è calcolata come $big\_values \cdot 2$.

bit	slen1	slen2
0000	0	0
0001	0	1
0010	0	2
0011	0	3
0100	3	0
0101	1	1
0110	1	2
0111	1	3
1000	2	1
1001	2	2
1010	2	3
1011	3	1
1100	3	2
1101	3	3
1110	4	2
1111	4	3

Tabella 5.9: Valori del campo scalefac_compress.

è asserito, tutti i valori (big values) che non sono contenuti nella region0 sono contenuti nella region1, rendendo la region2 di fatto vuota.

– *block_type* (2 bit, 4 bit): se il bit windows_switching_flag è asserito, allora questo campo serve a indicare il tipo di finestra utilizzato per il granulo corrente. I valori 1 e 3 sono finestre lunghe, mentre il valore 2 indica una finestra corta. Il valore 0 indicherebbe la finestra normale di default (lunga), ma dato che quando si usa la finestra di default il campo block_type non viene utilizzato, questo valore è vietato. Vediamo nel dettaglio il significato di questi valori in Tabella 5.10 (per i vari tipi di finestra si rimanda sempre al Capitolo 6).

bit	Tipo finestra
00	Vietato
01	Finestra inizio
10	3 finestre corte
11	Finestra fine

Tabella 5.10: Valori del campo block_type.

– *mixed_block_flag* (1 bit, 2 bit): se asserito, indica che le due sottobande più basse (si veda il Capitolo 6 per quanto riguarda le sottobande) sono trasformate utilizzando la finestra normale, mentre le restanti 30 utilizzando la finestra

indicata dal campo block_type. Questo campo viene utilizzato soltanto se il bit windows_switching_flag è asserito.

- *table_select* (10 bit, 20 bit) o (15 bit, 30 bit): lo standard MP3 definisce 32 possibili tabelle di Huffman con cui codificare i sample. Essendo le tabelle possibili 32, servono 5 bit (per canale, granulo e regione) per individuare univocamente una tabella. Il campo table_select definisce con quali tabelle di Huffman decodificare i sample delle regioni big values. Quindi se windows_switching_flag è settato a 1 la regione region2 sarà vuota e si avranno 10 bit (in modalità mono) necessari per l'individuazione delle tabelle di Huffman (20 se in modalità stereo). Se invece il bit è negato si avrà bisogno di altri 5 bit per canale per la tabella relativa alla regione region2, ovvero 15 bit per mono e 30 per stereo.
- *subblock_gain* (9 bit, 18 bit): se windows_switching_flag=1 e block_type=10, allora questo campo definisce, 3 bit alla volta, l'offset da aggiungere al global_gain per ogni sottoblocco.
- *region0_count*(4 bit, 8 bit): in questo campo viene indicato il numero di bande di fattori di scala presenti nella regione region0 diminuito di 1. Quindi se ad esempio region0_count=8, allora nella regione region0 ci sono 9 bande di fattori di scala. Inoltre se utilizzano finestre corte (block_type=10), allora in questo campo vengono contate le bande dei fattori di scala per ogni sottoblocco: se, ad esempio, in questo caso avessimo sempre region0_count=8, allora nella regione region0 ci sarebbero 9/3=3 bande di fattori di scala. Questo campo viene utilizzato per ricavare gli estremi della regione region0, infatti questi estremi vengono allineati (nella partizione dello spettro in regioni) per combaciare con la suddivisione delle frequenze in bande di fattori di scala.
- *region1_count*(3 bit, 6 bit): esattamente come il campo region0_count, solo rispetto alla regione region1.
- *preflag*(1 bit, 2 bit): è un bit che permette l'amplificazione di alte frequenze quantizzate. Se asserito, i valori di una tabella predefinita vengono sommati alle bande di fattori di scala. Se si usano finestre corte (block_type=10) questo campo non si usa mai.
- *scalefac_scale*(1 bit, 2 bit): i fattori di scala sono quantizzati logaritmicamente con uno step di quantizzazione pari a 2 o $\sqrt{2}$, secondo la Tabella 5.11.

bit	**Dimensione step**
0	$\sqrt{2}$
1	2

Tabella 5.11: Valori del campo scalefac_scale.

- *count1table_select* (1 bit, 2 bit): questo bit seleziona una delle due possibili tabelle di Huffman per la decodifica della regione count1.

5.1.4 Main Data

La parte di dati principali consiste nei fattori di scala e nei sample quantizzati codificati con Huffman. Lo schema in Figura 5.6 rappresenta la struttura del campo Main Data.

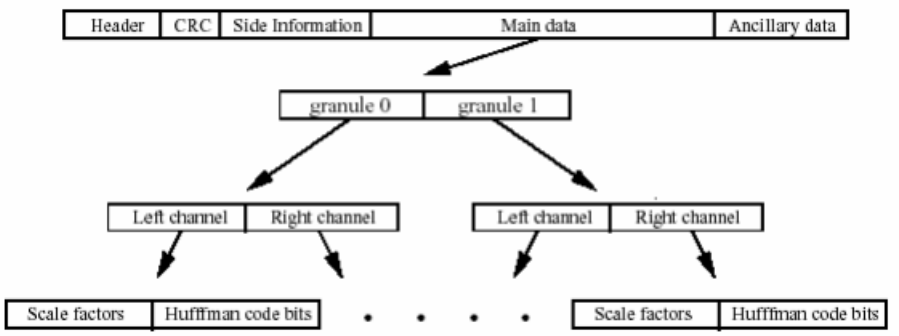


Figura 5.6: Struttura del campo Main Data in modalità stereo.

- *Fattori di scala*: i *fattori di scala* (o *scalefactor*) vengono utilizzati per ridurre il rumore da quantizzazione e rappresentano delle approssimazioni delle bande critiche dell'orecchio umano (che sono 24). Se scelti correttamente in fase di codifica, la maggior parte del rumore di quantizzazione sparirà, per effetto del mascheramento. Per ogni banda di fattori di scala, viene trasmesso (scelto) un solo fattore di scala. Il campo scfsi determina se (e quali) fattori di scala sono condivisi dai due granuli, in modo da trasmetterli una sola volta. I bit effettivi allocati per i fattori di scala dipendono dal valore del campo scalefac_compress.
  La suddivisione delle frequenze dello spettro in bande di fattori di scala è fissata a seconda del tipo di finestra e della sampling frequency. La suddivisione effettiva è memorizzata in tabelle all'interno dell'encoder e del decoder MP3.

- *Codifica Huffman*: questa parte dei dati principali contiene i valori codificati tramite codifica di Huffman. Le informazioni su come decodificare questi dati sono contenute nel campo Side Information. Ricordiamo che i valori delle regioni big values sono codificati a coppie, mentre quelli della regione count1 sono codificati a gruppi di 4.

5.1.5 Ancillary Data

La parte di dati aggiuntivi è opzionale e la sua lunghezza è variabile e non esplicitata. Inizia subito dopo la codifica Huffman del secondo granulo ed arriva fino alla word di sincronizzazione del frame successivo. Questi dati possono essere usati per scopi specifici all'interno di determinate applicazioni.

5.2 ID3

Anche se l'MP3 riesce ad ottenere una notevole compressione audio senza un'eccessiva perdita di qualità, esso manca della possibilità di aggiungere informazioni testuali. Infatti, specialmente per quando riguarda le canzoni, risulta molto comodo salvare all'interno del file `.mp3` informazioni testuali come autore o titolo della canzone.
Per soddisfare queste esigenze nacque il cosiddetto ID3, ovvero una serie di tag (di 128 byte) memorizzati alla fine del file `.mp3`. I tag disponibili erano i seguenti:

- titolo (30 byte);
- artista (30 byte);
- album (30 byte);
- anno (4 byte);
- commento (30 byte);
- genere (1 byte).

Inoltre un file `.mp3` che volesse fare uso dei tag ID3 doveva scrivere subito dopo i dati aggiuntivi la parola "TAG", per indicare dove iniziavano i tag ID3. Escludendo la parola "TAG", la somma dei tag ID3 misurava sempre 128 byte (lo spazio inutilizzato veniva riempito con zeri).

In una variante dell'ID3, detta ID3 v1.1 (Figura 5.7) al tag del commento sono stati sottratti due bit per aggiungere un tag *traccia* (riferendosi al numero di traccia della canzone nel CD di provenienza) di 1 byte ed un byte nullo tra il commento ed il numero di traccia.

'TAG' (3)	Title (30)	Artist (30)	Album (30)	Year (4)	Comment (28)	'0'	Track (1)	Genre (1)

Figura 5.7: Struttura dei tag ID3 v1.1.

Le prime versioni di ID3, tuttavia, avevano diversi aspetti negativi, come il fatto di avere pochi campi e che questi avessero una lunghezza massima di 30 byte. Inoltre, dato che le informazioni erano memorizzate alla fine del file `.mp3` non era possibile ottenerle durante uno streaming in tempo reale. Venne quindi rilasciata una versione più complessa, detta ID3 v2, che memorizzava i tag all'inizio del file `.mp3`, metteva a disposizione più campi e non imponeva un limite massimo di byte, in quanto supportava dimensioni dinamiche. Attualmente (settembre 2013) l'ultima versione rilasciata dell'ID3 è l'ID3 v2.4.

Capitolo 6

Codifica

Iniziamo finalmente l'analisi dell'algoritmo di compressione MP3. I passi principali di quest'algoritmo sono illustrati in Figura 6.1.

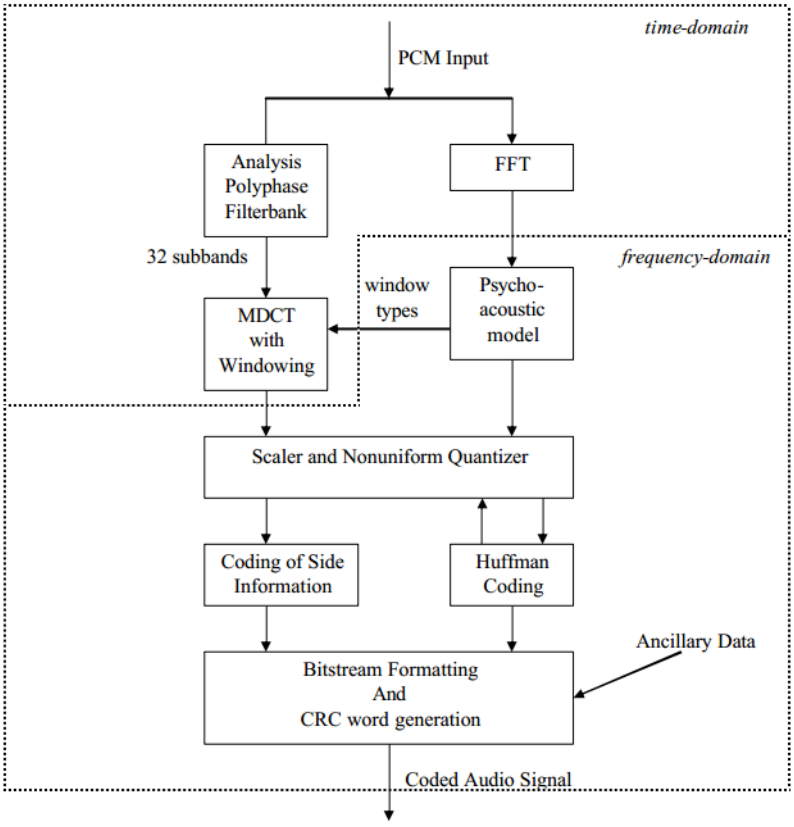


Figura 6.1: Schema della codifica MP3.

Di seguito verranno illustrate le varie fasi, senza tuttavia entrare troppo nel dettaglio o in ambito implementativo.

6.1 Banco filtri ibrido

Il primo passo della codifica MP3 consiste nel suddividere i sample PCM in sottobande e nel mapparli in sample a dominio di frequenza. Questo procedimento, detto *banco filtri ibrido*, è composto da un *banco filtri di analisi polifasico* e dall'algoritmo *MDCT*.

6.1.1 Banco filtri di analisi polifasico

La prima fase consiste in un banco di filtri di analisi polifasico, che prende in input 1152 sample PCM e restituisce come output 1152 sample divisi in 32 sottobande equispaziate, ovvero suddivide il segnale di input nelle sue componenti raggruppate per sottobanda. Gli estremi delle 32 sottobande sono definiti dalla sampling frequency utilizzata per ottenere i campioni PCM e dal limite di Nyquist corrispondente: se ad esempio la frequenza di campionamento è 44.1 *kHz*, le frequenze campionate vanno da 0 a 22.05 *kHz*, quindi ogni sottobanda sarà larga $22050/32 \approx 689$ *Hz*. Le sottobande saranno in questo caso 0-689 *Hz*, 689-1378 *Hz*, 1378-2067 *Hz*, ecc. . . .

Dal momento che, virtualmente, ogni sample PCM può contenere componenti di ogni sottobanda, il banco di filtri di analisi produce, di fatto, 32 sample (uno per ogni sottobanda) per ogni sample PCM, aumentando il numero di sample di un fattore 32. Quindi viene eseguito un *downsampling* (o *decimazione*) di fattore 32 (ovvero si prende un sample ogni 32), in modo da avere nuovamente 1152 sample. Ovviamente, il downsampling inserisce dell'aliasing (è quindi un procedimento lossy).
Il banco filtri di analisi polifasico è costruito mettendo in parallelo 32 filtri passabanda, ognuno destinato a far passare soltanto le frequenze appartenenti alla sottobanda corrispondente. Ogni sottobanda sarà composta da 36 (=1152/32) sample, suddivisi in 3 gruppi da 12, come mostrato in Figura 6.2.

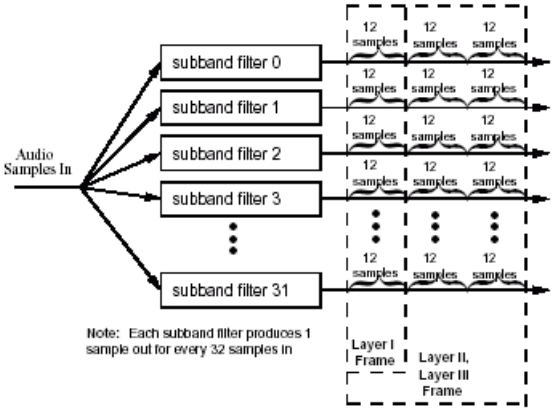


Figura 6.2: Banco filtri di analisi polifasico.

6.1.2 MDCT

Applicando una *Trasformata Discreta del Coseno Modificata* (*MDCT*, *Modified Discrete Cosine Transform*) ognuna delle 32 sottobande viene suddivida in 18 sottobande più fini, andando a formare un granulo di 576 linee di frequenza (anche se di fatto sono bande di frequenza, l'ampiezza di banda è talmente fine che vengono spesso chiamate linee di

frequenza). Inoltre, essendo un tipo particolare di Trasformata Discreta del Coseno, e quindi della Trasformata di Fourier, i sample in input a dominio di tempo vengono trasformati in coefficienti a dominio di frequenza.
L'MDCT prende in input i 1152 sample generati dal banco filtri di analisi polifasico (corrispondenti ad un frame, o due granuli) e produce in output 576 ($= 32 \cdot 18$) linee di frequenza (corrispondenti a un granulo). La MDCT si basa sulla sovrapposizione degli input (a due a due) per produrre gli output, secondo lo schema in Figura 6.3.

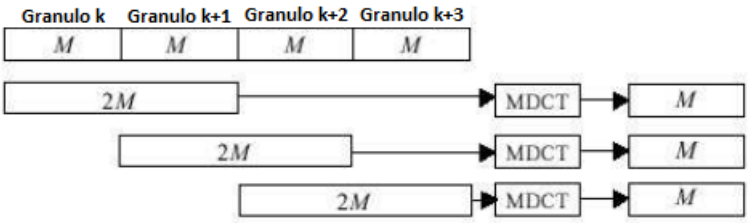


Figura 6.3: Trasformazione MDCT.

In generale, la MDCT prende in input $2N$ sample $\{y_n\}$, $n = 0, \ldots, 2N-1$, e restituisce in output N sample a dominio di frequenza $\{x_k\}$, $k = 0, \ldots, N-1$, secondo la seguente formula:

$$x_k = \sum_{n=0}^{2N-1} y_n \cdot \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2} + \frac{N}{2}\right)\left(k + \frac{1}{2}\right)\right], \qquad k = 0, \ldots, N-1. \tag{6.1}$$

Prima di applicare la MDCT è tuttavia necessario utilizzare una funzione di finestra per eliminare (assieme alla MDCT) gli artefatti temporali durante una transizione di frame. Lo standard MP3 definisce quattro tipi di finestre (Figura 6.4), a seconda della stazionarietà del modello psicoacustico.

Se il modello psicoacustico decide che (per ogni sottobanda) il segnale non cambia, o cambia poco, tra il frame corente e quello precedente, allora viene utilizzata una *finestra lunga*, che migliora la risoluzione spettrale (ovvero sulla frequenza) data dall'MDCT. Se al contrario vi sono notevoli differenze, allora vengono utilizzate *tre finestre corte* (sovrapposte) per migliorare la risoluzione temporale data dall'MDCT e quindi eliminare gli artefatti temporali.
Quando si passa da una finestra lunga a finestre corte, si utilizza una *finestra inizio*. Viceversa, se si passa da finestre corte ad una finestra lunga si utilizza una *finestra fine*. In concreto, l'applicazione della funzione finestra consiste nel moltiplicare ogni input per una funzione seno (a seconda della finestra utilizzata).

Come già accennato, la decimazione fatta dal banco di filtri di analisi polifasico introduce dell'aliasing, ma permette di inviare meno informazioni. L'aliasing in questione

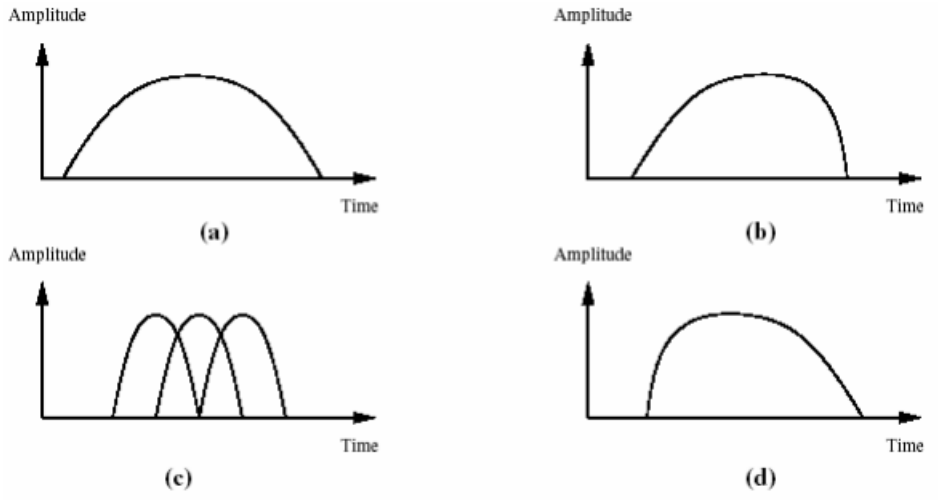


Figura 6.4: Tipologie di finestre: (a) finestra lunga; (b) finestra inizio; (c) 3 finestre corte; (d) finestra fine.

verrà ridotto in fase di decodifica tramite calcoli "a farfalla", come vedremo nel Capitolo 7.

6.2 Fast Fourier Transform

Dal momento che la risoluzione di frequenza data dal banco filtri è troppo bassa, simultaneamente al banco filtri di analisi polifasico viene effettuata una *Trasformata di Fourier veloce*, o *Fast Fourier Transform* (*FFT*), in modo da avere una miglior risoluzione e più informazioni sul cambiamento di frequenza nel tempo. La FFT (che non è altro che una versione ottimizzata della Trasformata Discreta di Fourier) mappa il segnale in entrata a dominio di tempo in un segnale a dominio di frequenza. In questo caso viene utilizzata una FFT a 1024 o 256 punti su 1152 sample PCM in entrata.

6.3 Modello psicoacustico

Questa fase rappresenta il modello psicoacustico. Esso prende in input i sample, a dominio di frequenza, generati dalla FFT e fornisce informazioni all'MDCT su quale finestra utilizzare. Inoltre da anche informazioni alla fase di Quantizzazione Non Uniforme su come quantizzare le diverse linee di frequenza.

Per decidere quale finestra inviare all'MDCT, vengono comparati gli spettri del frame corrente e del frame precedente. Se non vengono rilevati cambiamenti (*no attack*) allora si utilizza una finestra lunga. Se vengono registrati cambiamenti (*attack*), viene effettuato il passaggio da finestra lunga a finestre corte e quest'ultime verranno utilizzate finché il

segnale non si stabilizzerà nuovamente. In Figura 6.5 è rappresentato il diagramma di decisione della finestra da inviare all'MDCT.

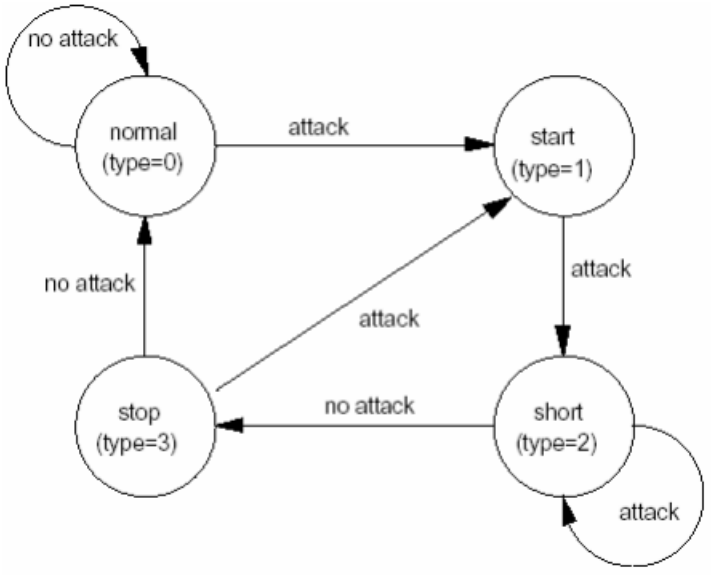


Figura 6.5: Modello psicoacustico: diagramma di decisione della finestra.

Il modello psicoacustico, inoltre, analizza lo spettro fornito dalla FFT per indiciduare le componenti dominanti e calcolare la soglia di mascheramento per ogni banda critica (le frequenze sotto questa soglia verranno mascherate). Le soglie di mascheramento di ogni banda critica verranno quindi utilizzate nella Quantizzazione Non Uniforme per diminuire il rumore di quantizzazione.

6.4 Quantizzazione Non Uniforme

In questa fase vengono scalati e quantizzati 576 sample a dominio di frequenza per volta. Lo schema generale presenta due cicli annidati: *rate control loop* (ciclo di controllo del tasso, esterno) e *distorsion control loop* (ciclo di controllo della distorsione, interno).

- *rate control loop*: in questo ciclo avviene la quantizzazione dei sample. I valori vengono quantizzati più volte, con step di quantizzazione sempre maggiori, fino a che i valori quantizzati non possono essere codificati con una delle tabelle di Huffman disponibili (step di quantizzazione più grandi implicano valori quantizzati più piccoli). Quindi viene calcolata la somma dei bit utilizzati per i valori quantizzati codificati con Huffman: se i bit necessari rientrano nei bit disponibili, allora si passa al loop interno, altrimenti si ripete il processo aumentando lo step di quantizzazione finché i bit disponibili non sono sufficienti. In questo step, inoltre, vengono calcolate anche

le regioni e sottoregioni (big values, count1, rzero, ecc...) ed i rispettivi limiti. Lo step di quantizzazione viene regolato dal campo global_gain nella Side Information di ogni granulo (vedi Capitolo 5).

- *distorsion control loop*: in questo ciclo, invece, viene controllato il rumore di quantizzazione prodotto nel ciclo precedente e vengono quindi aggiustati i fattori di scala finché il rumore non è mascherato per ogni banda di fattori di scala. Si applicano quindi i fattori di scala ad ogni linea di frequenza di ogni banda di fattori di scala, finché tutte le bande di fattori di scala non hanno il rumore inferiore alla soglia di mascheramento calcolata per quella banda (dal modello psicoacustico). Se una banda presenta un rumore di quantizzazione maggiore della soglia di mascheramento, allora il rumore sarà udibile dall'orecchio umano: in questi casi il fattore di scala corrispondente viene aumentato (maggiore è il fattore di scala, meno viene scalato il sample) e il processo ripetuto. Quando finalmente tutte le bande di fattori di scala saranno poco rumorose (ovvero con rumore sotto la soglia di mascheramento) si uscirà dal ciclo di controllo di distorsione, salvando fattori di scala e step di quantizzazione, per entrare nuovamente nel ciclo esterno, in modo da poter utilizzare step di quantizzazione più piccoli (se possibile) con i nuovi valori scalati.

6.5 Codifica Huffman

Una volta terminata la quantizzazione e la determinazione dei fattori di scala, i valori quantizzati vengono codificati con la tabella di Huffman corrispondente (a seconda delle tabelle scelte per le varie regioni e sottoregioni). Si osservi che, prima della codifica di Huffman, vengono riordinate le linee di frequenza relative alle finestre corte prima per sottobanda, poi per frequenza ed infine per finestra, in quanto sample con frequenza simile hanno più probabilità di avere valori simili (l'output dell'MDCT ordinava questi sample per sottobanda, per finestra e quindi per frequenza).

6.6 Strutturazione della Side Information

A questo punto tutti i dati generati nelle fasi precedenti (step di quantizzazione, tabelle di huffman, finestre, ecc...) vengono strutturati secondo il formato visto per la Side Information, essendo necessari per la decodifica del file `.mp3`.

6.7 Formattazione del flusso di bit

A questo punto l'encoder MP3 è in possesso di tutti i dati necessari: l'header, la CRC, la Side Information e i dati principali vengono messi insieme e strutturati secondo il formato visto nel Capitolo 5. Il risultato è un frame MP3, corrispondente a 1152 sample PCM, ovvero a 26 *ms* di audio[1].

[1] A 44.1 *kHz* di frequenza di campionamento.

Capitolo 7

Decodifica

Lo schema generale della decodifica di un file `.mp3` è rappresentato in Figura 7.1. In linea di massima, vengono ripercorse le fasi descritte nel Capitolo 6, ma al contrario

7.1 Sincronizzazione

Il flusso di bit letti dal file `.mp3` viene scorso finhé non viene trovata la word di sincronizzazione (12 bit a 1). Se non viene rilevata la word di sincronizzazione di un frame, i dati non possono essere codificati ed il file è corrotto.

7.2 Decodifica Huffman e dei parametri relativi

Innanzitutto è necessario decodificare i sample quantizzati tramite decodifica Huffman. Abbiamo visto che la codifica di Huffman è una codifica a lunghezza variabile ed inoltre dipende dalla tabella di Huffman impegata. Ad esempio non è possibile prendere una serie di bit a caso e sperare di decodificarli: è necessario leggere le sequenze codificate dall'inizio e sapere come sono stati codificati i dati. Vengono quindi decodificati (dalla Side Information) tutta una serie di parametri relativi alla codifica Huffman, come la suddivisione in regioni e sottoregioni, le tabelle di Huffman utilizzate, ecc ....
Una volta in possesso di tutti i parametri necessari, i 576 sample verranno decodificati, utilizzando le tabelle di Huffman. Si noti che nel caso in cui meno di 576 sample siano trovati, dovranno essere inseriti i sample mancanti come sequenze di zeri.

7.3 Decodifica dei fattori di scala

In questa fase vengono decodificati i fattori di scala, che si trovano all'inizio dei dati principali di ogni granulo. I dati relativi a dove trovare i fattori di scala sono codificati nella Side Information di ogni granulo.

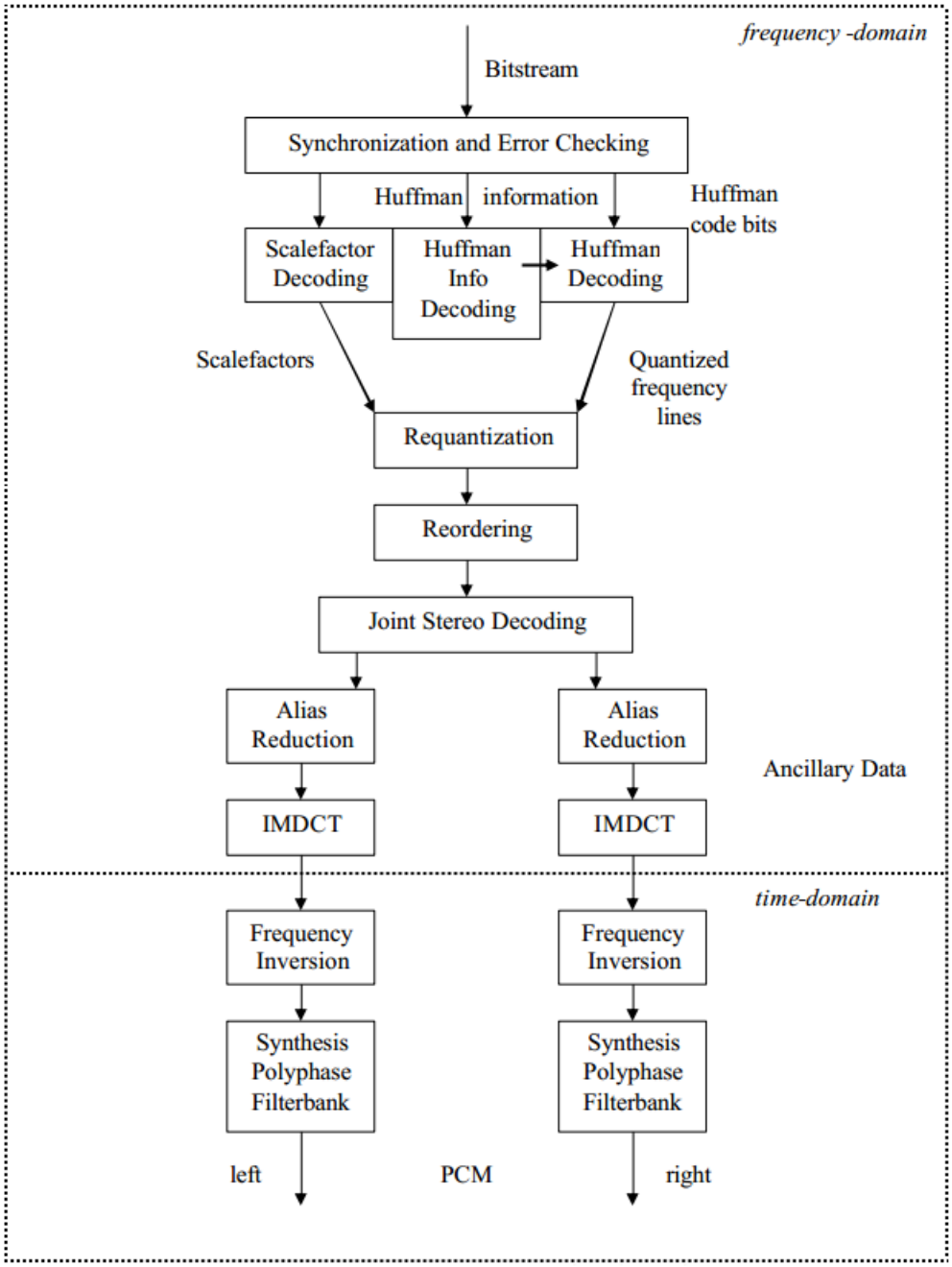


Figura 7.1: Schema della decodifica MP3.

7.4 Riquantizzazione

Ricavando i valori di campi come global_gain o scalefac_scale si possono riquantizzare i sample quantizzati e scalati ottenuti dalla decodifica di Huffman, ottenendo le linee di frequenza generate dall'MDCT. I dati sui fattori di scala, necessari per la riquantizzazione, vengono forniti dallo step precedente (decodifica dei fattori di scala). Si osservi che in questo step vengono utilizzate funzioni di riquantizzazione differenti a seconda della finestra usata.

7.5 Riordinamento

Le linee di frequenza generate dal processo di riquantizzazione non sono sempre nell'ordine in cui dovrebbero essere dopo l'MDCT. Infatti se nell'MDCT era stata usata una finestra lunga, allora le linee di frequenza erano ordinate prima per sottobanda e poi per frequenza, mentre se erano state usate finestre corte erano ordinate per sottobanda, per finestra e quindi per frequenza. Tuttavia per aumentare l'efficienza della codifica Huffman, le linee di frequenza relative a finestre corte venivano riordinate, prima per sottobanda, poi per frequenza ed infine per finestra, in quanto sample di frequenza vicina hanno molto probabilmente un valore simile.
Vengono allora cercate le linee di frequenza relative alle finestre corte in tutte le sottobande e, quando trovate, vengono riordinate, in modo da avere esattamente l'output dell'MDCT in fase di codifica.

7.6 Decodifica stereo

Questo passaggio serve per separare l'insieme di sample generato in due segnali distinti: destro e sinistro. Per ricavare la modalità di canale utilizzata si leggo i valori dei campi Mode e Mode Extension dall'header del frame.

7.7 Riduzione dell'aliasing

Prima di invertire l'MDCT è necessario ridurre l'aliasing prodotto dal banco filtri di analisi. Questo viene fatto applicando una serie di 8 *calcoli a farfalla* (si veda la Figura 7.2) per ogni sottobanda, che rimuovono gli artefatti dal segnale.

7.8 IMDCT

A questo punto è possibile applicare una *IMDCT*, ovvero una *Trasformata Discreta del Coseno Modificata Inversa*. L'IMDCT prende in input 576 linee di frequenza e restituisce 576 sample a dominio di tempo, suddivisi in 32 sottobande (18 sample per sottobanda).

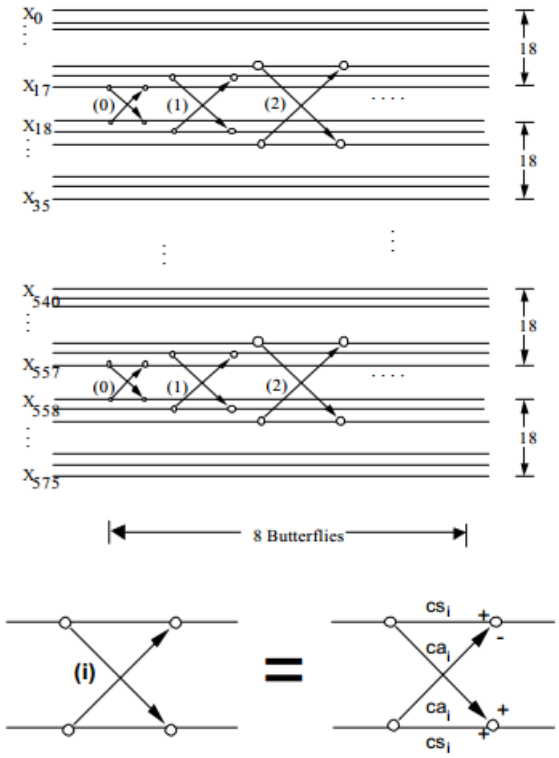


Figura 7.2: Riduzione dell'aliasing tramite calcoli a farfalla.

Analogamente alla (6.1), la formula dell'IMDCT è la seguente:

$$y_n = \frac{1}{N} \sum_{k=0}^{N-1} x_k \cdot \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2} + \frac{N}{2}\right)\left(k + \frac{1}{2}\right)\right], \qquad n = 0, \ldots, 2N-1. \tag{7.1}$$

7.9 Inversione di frequenza

Dal momento che il banco filtri di analisi polifasico ha invertito le frequenze (similmente a quanto accade con la Trasformata di Fourier e sue derivate), per compensare si moltiplica ogni sample di indice dispari di ogni sottobanda dispari per -1.

7.10 Banco filtri di sintesi polifasico

Infine, vengono fatti passare i sample di ogni sottobanda da un banco filtri di sintesi polifasico, che rappresenta l'inverso del banco filtri di analisi polifasico, visto in fase di codifica. Il risultato sono 576 sample PCM, per canale, che corrispondono al segnale audio decodificato.

Conclusioni

Concludiamo questo elaborato dicendo che la tecnologia MP3 ha sicuramente dato una svolta allo streaming e alla condivisione audio (e video). La qualità che l'MP3 riesce ad ottenere è notevole, ma ancor più notevoli sono le dimensioni del file `.mp3` compresso. Non ci meravigliamo se pensiamo che questo formato è ormai il più diffuso a livello globale per la memorizzazione di tracce audio.
A livello personale, sono rimasto colpito dalla complessità intrinseca dell'MP3, ma soprattutto dalle idee (a mio parere geniali) che hanno portato alla nascita di questo algoritmo di compressione. Chi ha sviluppato l'MP3 ha senz'altro avuto un'idea unica, senza la quale oggi non saremmo in grado di usufruire di tutti i servizi di mediatici di cui facciamo uso.

Bibliografia

[1] A/I. Dentro l' MP3, 2001. URL http://www.autistici.org/hacklab_fi/2001/netart/inside-mp3.html.

[2] ComeFunziona.net. MP3. URL http://www.comefunziona.net/arg/mp3/.

[3] Hydrogenaudio. MP3, Gennaio 2013. URL http://wiki.hydrogenaudio.org/index.php?title=MP3.

[4] Joebert S. Jacaba. Audio compression using modified discrete cosine transform: The mp3 coding standard, Ottobre 2011. URL http://www.mp3-tech.org/programmer/docs/jacaba_main.pdf.

[5] Mark Kilgore and Jamie Wu. MPEG, the MP3 Standard, and Audio Compression, Settembre 2003. URL http://www-ee.stanford.edu/~osgood/Sophomore%20College//Audio%20Compression%20and%20the%20MP3%20Standard.pdf.

[6] Hung-Chih Lai. REAL-TIME IMPLEMENTATION OF MPEG-1 LAYER 3 AUDIO DECODER ON A DSP CHIP, Giugno 2001. URL http://www.mp3-tech.org/programmer/docs/thesis_lai.pdf.

[7] Gregorio Landi. Elementi di Teoria dell'Informazione. Marzo 2013.

[8] Wen-Chieh Lee. Design of the Audio Coding Standards for MPEG and AC-3. URL http://www.mp3-tech.org/programmer/docs/paper-1-english.pdf.

[9] Mariella Baldussi. Notes about Audio MPEG 2. URL http://www.lim.dico.unimi.it/IEEE/MPEG2/MPEG2IDX.HTM.

[10] Michele Scarpiniti. Laboratorio per l'Elaborazione MultiMediale Lezione 4 - Banco Filtri, Aprile 2011. URL http://ispac.ing.uniroma1.it/scarpiniti/files/LabEMM/Less4.pdf.

[11] Tomas Pettersson. Basic sound theory and synthesis, Marzo 2010. URL http://drpetter.se/article_sound.html.

[12] MTU Physics. Physics of Music - Notes. URL http://www.phy.mtu.edu/~suits/Physicsofmusic.html.

[13] Pootle_1. Difference between Mp3gain and Replaygain, Luglio 2004. URL http://www.hydrogenaudio.org/forums/index.php?showtopic=24527.

[14] Rassol Raissi. The Theory Behind Mp3. Dicembre 2002. URL http://www.mp3-tech.org/programmer/docs/mp3_theory.pdf.

[15] Carlo Andrea Rozzi. Mascheramento, Ottobre 2010. URL http://fisicaondemusica.unimore.it/Mascheramento.html.

[16] Paul Sellars. Perceptual Coding: How Mp3 Compression Works, Maggio 2000. URL http://www.soundonsound.com/sos/may00/articles/mp3.htm.

[17] Szymon Stefanek. Harmony and Waveforms, 2007. URL http://www.pragmaware.net/articles/harmony/index.php.

[18] Giancarlo Vercellesi. MPEG / AUDIO TUTORIAL, Marzo 2005. URL http://www.lim.dico.unimi.it/teaching/materials_pdf/MPEG_infoMusicale.pdf.

[19] Wikipedia. MP3, Luglio 2013. URL http://en.wikipedia.org/wiki/MP3.

[20] Wikipedia. Sound, Agosto 2013. URL http://en.wikipedia.org/wiki/Sound.

[21] Wikipedia. Lossy compression, Agosto 2013. URL http://en.wikipedia.org/wiki/Lossy_data_compression.

[22] Wikipedia. Absolute threshold of hearing, Agosto 2013. URL http://en.wikipedia.org/wiki/Absolute_threshold_of_hearing.

[23] Wikipedia. Pulse-code modulation, Luglio 2013. URL http://en.wikipedia.org/wiki/Pulse-code_modulation.