


Linguaggi ed Applicazioni multimediali

02.02 – Media Compression

Media Compression

Maurizio Maffi

ISTI Information Science and Technology Institute

COMPRESSIONE DI IMMAGINI

L'obiettivo della compressione è quello di ridurre il consumo di spazio nella memorizzazione di un file grafico senza impiegare troppo tempo e minimizzando la perdita d'informazioni.

Compressione: $f(x,y) \xrightarrow{C} \hat{f}(x,y)$

Si comprime per:

Memorizzare ➔ minor spazio possibile

Trasmettere ➔ minor tempo possibile

DECOMPRESSIONE DI IMMAGINI

In fase di decompressione possiamo ottenere due risultati:

$$\hat{f}(x,y) \xrightarrow{D} f(x,y)$$

L'immagine decompressa è uguale a quella di partenza

$$\hat{f}(x,y) \xrightarrow{D} \tilde{f}(x,y)$$

L'immagine decompressa è SIMILE a quella di partenza, qualche dato è andato perso

DECOMPRESSIONE DI IMMAGINI

Nel primo caso è stato utilizzata una tecnica di **compressione LOSSLESS** in cui non si ha perdita d'informazioni; l'output prodotto in fase di decompressione è l'input della compressione.

Nel secondo caso si è adottata la tecnica di **compressione LOSSY** in cui si ha una perdita d'informazione: si vuole ottenere un compromesso tra accuratezza nella ricostruzione e aumento della compressione.

CLASSIFICAZIONE

- **LOSSLESS**
  - **HUFFMAN**
  - **ARITMETICO**
  - **LZW**
  - **RLL**
- **LOSSY**
  - **PREDICTIVE**
  - **FREQUENCY ORIENTED**
  - **IMPORTANCE ORIENTED**
  - **IBRIDE (Jpeg, Mpeg)**

RIDONDANZA DELL'INFORMAZIONE

Per valutare la compressione di un'immagine si utilizza la <u>COMPRESSION RATIO</u>

$$C_R = \frac{n_1}{n_2}$$

→ Immagine originale

→ Immagine compressa

$n_1 = n_2 \quad c_R = 1$ → l'immagine non ha dati ridondanti, nessun miglioramento con la compressione

$n_2 << n_1 \quad c_R \longrightarrow \infty$ → Compressione significativa. L'algoritmo utilizzato risulta adeguato

$n_1 << n_2 \quad c_R \longrightarrow 0$ → Aumento dell'informazione dovuto alla tecnica di compressione utilizzata. Occorre cambiare algoritmo

RIDONDANZA DELL'INFORMAZIONE

Alternativamente si può definire il **bitrate** che è il numero di bit trasmesso per ogni pixel. L'unità di misura è il bpp (bit per pixel).

$$bitrate = \frac{b}{C_R}$$ → N° pixel nell'immagine originale

CODIFICA RUN-LENGHT

È uno dei modi più naturali e semplice per comprimere

Esempio 1:

File ASCII AAAABBBAABBBBBCCCCCCCCDABC
File compresso **4**A**3**BAA**5**B**8**CDABC

Tecnica: si fa precedere al carattere il numero di occorrenze consecutive del carattere

Domanda: se la stringa contiene numeri?
Risposta: si adotta un alfabeto alternativo per rappresentare i numeri con simboli
Esempio 4 → å 2→ô

CODIFICA RUN-LENGHT

Esempio 2:

File ASCII	File di testo
File compresso	File di testo

La codifica Run-Lenght non è consigliata per comprime testi

Esempio 3

File BINARIO	0000001111110000
File Compresso	654

Non c'è bisogno di memorizzare il tipo di carattere, basta sapere con che caratteri si parte. Fissiamo, per convenienza, la partenza con lo zero e otteniamo 654

CODIFICA RUN-LENGHT

VANTAGGI:

- vantaggioso per lunghe sequenze uguali
  Esempio: 11111110000000011111111000
  AAAAAAXXXXXVVVEEEEEEEE

- Facile da implementare

SVANTAGGI

- Risulta svantaggioso per sequenze corte
  Esempio: 1011010101010
  acxasdaasdc..

CODIFICA DI HUFFMAN

Codifica utilizzata anche negli mp3 (e zip)

L'idea alla base dell'algoritmo è quella di assegnare codici più corti alle sequenze più probabili e un numero maggiore di bit per i codici di sequenze che compaiono più raramente

Si necessita di una **tabella di conversione** per memorizzare la corrispondenza (codice,simbolo).

CODIFICA DI HUFFMAN

Esempio. ABBBCCCDDEEEEEEEEEF…

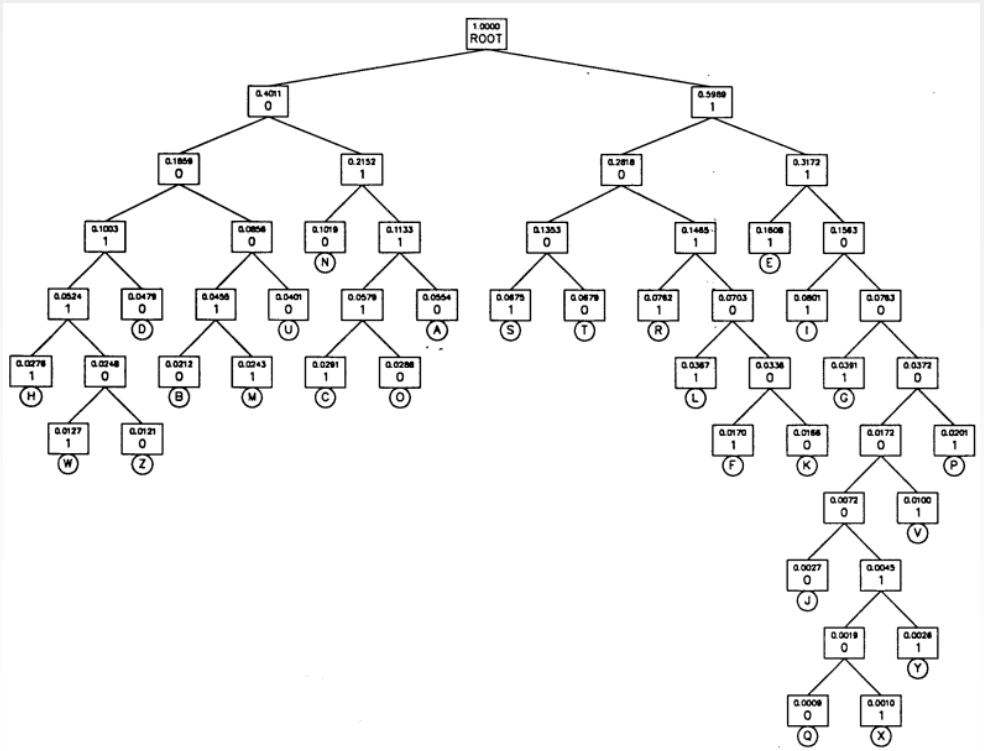
- valuto il numero di simboli differenti
  6 caratteri differenti → tabella a 6 entries

  - valuto la frequenza di ogni singolo carattere

A	B	C	D	E	F
1	3	3	2	9	1

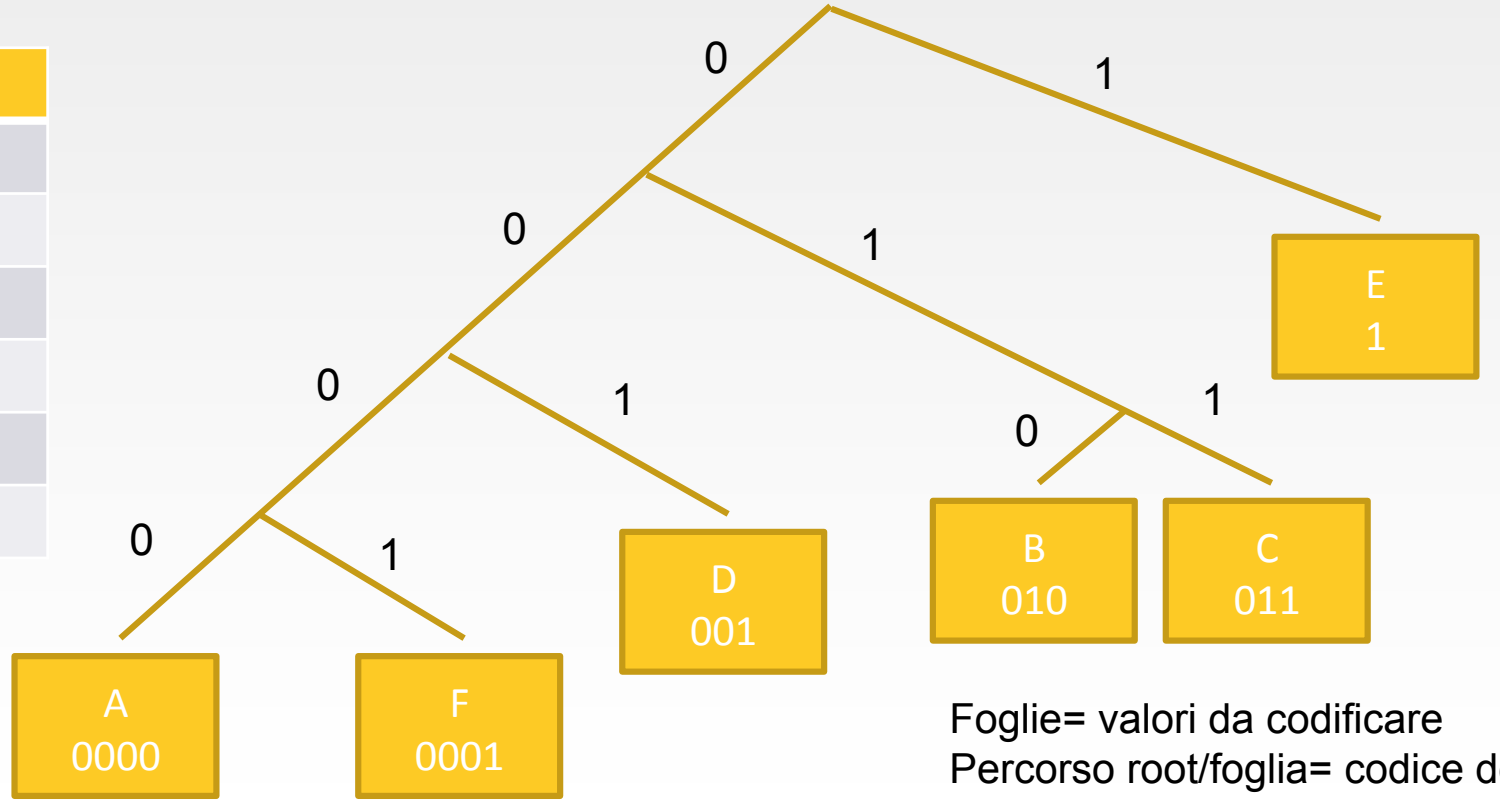
CODIFICA DI HUFFMAN

Anziché creare una tabella di corrispondenza(**costosa)** si crea un **albero binario** per codificare ogni singolo valore, partendo dal più frequente



CODIFICA DI HUFFMAN

A	1
B	3
C	3
D	2
E	9
F	1



Foglie= valori da codificare
Percorso root/foglia= codice del valore

CODIFICA DI HUFFMAN

PASSI DELL'ALGORITMO

1.Raccolta informazioni (conteggio caratteri differenti)
2.Costruzione dell'albero
3.Scrittura del file compresso

Il file compresso inerente all'esempio appena fatto è:

0000	010	010	010	010	011 ……..
A	B	B	B	B	C

VANTAGGI e SVANTAGGI
•Huffman risulta più adatto per valori singoli, non ripetuti
•Per file con valori ripetuti è preferibile l'utilizzo di RLL
•A volte può accadere che il file compresso risulti maggiore dell'originale

LZW (Lempel, Ziv, Welch)

Codifica utilizzata nei formati GIF e TIFF

L'idea alla base dell'LZW è quella di utilizzare una **tabella** iniziale con una entry per ogni simbolo (esempio 256 entries per dati a 8bit)

La tabella **cresce al comprimersi del file:** è adattiva. Vengono aggiunte nuove entries alla tabella per ogni pattern di valori unici trovati. Viene fissata la massima dimensione della tabella in modo da stabilire la lunghezza del codice.

ESEMPIO : ABABAAACAAAAD

TABELLA INIZIALE DEI CODICI

CODICE	VALORE
A	00
B	01
C	10
D	11

LZW (Lempel, Ziv, Welch)

L'algoritmo LZW procede cercando pattern più lunghi è inserendoli nella tabella

Il compressore trova il primo valore A già presente in tabella, allora cerca il pattern AB e non lo trova in tabella. Lo pone nella tabella assegnandoli il codice e aggiustando i codici esistenti. La tabella diviene:

CODICE	VALORE
A	000
B	001
C	010
D	011
AB	100

LZW (Lempel, Ziv, Welch)

Poi continua cercando il valore B lo trova e va avanti, poi cerca BC non lo trova, allora lo aggiunge ….. Ecc ecc….. Questa operazione si fa fino ad avere la tabella piena o sino a memorizzare il numero massimo di coppie di simboli

I dati perturbati da rumore sono difficili da comprimere con LZW

LZW esempio

Prendiamo in considerazione la seguente stringa ACGTACGTACG

Iterazione	Input	Buffer corrente	Simbolo in output	Voce aggiunta nel dizionario
1	ACGTACGTACG	(vuoto)	(nessuno)	(nessuna)
2	ACGTACGTACG	A		
3	ACGTACGTACG	AC		
4	ACGTACGTACG	C	A	AC (α_1)
5	ACGTACGTACG	CG		
6	ACGTACGTACG	G	C	CG (α_2)
7	ACGTACGTACG	GT		
8	ACGTACGTACG	T	G	GT (α_3)
9	ACGTACGTACG	TA		
10	ACGTACGTACG	A	T	TA (α_4)

Dizionario	
Sequenza	**Codice**
A	A
C	C
G	G
T	T
AC	α_1
CG	α_2
GT	α_3
TA	α_4
ACG	α_5
GTA	α_6

LZW (Lempel, Ziv, Welch)

11	ACGTACGTACG	AC		
12	ACGTACGTACG	ACG		
13	ACGTACGTACG	G	α_1	ACG ((α_5))
14	ACGTACGTACG	GT		
15	ACGTACGTACG	GTA		
16	ACGTACGTACG	A	α_3	GTA ((α_6))
17	ACGTACGTACG	AC		
18	ACGTACGTACG	ACG		
19	ACGTACGTACG		α_5	

Dizionario	
Sequenza	**Codice**
A	A
C	C
G	G
T	T
AC	α_1
CG	α_2
GT	α_3
TA	α_4
ACG	α_5
GTA	α_6

La stringa codificata risulta

$ACGT\alpha_1\alpha_3\alpha_5$

LOSSY COMPRESSION

Caratterizzata da una perdita d'informazione in fase di compressione e decompressione

Idea di base : compromesso tra accuratezza nella ricostruzione e aumento di compressione

In una tecnica LOSSY, occorre quantificare la perdita d'informazione. Infatti nell'eliminare informazioni ridondanti possono andare perse informazioni visive importanti.

Per quantificare la natura e l'entità dell'informazione persa ci si avvale di due criteri:

1.FEDELTÀ SOGGETTIVA

2.FEDELTÀ OGGETTIVA

LOSSY COMPRESSION – fedeltà soggettiva

Si basa sul concetto che la qualità dell'immagine debba essere valutata da un essere umano. Quindi si definisce una scala di valutazione del tipo

- INUTILIZZABILE
- SUFFICIENTE
- BUONA
- OTTIMA
- …….

E si fa una media delle valutazioni degli osservatori presi a campione, che osservano

$f(x,y)$ $\tilde{f}(x,y)$ e danno giudizio

LOSSY COMPRESSION – fedeltà oggettiva

Si basa sulla valutazione di un errore che esprime il livello di informazione persa in funzione di $f(x,y)$ $\hat{f}(x,y)$ $\tilde{f}(x,y)$

La funzione errore è detta **ROOT MEAN SQUARE ERROR (RMS)** che indica la discrepanza quadratica media fra i valori dei dati osservati ed i valori dei dati stimati Sia l'immagine errore così definita:

$$e(x,y) = f(x,y) - \tilde{f}(x,y) \;\; \forall x,y \in M \times N$$

Si definisce:

ROOT MEAN SQUARE ERROR

$$RMSE = \left[\frac{1}{MN} \sum_{X=0}^{M-1} \sum_{Y=0}^{N-1} e(x,y)^2 \right]^{1/2}$$

LOSSY COMPRESSION – fedeltà oggettiva

Si può avere anche il

MEAN SQUARE ERROR

$$MSE = \frac{1}{MN} \sum_{X=0}^{M-1} \sum_{Y=0}^{N-1} e(x,y)^2$$

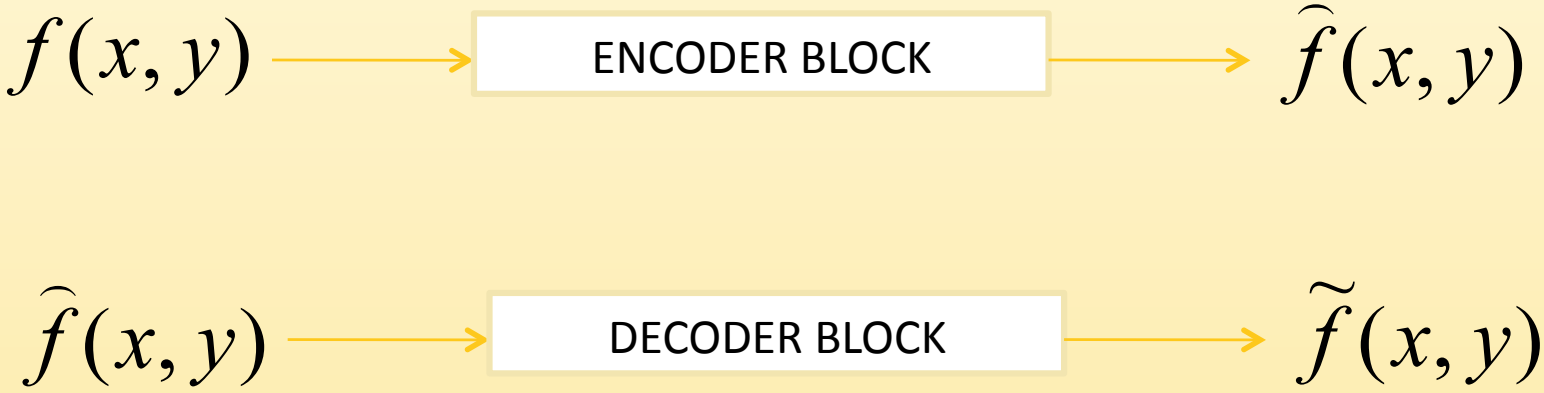
Inversamente all'errore si definisce

SIGNAL TO NOISE RATIO

$$SNR = \frac{\sum\sum \tilde{f}(x,y)^2}{\sum\sum e(x,y)^2}$$

Mette in relazione la potenza del segnale utile rispetto a quella del rumore in un qualsiasi sistema di acquisizione, elaborazione o trasmissione dell'informazione

SISTEMA DI COMPRESSIONE E DECOMPRESSIONE



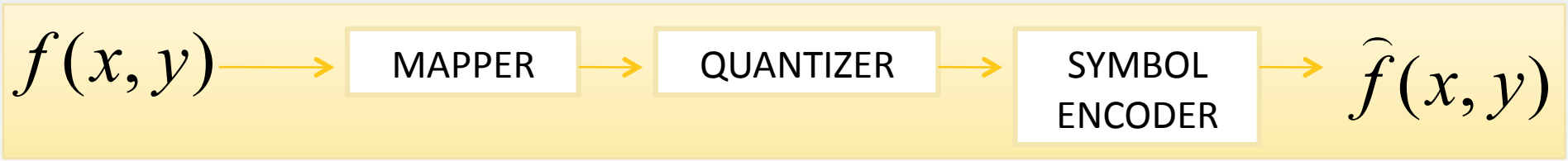
SISTEMA DI COMPRESSIONE E DECOMPRESSIONE

$\hat{f}(x,y)$ IMMAGINE COMPRESSA

$\tilde{f}(x,y)$ IMMAGINE DECOMPRESSA

Lo schema è lo stesso sia per le tecniche lossy che lossless

ENCODING



MAPPER: trasforma i dati dell'immagine in un formato che riduce la ridondanza nel codice (interpixel redundancy)

- **OPERAZIONE REVERSIBILE**

ENCODING

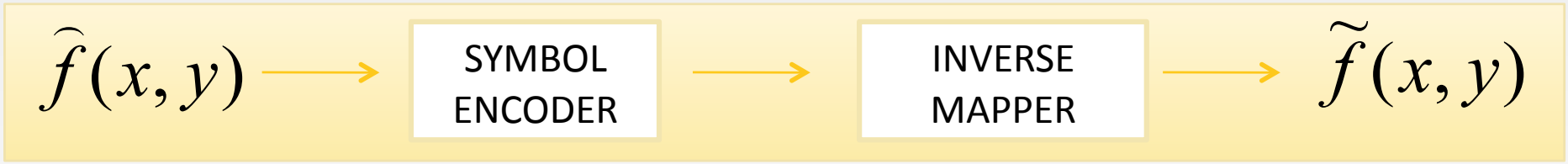
QUANTIZER: Riduce l'accuratezza dell'output del mapper seguendo predeterminati criteri di fedeltà dell'immagine

- **PROCESSO IRREVERSIBILE (non presente nelle tecniche error free)**

SYMBOL ENCODER: Crea un codice a lunghezza fissa o variabile per rappresentare l'output del quantizer

- **OPERAZIONE REVERSIBILE**

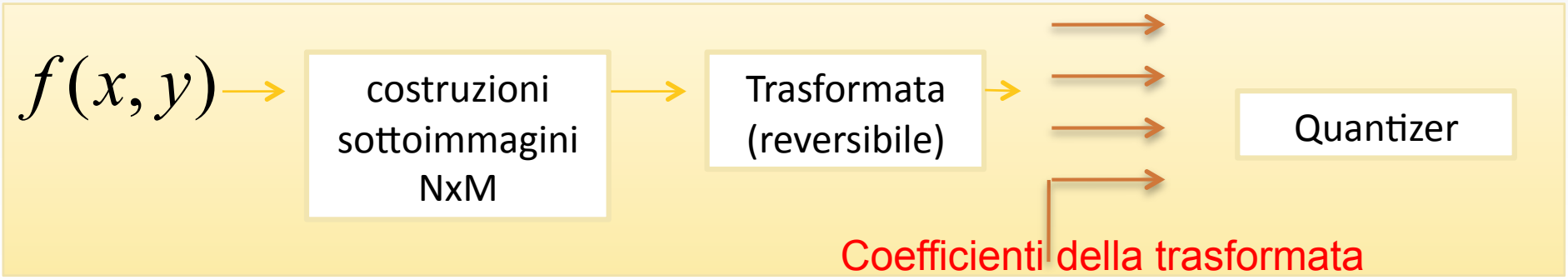
DECODING



MAPPER

Il mapper lavora in due fasi:

1.Costruisce una sottoimmagine, dividendo l'immagine originale in più blocchi
2.Si passano le immagini dal dominio discreto al continuo tramite l'utilizzo di trasformate.



l'**inverse mapper** effettuerà la trasformata inversa e il merge delle sottoimmagini

TRASFORMATE

Le trasformate che vengono utilizzate nel mapper sono:

1.DCT	Discrete Cosin Trasform
2.DFT	Discrete Fourier Trasform
3.WHT	Walsh-Hadamard Trasform
4.KLT	Karhunen-Loéve Trasform
5.DWT	Discrete Wavelet Trasform

COMPRESSIONE VIDEO MPEG

MPEG = Moving Picture Experts Group

Evoluzione:

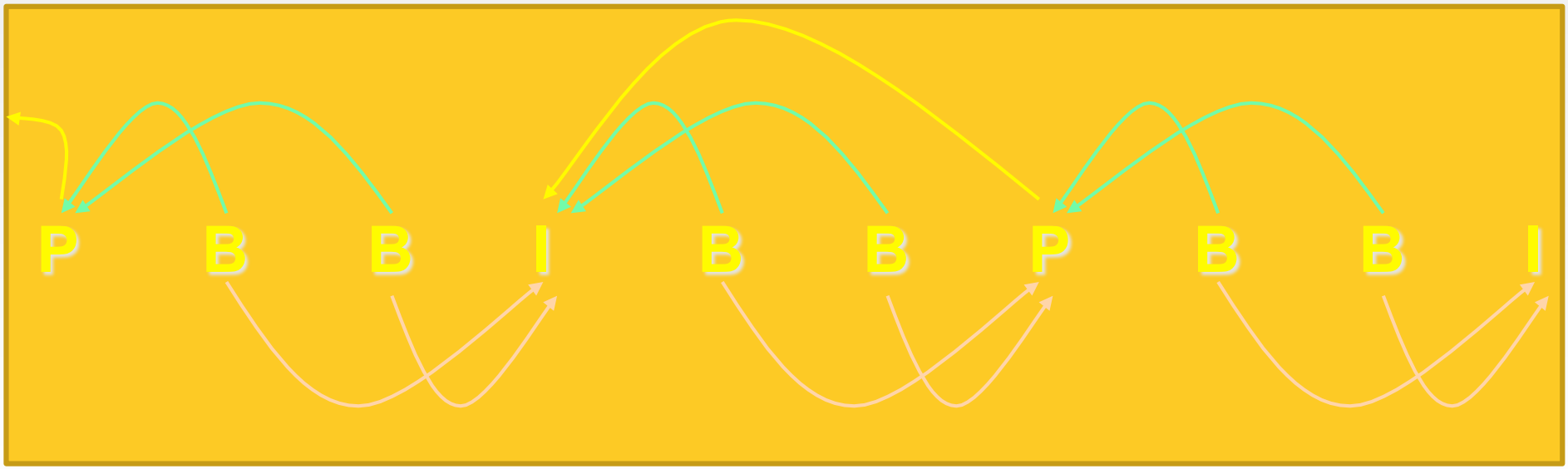
MPEG-1	permettere video da CD-ROM (circa 1.2 Mbps)
MPEG-2	video di grandi dimensioni per HDTV
MPEG-4	trasmissione su mezzi a bassa banda (< 64Kbps)
MPEG-7	descrizione ad alto livello dei contenuti
MPEG-21	futuro

STRUTTURA MPEG 1 - 2

Tre tipi di frame:

- **I-frame** è il fotogramma di riferimento principale (keyframe, poco compresso).
- **P-frame** è compresso con algoritmi di *motion compression ed estimation.* Questi confrontano il fotogramma di elaborazione con l'**i-frame** o con il precedente **p-frame,** memorizzando esclusivamente le zone del fotogramma che siano diverse
- **B-frame** è compresso come il **p-frame** ma facendo il confronto tra gli **i-frame** e i **p-frame** precedente e successivo (bidirezionale, maggior compressione, non usate come riferimento)

STRUTTURA MPEG 1 - 2



BIDIRECTIONAL FRAME: B FRAME

I B FRAMES nascono dall'esigenza di codificare i macroblocchi come differenza o con i blocchi presenti nel frame I o P precedente o con i blocchi presenti nel frame I o P seguente.

I macroblocchi codificati come differenza dal frame (I o P) precedente prendono nome di **FORWARD PREDICTED BLOCK (FP)**

I macroblocchi codificati come differenza dal frame (I o P) seguente prendono il nome di **BACKWARD PREDICTED BLOCK (BP)**

BIDIRECTIONAL FRAME: B FRAME

f= Frame ma = macroblocco

f 1	f 2	f 3	f 4	f 5	f 6	f 7	f 8	f 9	f 10	f 11	f 12	f 13
I	B	B	P	B	B	P	B	B	P	B	B	I

I ma del f1 sono tutti codificati
I ma del f2 cercheranno i blocchi simili nel f1 (I) e nel f4 (P)
I ma del f3 cercheranno i blocchi simili nel f1 (I) e nel f4 (P)
I ma del f4 cercheranno i blocchi simili nel f1 (I)
I ma del f5 cercheranno i blocchi simili nel f4 (P) e nel f7 (P)
I ma del f6 cercheranno i blocchi simili nel f4 (P) e nel f7 (P)
I ma del f7 cercheranno i blocchi simili nel f4 (P)
I ma del f8 cercheranno i blocchi simili nel f7 (P) e nel f10 (P)
I ma del f9 cercheranno i blocchi simili nel f7 (P) e nel f10 (P)
I ma del f10 cercheranno i blocchi simili nel f7 (P) ….

BIDIRECTIONAL FRAME: B FRAME

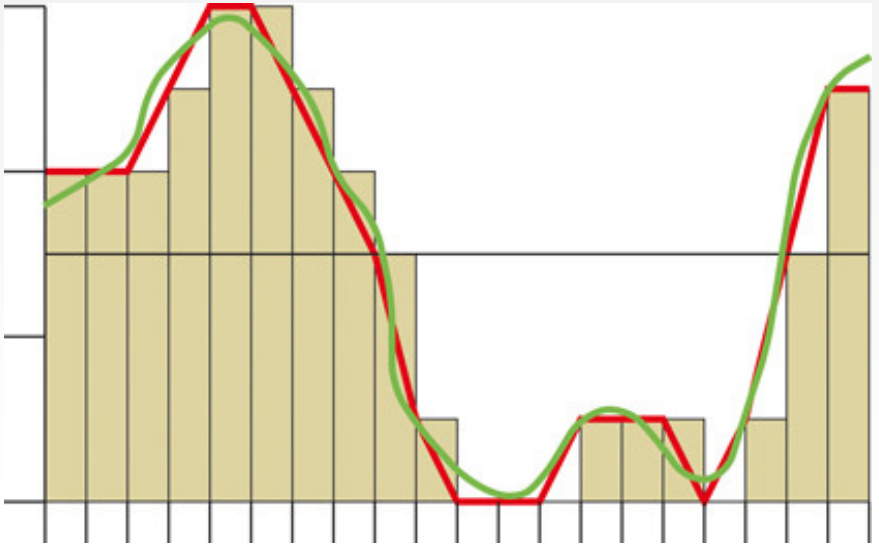
L’encoder non può comprimere il frame B2 se non conosce già il frame 4 da cui potenzialmente può calcolarsi le differenze , per questo occorre prima comprimere il frame 4 (P) e poi il 2 (B).

Stesso discorso per il player che non può visualizzare il frame 2 se non ha già decompresso il frame 4 (ovviamente il frame 4 verrà visualizzato al momento giusto)

COMPRESSIONE AUDIO MP3

Il formato MP3 è diminutivo di MPEG 1 Layer 3 che è l'algoritmo che viene utilizzato per la compressione di dati.

Il layer 3 (tasso di compressione) di Mpeg 1 consente di ottenere una compressione con un bitrate di 64 Kbit/s per canale (essendo, il segnale da comprimere un suono stereo, saranno necessari 2 canali avendo così un bitrate di 128 Kbit/s).



COMPRESSIONE AUDIO MP3

Il bitrate di un brano audio è il numero di bit necessari per codificare un secondo di canzone. Tenendo conto che il bitrate di un file WAVE (tipo di file utilizzato per i CD audio) è di 1411 Kbit/s è possibile fare un confronto con l'MP3.

È possibile notare che con un bitrate notevolmente ridotto è possibile ottenere lo stesso risultato. In realtà però l'algoritmo Mpeg 1 Layer 3, come abbiamo già detto, comporta la perdita di alcuni dati.

Questo taglio di informazioni dal file di partenza è possibile grazie ad alcuni criteri psico-acustici umani: si basa cioè sui suoni che effettivamente il nostro cervello è in grado di percepire.

COMPRESSIONE AUDIO MP3

L'orecchio umano, infatti, non è in grado di percepire suoni al di sopra dei 20.000 Hz. La compressione MP3 tiene conto di questo fatto, tagliando tutte le frequenze al di sopra dei 20.000 Hz, risparmiando spazio prezioso.

Questo tipo di compressione quindi analizza lo spettro del segnale acustico e, applicando il modello psico-acustico, calcola il rapporto dei dati superflui, eliminandoli definitivamente.

Da ciò possiamo dedurre che la decompressione dal formato MP3 a wave non porterà alla stessa forma d'onda del segnale utilizzato per la codifica, ma in termini di ascolto del file audio il risultato è lo stesso