Sistemi Informativi Multimediali

Appunti del corso a cura dello studente Gianni Adamo

A.A. 2022- 2023

Sommario

0. Premessa

Questi appunti sono stati scritti con il solo scopo di approfondire e assistere lo studio degli argomenti trattati nell'omonimo corso *"Sistemi Informativi Multimediali"*, tenuto dal prof. Walter Balzano presso l'*Università degli Studi di Napoli Federico II*, per il CDL in Informatica. Tale documento fa riferimento al corso dell'A.A 2022-2023.

Ulteriori informazioni sono contenute in fondo alla pagina corrente.

0.1 Riferimenti

Il materiale didattico utilizzato per la creazione di questo documento è così elencato:

- *Slides del corso "Sistemi Informativi Multimediali" – Prof. Walter Balzano*
- *Appunti degli anni precedenti (riferiti al medesimo corso)*
- *Annotazioni scritte a lezione*

Tutte le informazioni contenute in questo documento sono state accuratamente selezionate per coprire al meglio gli argomenti trattati durante il corso e fornire allo studente una guida per assimilare i concetti spiegati a lezione. L'autore non si assume alcuna responsabilità riguardo la completezza dell'elaborato e agli esiti che ne susseguono. Eventuali variazioni che il corso potrebbe subire negli anni a venire non saranno indicate all'interno del documento. Per ulteriori informazioni, consultare la sezione seguente *Info e Contatti.*

Tale documento è stato interamente scritto dallo studente Giovanni Adamo, iscritto al CDL in Informatica presso l'*Università degli Studi di Napoli Federico II.*

0.2 Info e Contatti

Di seguito, alcuni canali e piattaforme tramite cui è possibile raggiungermi, per contatti o visione di ulteriori produzioni (in ambito accademico e/o personale).

- **Email:** gianni.adamo@hotmail.it
- **GitHub:** https://github.com/Gazen27
- **Social:** Mi trovi su tutti i social come @gazen27

1. Struttura e archiviazione dei dati

In questa sezione, analizziamo i diversi tipi di dato e come questi ultimi sono memorizzati, oltre al modo in cui essi sono reperibili all'uomo.

1.1 Tipi di dato

Possiamo distinguere principalmente due tipologie di dati: **analogico** e **digitale**; definiamo analogico un segnale/informazione percepita come un'onda, che riproduce esattamente una vibrazione meccanica, mentre ci riferiamo ad un dato di tipo digitale quando quest'ultimo è rappresentabile mediante una sequenza di numeri (ad esempio il codice binario, una sequenza di 0 e 1).

Inoltre, è possibile passare da un segnale analogico a quello digitale mediante un processo di **digitalizzazione dei dati**, trasformando un tipo di dato analogico (come una fotografia), ad un dato digitale o "digitalizzato" mediante l'uso di appositi strumenti (ad esempio, uno scanner). Successivamente, si passa al **management** dei dati rappresentati (sezione *1.3 Archiviazione dei dati*).

1.2 Media e Multimedia

È possibile classificare in due macrocategorie i tipi di dato, ovvero:

- **Media:** corrispondono a tipi di informazione o rappresentazioni semplici, come dati alfanumerici, immagini, audio, video. Inoltre, possiamo differenziare i media in base alla loro relazione con la dimensione temporale, dividendoli in *statici* (ovvero privi di relazione temporale, come immagini o grafici), oppure *dinamici* (che hanno quindi una stretta relazione con la dimensione temporale, la quale influisce su di essi in base alla velocità con qui vengono rappresentati, come nel caso di audio, video, animazioni).

- **Multimedia:** semplicemente, una raccolta di vari media utilizzati contemporaneamente e che interagiscono tra loro (basti pensare ad una pagina web). La definizione di "multimediale" è tuttavia molto discussa.

Vediamo ora come questi dati vengono conservati e catalogati.

1.3 Archiviazione dei dati

I dati di cui abbiamo discusso, tuttavia, devono essere memorizzati e indicizzati in qualche modo; introduciamo dunque il concetto di **Database** (abbreviato DB), ovvero un sistema nel quale in nostri dati vengono conservati. Possiamo fare, inoltre, una distinzione importante su due "tipi" di Database:

- **DBMS (Database Management System):** include dati di tipo alfanumerico e, per effettuare una ricerca di questi ultimi, la corrispondenza (o "matching") deve essere di tipo testuale esatto.

- **MMDBMS (Multimedia DBMS):** ne fanno parte dati di tipo multimediale e il matching avviene mediante similitudini delle caratteristiche di questi ultimi.

Per accedere a tali dati conservati nei DBMS effettuiamo delle vere e proprie richieste al sistema per una risposta precisa: tali richieste vengono definite **query**.
Esistono diverse tipologie di query, di cui ne distinguiamo diverse tipologie:

(A) Query per similitudine, come la ricerca di un brano musicale nel quale sono presenti determinate note o un numero di colpi di tamburo

(B) Query per concetto, come la ricerca di tutte le sequenze in cui, in un film, compare un determinato attore che effettua una determinata azione.

(C) Query spazio/tempo, come la ricerca tra registrazioni di incontri calcistici, in cui un dato giocatore effettua un passaggio in una determinata zona di campo.

(D) Combinazioni più o meno complesse di (A), (B) e (C).

Per effettuare tali query, utilizziamo degli appositi linguaggi in grado di operare sui DBMS, come **SQL** (Structured Query Language). Tra i tipi di dato che andiamo a identificare nella scrittura del nostro DB, vi è la possibilità di definire dei **BLOB** (Binary Large Object), ovvero una stringa in codice binario di lunghezza variabile che si utilizza per la memorizzazione di dati in formato binario, come per le immagini.

Esistono inoltre diversi tipi di DBMS; uno dei più utilizzati è sicuramente il **RDBMS** (Relational DBMS), che possiede una struttura tabulare (rappresentiamo le entità che vogliamo memorizzare in formato di tabella). Differente è, ad esempio, un **OODBMS** (Object-Oriented DBMS), che combina le capacità dei DB con le note proprietà degli Object-Oriented (come ereditarietà, incapsulamento).

Introduciamo inoltre il concetto di **IR** (Information Retrieval), un sistema di ricerca che, a differenza di quanto espresso fino ad ora sui DBMS, non si basa sul ricercare un tipo di dato, bensì una *informazione*. Un IR opera esclusivamente su elementi di tipo testuale (un esempio di ricerca tramite IR potrebbe essere la ricerca di tutti i documenti che contengono informazioni circa un dato evento storico).
Tuttavia, il sistema IR presenta diversi limiti, come la difficoltà nella descrizione di alcuni elementi, una ricerca esclusivamente manuale e, di conseguenza, un maggior dispendio di tempo richiesto.

1.4 MIRS: Multimedia Indexing and Retrieval Systems

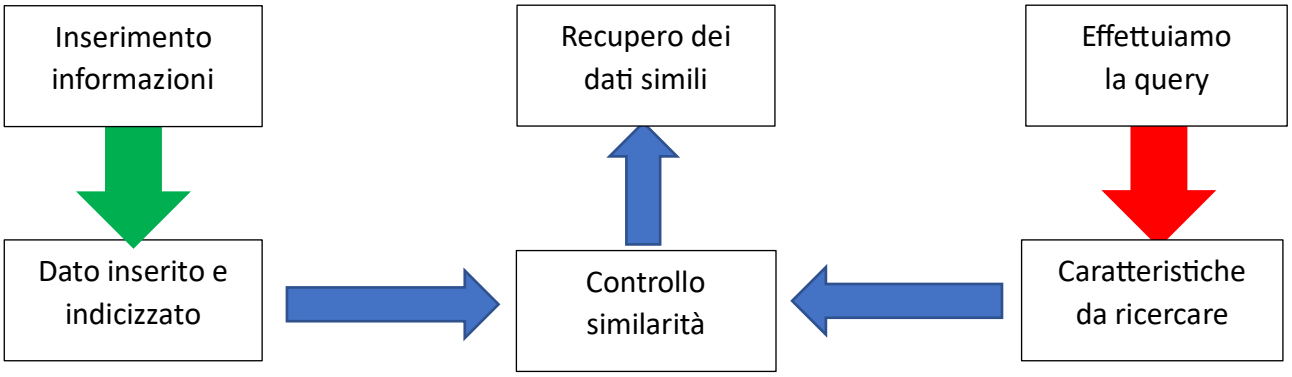
Il **MIRS** è un sistema di memorizzazione e recupero dati, il cui obiettivo è quello di *indicizzare* e *ricercare* i media (tra cui testi, immagini, audio e video), comprendendone le caratteristiche e catalogandole. Un MIRS, dunque, è molto diverso da un DBMS; infatti, per la progettazione di un MIRS è necessario conoscere la struttura e le caratteristiche principali dei dati multimediali, come ad esempio il modo in cui questi dati vengono memorizzati, le tecniche di compressione, le modalità di indicizzazione, ecc…

In un MIRS, deve essere inoltre possibile effettuare query su metadati, annotazioni, modelli di dati (o caratteristiche), esempi o applicazioni specifiche;

ad esempio, possiamo ricercare qualcosa mediante informazioni dettagliate su un oggetto (query su applicazioni specifiche), mostrare un'immagine/suono/disegno per ricercare un dato multimediale (query su esempi), e così via.

Un MIRS trova molteplici applicazioni, come in ambito della medicina, sicurezza, didattica, editoria, intrattenimento e copyright (ad esempio la ricerca in un DB di brevetti di eventuali oggetti simili a quello che dovrebbe essere brevettato).

A livello strutturale, il funzionamento del MIRS è così rappresentato:



2. Il Testo

In questa sezione analizziamo un particolare tipo di dato, nonché il più diffuso: il testo. Vediamo diverse tipologie di dati in formato testuale e varie codifiche di testo.

2.1 Tipi di testo

Possiamo distinguere diverse tipologie di dati in formato testuale, così elencate:

Testo piano: tra i tipi più conosciuti vi è il testo piano (o "plain text"), che corrisponde al testo semplice, una sequenza di caratteri alfanumerici privo di ulteriori attributi; è codificato mediante il codice ASCII (inizialmente rappresentato da 7 bit, esteso successivamente con l'EXTENDED ASCII per consentire la codifica anche delle lettere accentate). Successivamente si passa alla codifica UNICODE (21 bit), per consentire la rappresentazione di ancora più caratteri, passando così alla possibilità di avere oltre 1 milione di caratteri codificabili.

Testo strutturato: a differenza del testo piano, il testo strutturato (il più utilizzato per la diffusione di informazioni), è un tipo di testo che può essere formattato, presentando ciò degli *attributi* (ad esempio testo in grassetto, titoli, sottotitoli, ecc…). Esistono diversi standard per rappresentare questa tipologia di testo, come il formato Microsoft Word (.docx), formato LaTeX (.tex), o il Portable Document Format (.pdf); per ogni formato, utilizziamo gli strumenti adatti per estrarre le informazioni e ricavare del testo decodificato.

Testo compresso: consente di conservare spazio nella memoria di archiviazione, nonostante un file di testo occupi decisamente meno di altri media, come immagini o video. La compressione di file testuali, inoltre, è una compressione di tipo LOSSLESS (senza alcuna perdita), infatti il dato è esattamente lo stesso prima e dopo la compressione, senza subire alcuna variazione. Tale codifica è giustificata dal fatto che alcuni caratteri hanno una ricorrenza nettamente superiore rispetto ad altri, e ci sono gruppi di caratteri ripetuti frequentemente all'interno del documento.

Esistono diversi algoritmi di compressione di un documento testuale, come la codifica di Huffman, la codifica Run-Length e quella LZW. Queste tipologie di compressione verranno esaminate nelle sezioni successive.

Per effettuare la compressione di file testuali, possiamo ricorrere a diversi algoritmi di compressione che si occupano di comprimere il testo in modo da occupare meno memoria nello spazio di archiviazione (ricordiamo comunque che, rispetto agli altri tipi di file media, quelli di tipo testuale sono i più "leggeri").

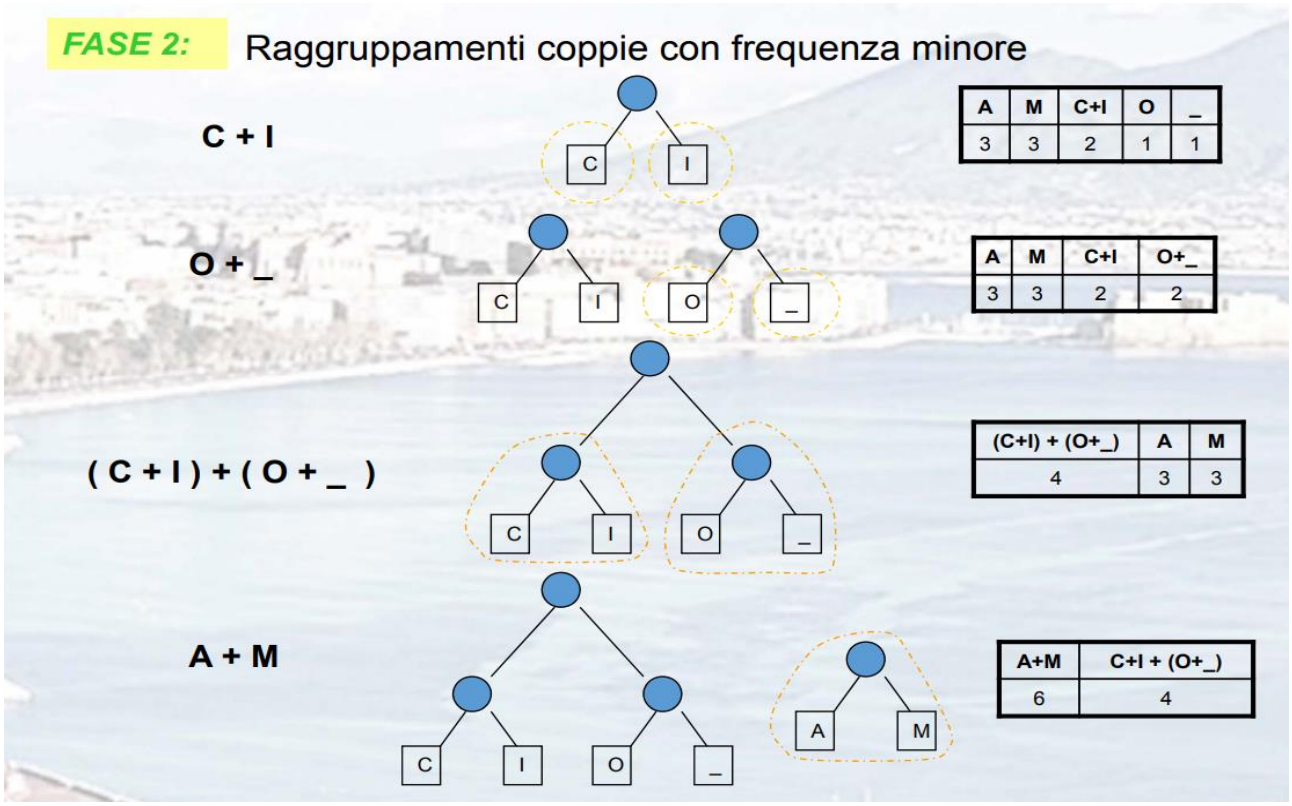
2.2 Codifica di Huffman

Questa particolare codifica è basata sull'analisi statistica del dato da comprimere, in particolare sulla frequenza con la quale si ripetono i suoi elementi (caratteri). Maggiore è la variazione di caratteri da comprimere, maggiori saranno le prestazioni.

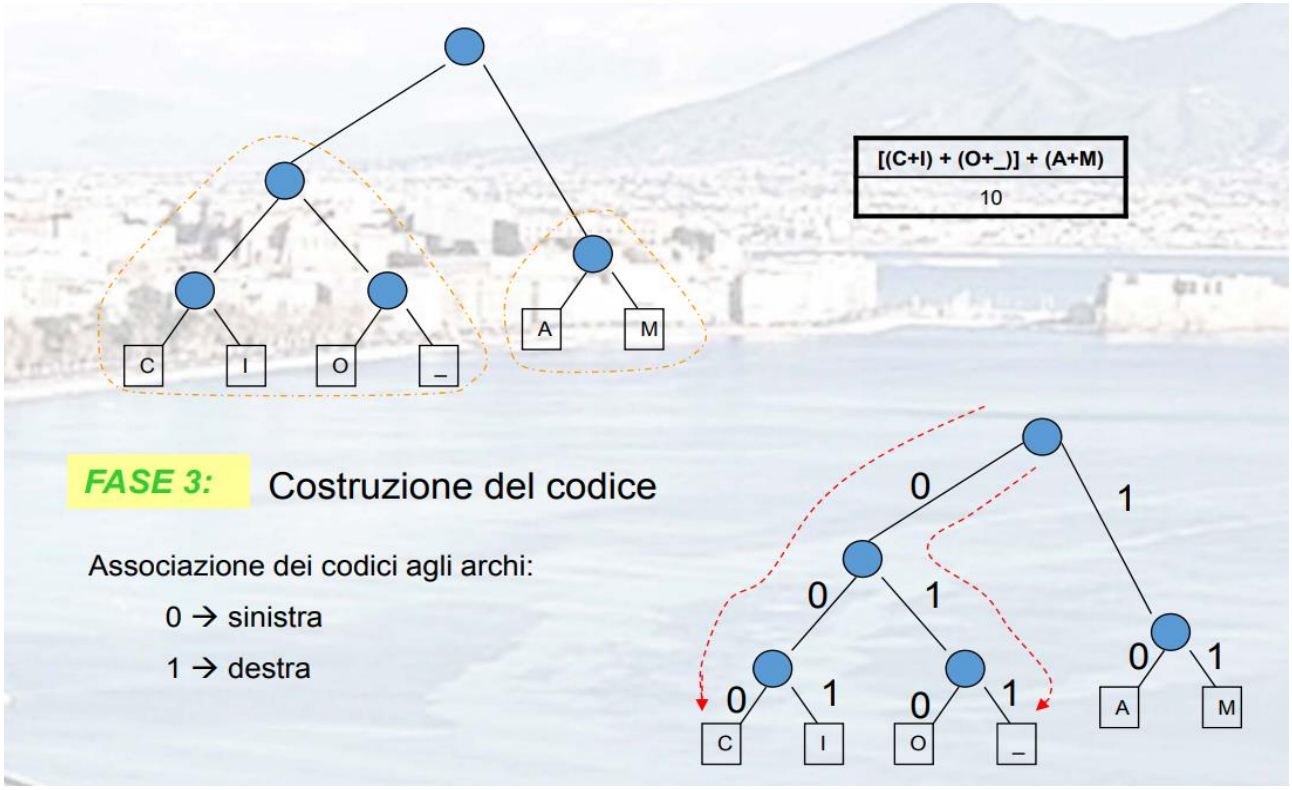
Sarà inoltre possibile utilizzare una codifica variabile, associando ai valori che si presentano più frequentemente, un numero di bit minori. Vediamo come si compone il processo di compressione secondo la codifica Huffman:

Fase 1: dopo aver preso la stringa da comprimere (S), e aver convertito i caratteri che la compongono in codice ASCII, passiamo all'analisi delle frequenze di tali caratteri.
A partire dalla stringa S, otterremo la stringa codificata S'.

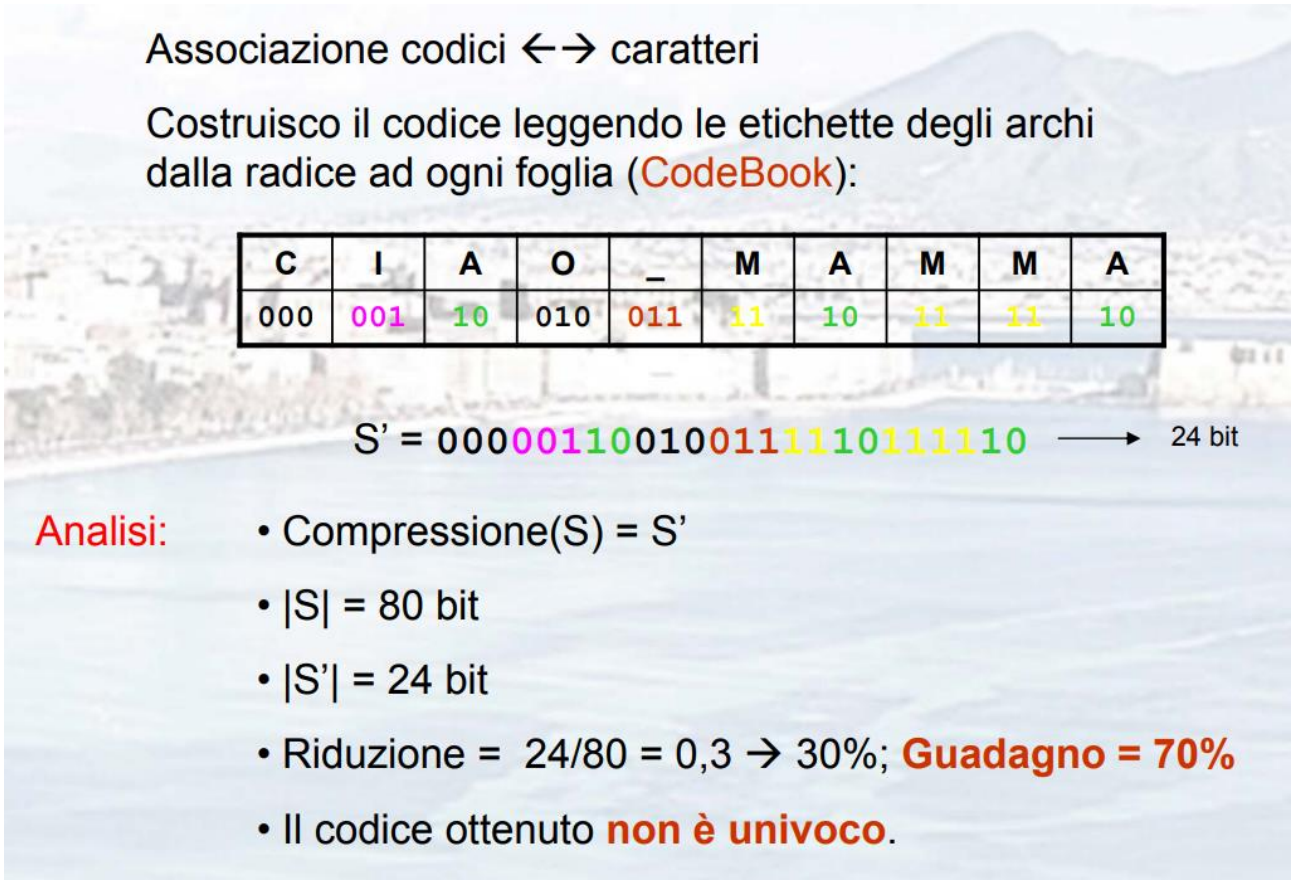
Fase 2: raggruppiamo le coppie con frequenza minore, costruendo un albero binario, e continuiamo a raggruppare le coppie successive (in ordine crescente di frequenza)



Fase 3: dopo aver costruito interamente l'albero, e aver dunque raggruppato tutte le coppie, associamo dei codici agli archi, come indicato in figura, sull'albero risultante.



Fase 4: alla fine, associamo ad ogni carattere il codice ottenuto percorrendo l'albero risultante a partire dalla radice, fino alla foglia desiderata (contenente, dunque, il carattere che stiamo andando a codificare); come possiamo vedere dall'esempio sottostante, la stringa iniziale (S) aveva lunghezza di 80 bit, mentre quella compressa (S'), una lunghezza di soli 24 bit, per un guadagno del 70%.



Come possiamo notare, però, tale codifica potrebbe generare ambiguità, in quanto il codice ottenuto non è univoco, e potrebbero quindi esistere stringhe diverse che hanno la stessa codifica risultante. Pertanto, l'algoritmo della codifica di Huffman non è oggigiorno utilizzato per effettuare compressioni di file testuali.

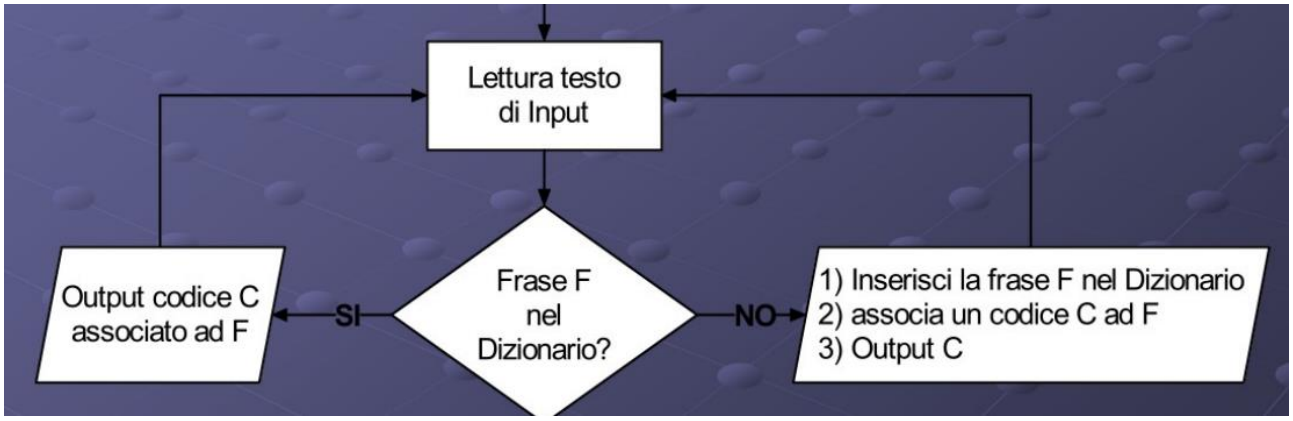
2.3 Codifica Run-Length

Questo tipo di codifica è molto semplice, in quanto si limita a ridurre le ripetizioni di caratteri sostituendo ad esse un RUN, ovvero un elemento composto da simbolo speciale, il carattere ripetuto e il numero di ripetizioni di quest'ultimo, come segue:

Un esempio di codifica Run-Length è la seguente: *eeeeeeetnnnnnnnn* → *@e7t@n8*
infatti il carattere *e* viene ripetuto 7 volte, *t* non ha alcuna ripetizione ed *n* 8 volte.

2.4 Codifica LZW

Questo tipo di codifica sfrutta le ripetizioni di gruppi di caratteri e/o intere frasi, inserendole in un dizionario, o verificandone la presenza. Dopo aver ricevuto il testo in input, ne verifica la presenza nel dizionario, rilasciando in output il codice (C) associato alla frase (F) presa in input; nel caso in cui la frase non è presente nel dizionario, prima di rilasciarne il codice, viene inserita.



Le prestazioni sono migliori per input che presentano molte ripetizioni, ed è preferibile utilizzarlo su linguaggi naturali (italiano, inglese, ecc…)

Dopo aver analizzato tutti gli elementi che contraddistinguono i dati di tipo testuale, possiamo passare a quelli di tipo audio.

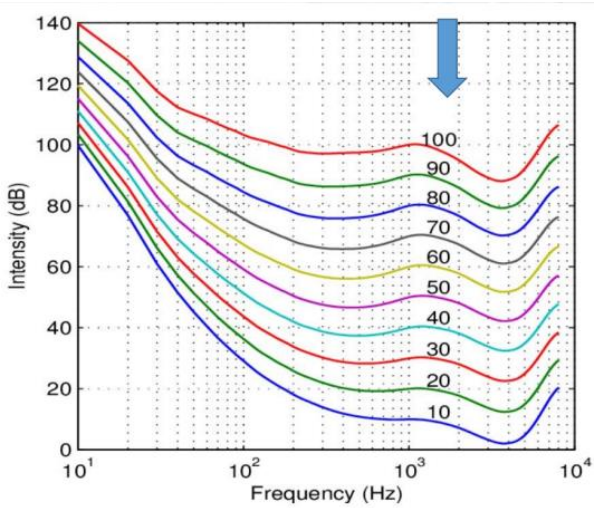
3. Audio

Un segnale audio è causato da variazioni di pressione dell'aria; le caratteristiche che descrivono un segnale audio sono **Ampiezza** e **Frequenza**, variabili nel tempo.

Soglia udibile e zone di percezione: le frequenze udibili si trovano nell' intervallo tra i 20 e i 20.000 Hz, ma la maggiore sensibilità è tra i 1.000 e i 5.000 Hz (zona di percezione dove si concentra buona parte del contenuto informativo sonoro, come il parlato, e il nostro udito si è adeguato e sensibilizzato di conseguenza). Le capacità uditive dell'uomo non sono lineari né rispetto all'ampiezza, né alla frequenza.

Curve isofoniche: descrivono quali livelli di pressione sonora percepiamo con ugual "volume" (phon) al variare della frequenza; le curve isofoniche, quindi, descrivono il rapporto tra frequenza (Hz) e **Intensità** (dB).

L'unità di misura della pressione acustica è il phon, e possiamo notare come al variare della frequenza, alcune onde ci risultano uguali (nonostante la variazione di dB).



3.1 Conversioni di segnale (ADC e DAC)

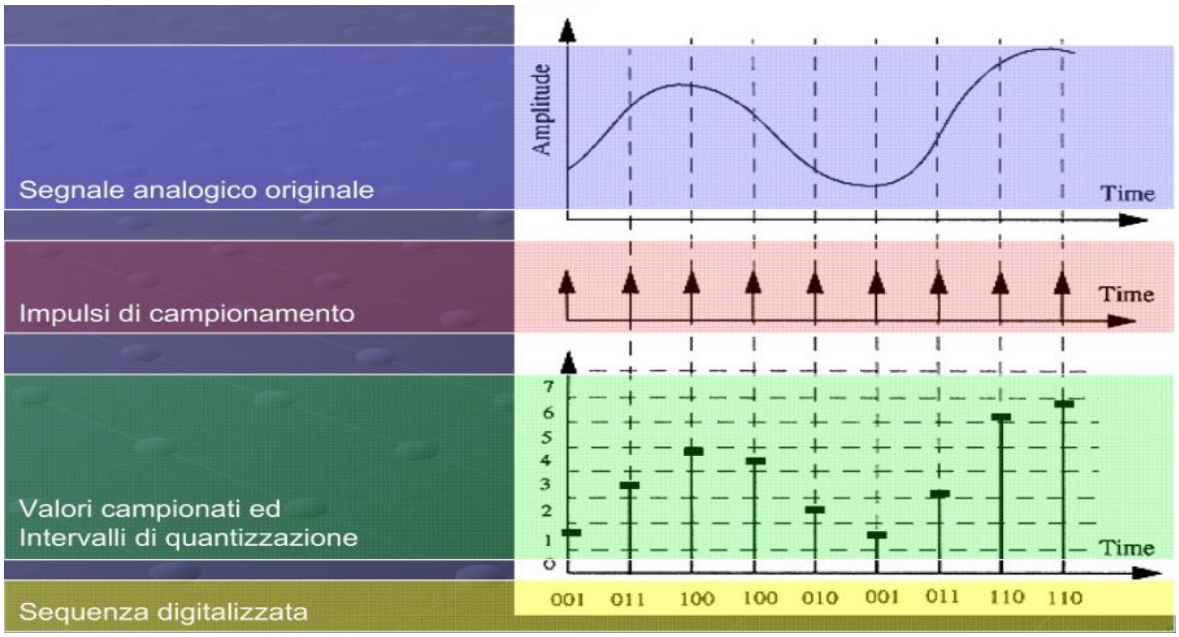
Un segnale acustico può essere in formato analogico o digitale; tuttavia, è possibile effettuare una conversione da uno all'altro, secondo quanto indicato di seguito.

ADC (Analog to Digital Converter): processo di conversione da segnale analogico a segnale digitale; ciò si suddivide in tre fasi fondamentali, ovvero:

- **Campionamento:** prelievo dei valori assoluti dal segnale (prendiamo quindi dei valori significativi che identificano il segnale) ad intervalli discreti di tempo (gestiti da un clock); tali valori, sono ancora di tipo analogico.

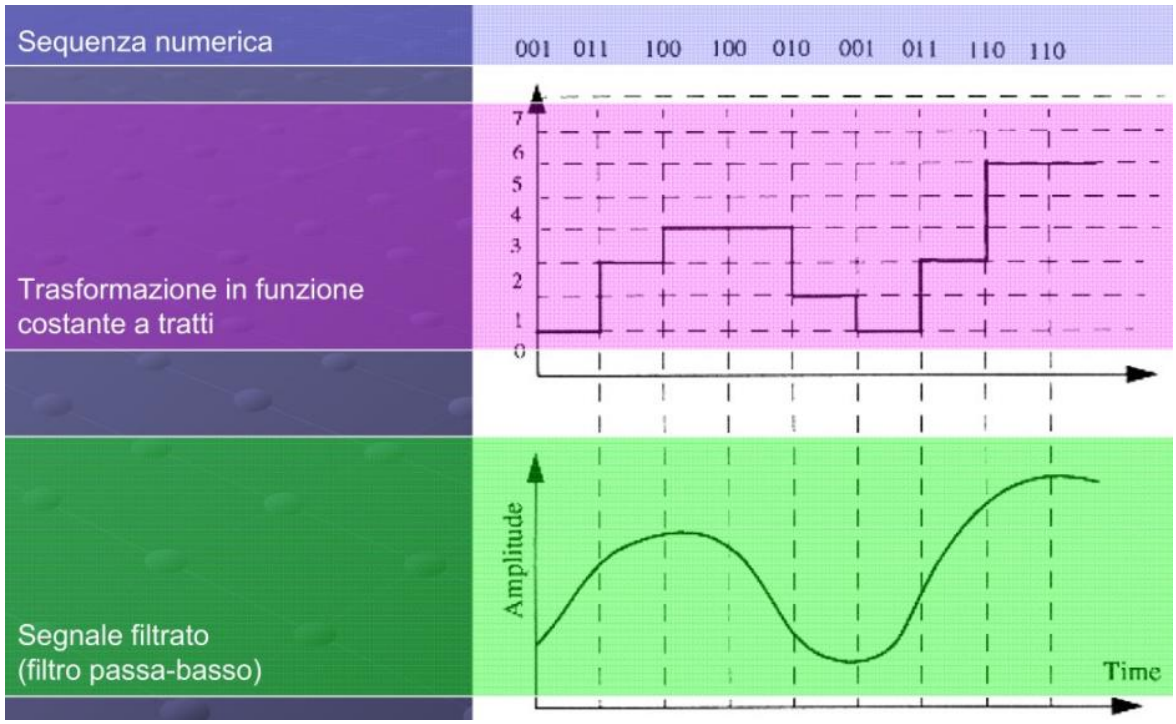
- **Quantizzazione:** conversione dei valori continui in valori discreti, suddividendo l'intervallo del segnale in un numero fisso di sotto-intervalli, la cui grandezza è detta *passo di quantizzazione*. Viene poi assegnato un valore ad ogni intervallo

- **Codifica:** vengono codificati i valori precedentemente assegnati. Maggiore sarà la frequenza di campionamento e il numero di livelli di quantizzazione, tanto più il segnale digitalizzato sarà fedele all'originale.



ADC – Analog to Digital Converter

DAC (Digital to Analog Converter): al contrario dell'ADC, questo processo inverso, partendo da una sequenza numerica (segnale digitale), ne ricava una funzione costante a tratti (non molto precisa), per poi applicare il *filtro passa-basso*, ovvero un filtro che impedisce il passaggio di frequenze alte (da qui il nome "passa-basso").



DAC – Digital to Analog Converter

3.2 Teorema di Nyquist

La frequenza di campionamento è strettamente dipendente dalla frequenza massima del segnale da convertire, infatti, come afferma il teorema di Nyquist:

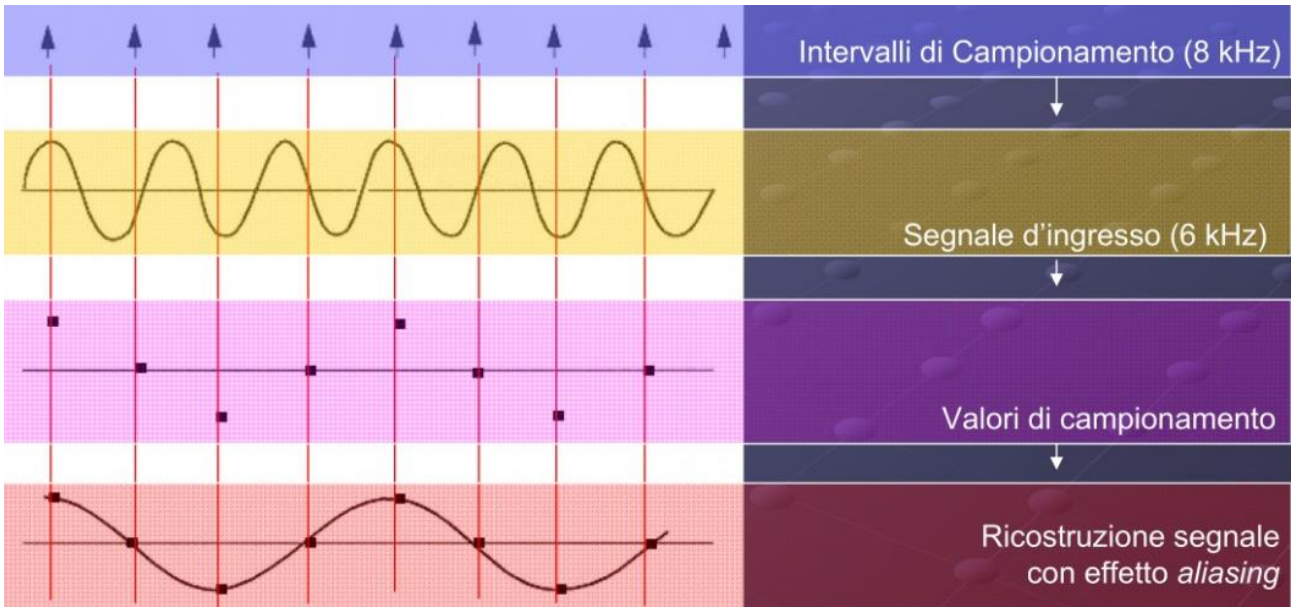
Se in un segnale analogico c'è una componente con frequenza fino a f Hz, allora la frequenza di campionamento dovrebbe essere almeno di 2f Hz.

Alcuni esempi di qualità standard delle **frequenze di campionamento:**

- CD-Audio: 44.1 kHz
- DAT (Digital Audio Tape): 48 kHz
- Telefono: 8 kHz

Come possiamo notare, siccome la voce si aggira tra i 1.000 e i 5.000 Hz, il telefono non rispetta il Teorema di Nyquist, campionando il segnale a 8 kHz invece di 10 kHz.

Se si utilizza una frequenza di campionamento troppo bassa, non sarà possibile ricostruire la curva in modo corretto, perdendo parte dell'informazione dal segnale e ottenendo quindi una ricostruzione errata, generando l'**effetto Aliasing**.

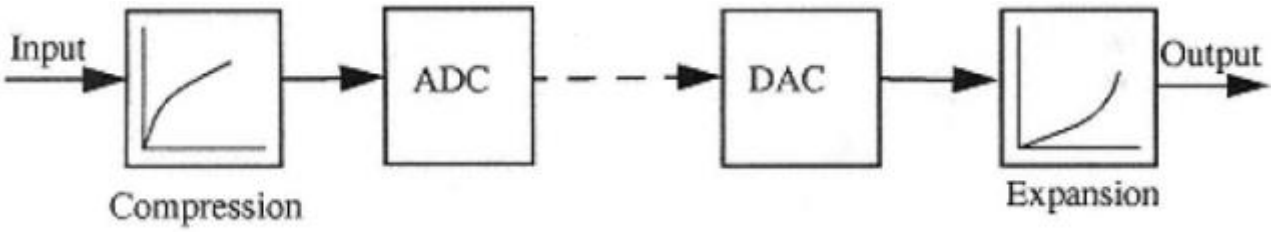


Frequenza di campionamento bassa ed effetto Aliasing

Scegliere una frequenza di campionamento troppo alta, tuttavia, potrebbe ugualmente generare "errori" o quello che viene definito come *rumore*.

3.3 Compressione audio: Companding

Il Companding è un metodo di compressione capace di aumentare la fedeltà della traccia risultante rispetto all'originale; permette a segnali con un ampio intervallo dinamico di essere trasmessi su attrezzature che hanno una minore capacità. Questa tecnica consiste nel comprimere il segnale, portandolo da analogico a digitale (ADC), per poi portarlo di nuovo da digitale ad analogico (DAC), espandendolo, perdendo così meno informazioni possibili; inoltre, viene eseguito un tipo di campionamento variabile, quindi campionando ad intervalli di tempo non sempre uguali (nelle zone con cambi di frequenza più elevati si effettuano più campioni).



3.4 Predictive Coding e DPCM

Con il Predictive Coding, invece di codificare il valore del campione da trasmettere, viene codificata la differenza tra la predizione del valore del campione e il valore del campione attuale (DPCM – *differential pulse-coded modulation*).

Il valore della predizione si calcola basandosi sui valori precedentemente assunti dal segnale, rendendolo quindi un valore noto sia al codificatore che al decodificatore; se la differenza tra un valore e una predizione risulta essere eccessivamente elevata, vengono introdotti degli appositi algoritmi di correzione.

L'efficacia del Predictive Coding si basa sul fatto che:

- Campioni vicini sono significativamente correlati.
- Occorre un numero inferiore di bit per codificare una differenza.

(N.B. il Predictive Coding è una tecnica utilizzabile in diversi ambiti; in questo caso, facciamo riferimento all'applicazione in ambiente di compressione di segnali audio)

3.5 Mpeg Audio

La tecnica di compressione Mpeg audio si basa sul *mascheramento*, facendo sì che suoni di maggior intensità "coprano" quelli a bassa intensità, che potranno essere ignorati (compressi); questa tecnica, tuttavia, è *lossy* (a perdita), e ha delle frequenze di campionamento prestabilite di 32, 44.1 e 48 kHz.

Il rapporto di compressione ha anch'esso un range prestabilito, che dipende dalla frequenza di campionamento (fino ad un rapporto di 6:1 la "perdita" non è percepibile ad orecchio).

3.6 Audio sintetico (MIDI)

L'acronimo **MIDI** (Musical Instrument Digital Interface) indica il protocollo standard per l'interazione di strumenti musicali elettronici; questo formato non contiene musica pre-registrata, ma le specifiche e le direttive per riprodurla, quindi informazioni che vengono interpretate da sistemi specifici (Hardware o Software), realizzandone l'esecuzione, come se fosse uno spartito musicale, che contiene quindi solo direttive e non musica effettiva. Un esempio di direttiva in MIDI è:

*"esegui la nota **N** con una durata **T** e con lo strumento **S**"*

Un file MIDI ha dimensioni molto ridotte, ma la riproduzione non è univoca.

4. Le Immagini

Un altro tipo di file media sono le immagini digitali, la base di molti file di tipo multimediale. L'origine delle immagini digitali è varia (macchine fotografiche, Scanner, Fotogrammi di filmati, Disegni elettronici, ecc…).

4.1 Colori: percezione e sintesi

Gli algoritmi di ricerca e indicizzazione di immagini devono essere basati su una buona **rappresentazione dei colori**; differenti lunghezze d'onda della luce visibile producono differenti sensazioni di colore (diverse tonalità).

Le proprietà fisiche delle radiazioni di colore sono:

Luminanza (illuminazione) **Tinta** (colore) **Saturazione** (purezza)

I colori sono riproducibili tramite Sintesi additiva o sottrattiva (dei colori primari).

Sintesi Additiva

Partendo dalle frequenze di tre colori primari, per convenzione Rosso, Verde e Blu (RGB), è possibile ottenere altre frequenze di colori (si producono quindi per mescolanza di tre colori primari).

Sintesi Sottrattiva
Partendo dalla radiazione assorbita, le superfici colorate sottraggono alla nostra visione una parte dello spettro visibile; osserviamo dunque l'effetto prodotto dalla riflessione dei colori.

Lo spazio di colori RGB, inoltre, è definito uno spazio di colori *tridimensionale*.

Tra i sistemi di rappresentazione dei colori, troviamo il sistema SPD (Spectral Power Distribution), per il quale rappresentiamo ogni colore "fisico" associandolo ad una distribuzione dell'energia radiante alle varie lunghezze d'onda, ottenendo così un sistema molto più preciso e accurato, ma ha una complessa rappresentazione numerica e una difficile descrizione delle proprietà del colore.

Quando parliamo di **modelli di colore**, ci riferiamo a modelli "matematici" che permettono di rappresentare i colori delle immagini in forma numerica.
Tra i modelli più in uso troviamo:

- **RGB** (Red, Green, Blue – Sintesi Additiva)
- **CMYK** (Cyan, Magenta, Yellow, Black – Sintesi Sottrattiva)
- **YUV** (Ambito analogico)
- **YCbCr** (Digitale di YUV; Y = Luminanza, Cb = Croma blue, Cr = Croma red)
- **HSV** (Hue, Saturation, Value)

È inoltre molto importante considerare il numero di bit assegnati per pixel per definire il colore: con 8 bit/pixel si possono avere $2^8 = 256$ colori per pixel.

L'occhio umano non è capace di cogliere qualsiasi sfumatura di colore, quindi non è necessario avere più sfumature di quelle che l'occhio umano può vedere. Nella ricerca e indicizzazione di immagini basata sul colore, bisogna assumere che siano tutte rappresentate nello stesso spazio di colore (tuttavia, data ad esempio un'immagine RGB non è possibile interpretarla correttamente, in quanto non vengono indicate le definizioni delle primitive RGB, quindi non è possibile interpretare l'immagine in uno spazio comune, e non conosciamo i valori della correzione gamma applicati al dispositivo).

4.2 Correzione Gamma

Consideriamo l'ipotesi per cui ogni dispositivo ha le sue caratteristiche e il proprio margine d'errore sulla lettura di frequenze luminose; conoscendo tale margine, è possibile bilanciare le frequenze ed ottenere un segnale quanto più fedele possibile.

Ogni strumento fisico di acquisizione o riproduzione dei colori applica una funzione non lineare all'intensità di luce catturata o emessa, in relazione al segnale in Volt emesso o applicato. Vediamo come agisce dunque la correzione gamma:

$$\text{Luminanza} = V^{\gamma} \qquad \text{Correzione gamma} = 1/\gamma$$

La correzione della gamma è quindi un'operazione utilizzata per codificare e decodificare la luminanza in un sistema video o fotografico.

4.3 Tipi di immagini digitali

Un'immagine digitale può essere:

- **Raster** (o **Bitmap**): come da traduzione, definibile tramite una griglia (o più precisamente una scacchiera) in cui ogni slot è chiamato pixel, e ad ognuno è associato un colore. La sua memorizzazione può essere *LOSSY* o *LOSSLESS.*

- **Vettoriale**: immagine definita dall'insieme di primitive geometriche (come punti, linee, curve, poligoni), ai quali sono associati vari attributi, tra cui il colore; hanno una qualità altissima e un peso irrisorio (come per i MIDI, questo formato non contiene il prodotto in sé, ma le direttive per crearlo; in questo caso, ogni luogo geometrico è rappresentato da un'equazione).
  Richiede tuttavia maggiore capacità di elaborazione (macchine più potenti).

Per migliorare la qualità delle immagini Raster, viene usata la Super-Resolution, la quale ha due modi di agire:

- **Single-Frame**: tramite algoritmi di AI riesce a ricostruire parte dell'informazione e, applicandone filtri, rende l'immagine ancora più nitida.

- **Multi-Frame**: partendo da più foto uguali, vengono applicati meccanismi di correzione basati su differenze tra i frame; i moderni cellulari con fotocamere multiple sfruttano questa tecnica per correggere ombre e sfocature.

4.4 Compressione di immagini

Considerando il fatto che l'occhio umano è limitato e molta dell'informazione non viene percepita, è possibile utilizzare come metodo di compressione la **Ridondanza Spaziale**; questa tecnica consiste nel rimuovere i punti adiacenti a determinati pixel (ciò è possibile in quanto pixel adiacenti hanno colore molto simile). Questa tecnica, nonostante in caso di zoom sarà possibile verificare che parte dell'informazione è stata scartata, all'occhio umano non risulterà differente dall'originale e non verrà alterato il significato dell'immagine.

I tipi di compressione possono essere:

- **LOSSLESS**: permettendo di ricostruire perfettamente l'oggetto (Huffman).
- **LOSSY**: permettendo di ricostruirlo solo in parte.

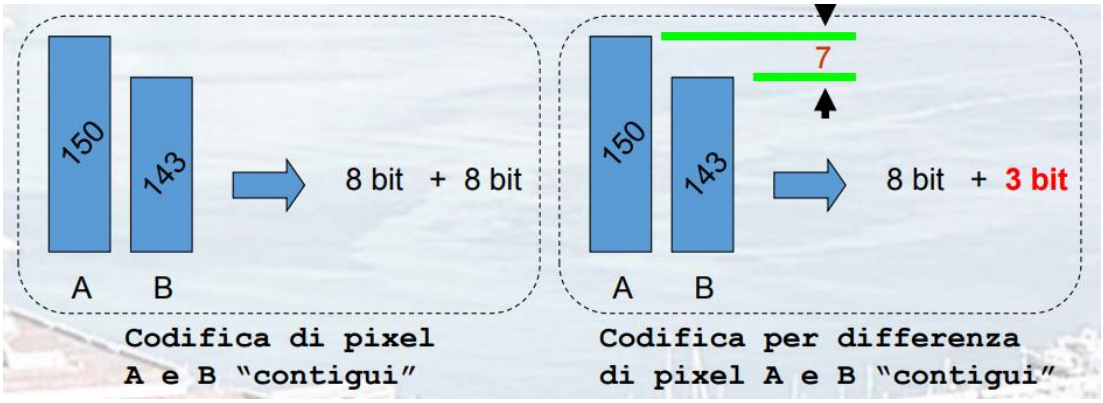
Tra le compressioni *LOSSY* vediamo in dettaglio le seguenti:

Compressione con sottocampionamento

Consiste nell'utilizzo della *Ridondanza Spaziale* (vista in precedenza); in fase di decodifica vengono ricostruiti alcuni dei pixel mancanti mediante interpolazione, in maniera approssimativa. Siccome il nostro occhio è più sensibile alla *luminanza* rispetto alla *crominanza*, tale ricostruzione verrà effettuata solo su quest'ultima.

Predictive Coding

Come visto nel caso del *predictive coding* per i file audio, valori vicini sono fortemente correlati; se A e B sono due pixel vicini, e abbiamo già codificato A, invece di codificare per intero anche B, ci basta codificare la differenza tra i valori di A e B (utilizzando quindi un minor numero di bit).



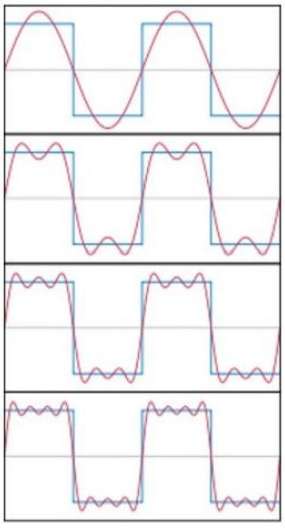
Compressione mediante trasformazione

Tramite questo metodo, si considera ogni pixel dell'immagine in modo indipendente, convertendo gli elementi dell'immagine in un insieme di coefficienti indipendenti; un'immagine viene suddivisa in sottoimmagini rettangolari su cui viene applicata una trasformazione (dal dominio spaziale a quello delle frequenze).

Le trasformazioni principalmente applicate sono la **DCT** (Discrete Cosine Transform) e la **DFT** (Discrete Fourier Transform), che stiamo per analizzare.

Dopo aver applicato le eventuali trasformazioni, allora i dati nel dominio delle frequenze sono pronti per essere compressi (Huffman, Run-Length, ecc…).

4.5 Serie di Fourier



Fourier dimostrò che un qualsiasi segnale periodico $f(x) = f(x+T)$ può essere scomposto in una somma di infiniti segnali sinusoidali; un segnale, quindi, può essere decomposto in modo da ricavarne solo l'informazione principale, eliminando la portante (che è periodica e, pertanto, non informativa).

Tale trasformata crea una decomposizione considerata unica. In sostanza, analizzando lo spettro di un'immagine o di un suono, è possibile decomporlo in frequenze, eliminando la portante.

Con la trasformata di Fourier si passa dal dominio spaziale al dominio delle frequenze.

4.6 Immagini JPEG

La immagini in formato JPEG (Joint Photographic Experts Group) corrispondono a un modello di rappresentazione dell'immagine *LOSSY* a colori 24 bit. Costituiscono inoltre un modello Open Source. Il JPEG ha un ottimo rapporto qualità/peso, basandosi sul limite della percezione visiva dell'uomo e sulla trasformazione DCT.

Le fasi della sua rappresentazione iniziano con la **preparazione in blocchi**, decomponendo l'immagine in blocchi 8x8 e convertendo lo spazio di colore da RGB a YUV, per poi effettuare il **passaggio al dominio di frequenze** (applicando la DCT). Successivamente vi è la **quantizzazione**, tramite la quale vengono arrotondati gli opportuni valori (alti o bassi delle frequenze). Infine, vi è la fase di **codifica**, mediante codifica Run-Length o Huffman, scansionando l'immagine "a zigzag".

4.7 Immagini frattali

Le immagini frattali, così chiamate per la loro somiglianza ai frattali (figure geometriche che si ripetono "su sé stesse", rimpicciolendosi di volta in volta), rappresentano un intermedio tra Raster e Vettoriali per la memorizzazione dell'immagine; in sostanza, è come se ci riferissimo ad un'immagine vettoriale con la proprietà di ripetizione dei dettagli di un Raster.

Quella delle immagini frattali è un metodo di rappresentazione *LOSSY*, che cerca di decomporre l'immagine in parti più semplici e piccole, quali figure geometriche elementari, da memorizzare insieme alle istruzioni per la ricomposizione (anche queste, con approccio matematico).

5. Il Video

Un video digitale è riconducibile all'essere una sequenza di fotogrammi o immagini visionate a frequenza; la velocità con cui tali fotogrammi (detti Frame) scorrono è detta **Frame Rate**, e lo misuriamo tipicamente in **FPS** (Frame Per Second). Un frame rate più alto genera un video più fluido, ma necessita di una banda per la trasmissione maggiore, dato l'elevata richiesta di dati (il video rimane il media più "pesante" tra tutti). Esistono due tecniche di composizione dell'immagine in video:

- **Progressiva:** l'immagine viene costruita pixel per pixel a partire dall'angolo in alto a sinistra e procedendo via via per righe.
- **Interlacciata:** la composizione dell'immagine avviene componendo prima tutte le righe dispari, poi tutte quelle pari.

Come abbiamo detto, il video di per sé è molto pesante, per tanto risulta necessaria la sua ottimizzazione e compressione.

5.1 Codifica Intraframe / Interframe

Distinguiamo due tipologie di codifica video: la **codifica intraframe**, che descrive ogni singolo frame, rispettando dunque l'approccio di quantizzazione come *sequenza di immagini statiche*;

questa codifica è utile per sequenze video molto movimentate, e semplifica l'accesso diretto ad una precisa scena. La **codifica interframe**, invece, partendo da un fotogramma iniziale (descritto in intraframe), descrive i cambiamenti che intercorrono tra un frame ed il successivo (si basa quindi sul *predictive coding*); utile per video con pochi frame, ma risulta complesso l'accesso diretto ad una scena.

5.2 Compressione video

Per la compressione di video, viene applicato un processo che si basa sulla **ridondanza temporale**, trattata mediante la similitudine di fotogrammi adiacenti. Ogni fotogramma viene diviso in blocchi e si cerca la maggiore corrispondenza tra i blocchi dei fotogrammi adiacenti; per ogni coppia di blocchi, si determina lo **spostamento** del blocco (Motion Vector) e la **differenza** tra i due blocchi.

Ciò che sarà conservato è il Motion Vector e la relativa differenza.

6. Database Multimediali

Come accennato nella *sezione 1.4*, un **MIRS** (Multimedia Indexing and Retrieval System) ha il compito di indicizzare, conservare e gestire i dati multimediali, comprendendone le singole caratteristiche e catalogandoli. Possiamo definirlo come un "database intelligente", in grado di comprendere gli aspetti di ogni tipo di dato.

6.1 Architettura di un MIRS

Definiamo alcune caratteristiche fondamentali per l'architettura di un MIRS.

Modularità:
Le varie funzionalità fornite sono di modulari, in modo da garantire forte flessibilità e adattamento, gestione degli aggiornamenti e aggiunta di moduli opzionali, come:

- **Thesaurus Manager:** gestisce sinonimi e relazioni tra parole per query precise.
- **Integrity Rule Base:** testing dell'integrità di una determinata applicazione.
- **Context Manager:** controlla il contesto dell'applicazione.

Distribuzione:
Possibilità di gestione dei dati multimediali tramite un *Client-Server* e di accessi simultanei (come per video-on-demand o librerie digitali).

Le due funzionalità e richieste principali per le quali il MIRS viene impiegato, sono l'**indexing**, ovvero l'indicizzazione dei vari dati, ai quali il MIRS fornisce una chiave univoca (dopo averli inseriti ed estratto le relative caratteristiche, che verranno inviate al server), e il **retrival**, che è l'effettivo recupero dei dati (utilizzando, appunto, le chiavi generate durante il processo di indexing), tramite query specifiche.

6.2 Modello dei dati e Requisiti del MIRS

In un MIRS (o qualsiasi altro MMDBMS), bisogna considerare, tra le varie finalità della modellazione di un dato, una specifica dei diversi livelli di astrazione dei dati multimediali, considerando cioè le diverse caratteristiche degli oggetti da contenere.
Un modello di dati in un MIRS deve descrivere:

- **Proprietà statiche:** riguardano gli oggetti stessi che costituiscono i dati multimediali, le loro relazioni e gli attributi

- **Proprietà dinamiche:** riguardano invece le interazioni tra i vari oggetti, le operazioni disponibili su di essi e l'interazione con gli utenti

Il funzionamento di un MIRS è fortemente condizionato dal modello di dati che ospita; deve inoltre rispettare dei requisiti fondamentali per continuare ad operare ed essere ancora funzionale nel tempo, tra cui:

- **Estensibilità:** deve essere adattabile e utilizzabile anche a nuovi tipi e formati, non previsti dal principio.

- **Flessibilità:** deve permettere l'inserimento e la ricerca in vari livelli di astrazione (posso gestire il dato secondo qualsiasi sua caratteristica).

- **Predisposizione:** per rappresentare al meglio i dati multimediali (semplici e composti), e le relazioni spazio/temporali che intercorrono tra di loro.

- **Efficienza:** nei metodi di memorizzazione e di ricerca.

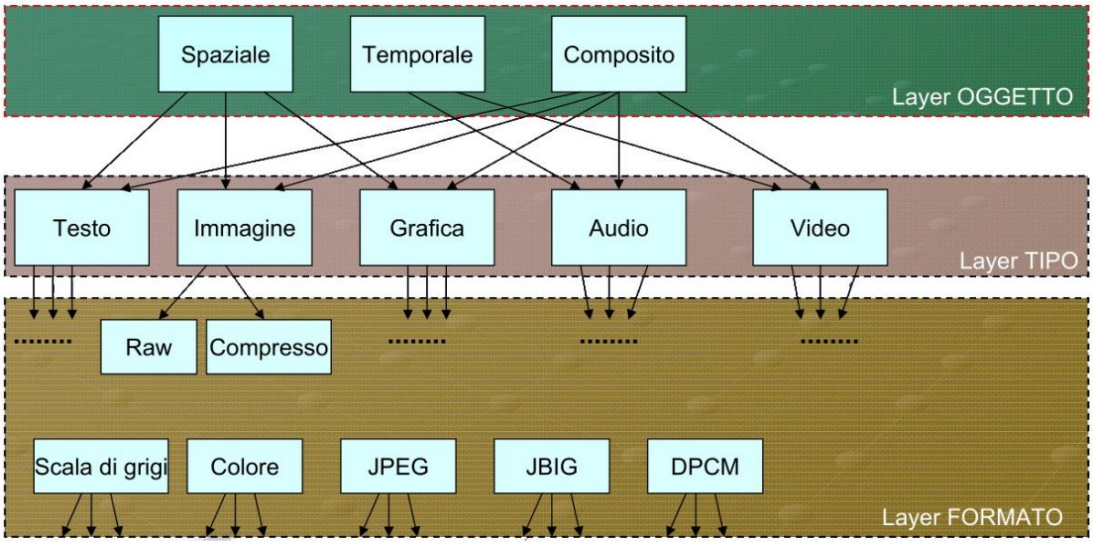
- **Incapsulamento:** riguarda codice e dati, incapsulati in una singola unità chiamata oggetto; questa è una prerogativa del paradigma *Object Oriented*, meglio adattabile alla modellazione dei dati multimediali.

6.3 Layer: Oggetto, Tipo e Formato

Possiamo analizzare la strutturar di un MIRS secondo tre Layer specifici:

- **Layer Oggetto:** costituito da uno o più media con specifiche relazioni, che possono essere di tipo spaziale (come la dimensione di ogni immagine) o temporale (come la durata di un video).
- **Layer Tipo:** contiene i tipi di media comuni; a questo livello, sono specificati gli attributi del media (come la dimensione in pixel, istogramma del colore, ecc…).

- **Layer Formato:** specifica, appunto, il formato in cui il media è memorizzato (ad esempio un tipo RAW, file compresso, il formato di compressione, ecc…).



6.4 User interface

Un'interfaccia utente MIRS ha tra le principali caratteristiche, l'obbligo di fornire strumenti per l'**inserimento** di oggetti nel database, tenendo in considerazione tutte le caratteristiche (a differenza di un normale Database, in un MIRS i dati sono costituiti da media, e quindi non hanno una struttura prefissata); inoltre, deve fornire gli strumenti per **definire le query** al meglio e raffinare le esigenze di ricerca, oltre a **presentare i risultati**, in maniera diretta ed efficiente dopo averne effettuata una. L'interfaccia utente, dunque, deve essere user-friendly.

6.5 Fase di ricerca e raffinamento query

Possiamo effettuare una ricerca secondo due criteri differenti, ovvero per:

- **Specifica:** una query da interpretare per passare da semplice testo, a un "concetto", che descriva pienamente un attributo (esempio: *auto rossa = insieme di pixel a forma di "oggetto auto" di colore principalmente rosso*).
- **Esempi:** strumenti che consentono l'inserimento di media a cui fare riferimento per trovare il dato richiesto (immagine, traccia audio, ecc…).

Bisogna però considerare l'eventuale incertezza da parte dell'utente riguardo la sua stessa ricerca (criteri troppo vaghi o scarsa sicurezza riguardo il dato da ricercare); pertanto, il sistema deve essere capace di **raffinare la query** effettuata da parte dell'utente, restituendo un risultato quanto più pertinente possibile e permettendo di riutilizzare quello stesso risultato per effettuare una nuova ricerca ancora più accurata. Spesso una conoscenza del profilo dell'utente e/o feedback riguardo le ricerche possono aiutare a migliorare e "personalizzare" ulteriormente il risultato.

Gli oggetti indicizzati in un MIRS possono essere ricercabili mediante diverse *features* e *attributi*. Il processo di ricerca, infatti si basa proprio sull'estrazione e la comparazione di tali **features**, tra quelle espresse nella query e quelle invece del dato da ricercare (l'estrazione di features non avviene in tempo reale, ma al momento dell'inserimento e indicizzazione del dato, come visto nella *sezione 1.4*)

Possiamo inoltre identificare diversi tipi di *features*, quali:

- **Metadata:** informazioni di contesto che NON descrivono il dato e non ne interpretano il contenuto, ma sono "di contorno" (come autore, titolo, data).

- **Annotazioni testuali:** descrizioni testuali del contenuto di un dato multimediale, soggettive ed incomplete.

- **Features di basso livello:** features che, in genere, vengono estratte automaticamente e descrivono i dati e le caratteristiche di un oggetto, come le relazioni di tipo spazio/temporali. Ad esempio, per l'audio prendiamo in considerazione volume e frequenze, per le immagini la distribuzione del colore, la forma degli oggetti; per il video, ad esempio, la struttura temporale, le caratteristiche dei frame chiave, ecc…

- **Features di alto livello:** vengono generalmente estratte tramite l'intervento dell'uomo o di algoritmi di AI già istruiti; tali features, descrivono il contenuto di un media, analizzandolo (ad esempio se il media contiene musica, quali parole chiave contiene, chi è un determinato soggetto in primo piano, ecc…).

Un sistema di gestione di dati multimediali deve supportare tutti e quattro i livelli di features, che si completano a vicenda e rendono la descrizione di un oggetto più completa. Un recupero basato, ad esempio, sugli ultimi due tipi di features è detto "basato sul contesto".

6.6 Indicizzazione dei dati

Dopo aver estratto le feature e gli attributi di un dato come dovuto, si passa all'indicizzazione, per organizzare la memorizzazione del dato e "catalogarlo", in relazione ancora alle feature e agli attributi estratti; data la loro natura differente, tuttavia, servono delle strategie di indicizzazione adeguate. L'indicizzazione può avvenire su più livelli e prende in considerazioni tutte le specifiche dell'oggetto.

Il retrieval multimediale è basato sulla similarità, e non su un matching di query esatto; tale similarità, dunque, è calcolata in base agli attributi e features estratte ed è espressa come uno o più valori.

I processi specifici di indicizzazione saranno trattati nelle sezioni successive.

6.7 Quality of Service (QoS)

I sistemi MIRS sono tipicamente dei **sistemi distribuiti**, in modo da garantirne l'utilizzo anche in caso di problemi ad un server e prestazioni migliori in diverse regioni e stati distanti. Il QoS specifica un insieme di parametri essenziali di qualità, divisi in *qualità preferibile* e *qualità accettabile* (minimo indispensabile).
Il QoS gestisce i parametri in uno dei seguenti modi:

- **Deterministico:** qualità richiesta garantita pienamente.

- **Statistico:** qualità richiesta garantita con una certa probabilità.
- **Best-Effort:** qualità non garantita, si cerca di fornire il meglio che si può.

Tali caratteristiche di qualità gestite dal QoS, intercorrono nella relazione tra *Server* (fornitore) e *client* (richiedente); il client richiederà sempre il massimo possibile che sia in grado di gestire, mentre il server cercherà di fornire al singolo utente il minimo necessario, in modo da gestire più client possibili contemporaneamente.

6.8 Multimedia Data Compression

Quando i dati vengono inseriti in un DB (in particolare parlando di media/multimedia come audio, immagini e video), vengono salvati dopo esser stati compressi; per estrarne le caratteristiche, bisogna dunque prima decomprimerli, mediante un processo molto frequente che, però, richiede un'elevata capacità computazionale.

Esistono tre metodi di approccio efficienti, quali:

- **Metodo 1:** sul server, per ogni grande immagine, si salva anche una copia ridotta; la query dell'utente recupera la copia ridotta e, se occorre maggior dettaglio, viene recuperata anche l'immagine originale. Lo svantaggio consiste nell'avere una ridondanza di dati sul server.

- **Metodo 2:** ogni query recupera unicamente il dato originale, che viene poi ridotto per poter essere registrato dal client; se è necessario maggior dettaglio, il server trasmette allora il dato originale: Lo svantaggio è uno spreco di banda per la trasmissione.

- **Metodo 3:** Vengono utilizzati metodi di decompressione scalabili, progressivi e gerarchici (come GIF e JPEG progressive).

Questi metodi non alterano il dato, rispettando i margini di tollerabilità.

6.9 Differenza tra IR e DBMS

In un **DBMS** la struttura è omogenea e i componenti (record) lo sono altrettanto, ed ognuno di essi è definito completamente ed univocamente dai propri attributi.
Il retrieval (recupero) delle informazioni da un DBMS avviane tramite *match esatto*.

In un **IR** (Indexing and Retrieval), invece, i record NON sono strutturati; ogni oggetto è indicizzato tramite l'utilizzo di keywords, descrittori e indici *soggettivi* e *approssimativi*. Il recupero NON avviene mediante match esatto, ma approssimativo e parziale (come spiegato nel corso di questa sezione - MIRS).

Come visto in precedenza (in particolare nella *sezione 1.4*) vi è una fase di preprocessing e una di processing del dato; in una fase (**offline**), il dato viene caricato e indicizzato, ne vengono comprese

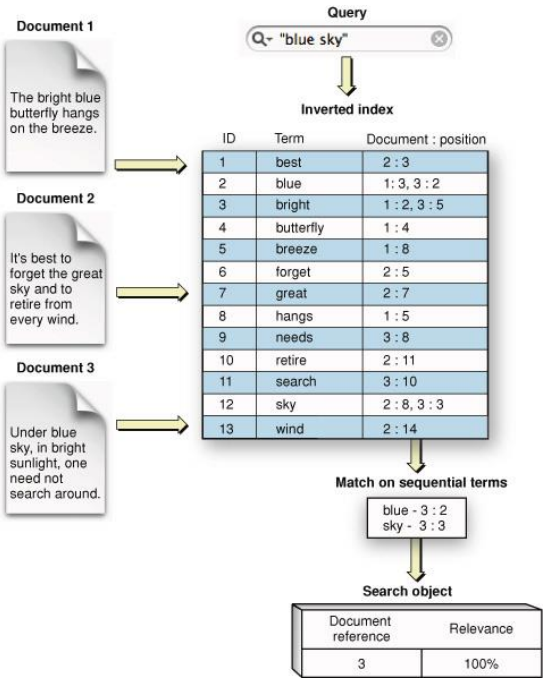
le caratteristiche, per poi caricare una sua versione sul server. In un'altra fase (**online**), la query scritta viene processata e rappresentata in una determinata forma. Vi è dunque una fase di **confronto**, in cui la query dell'utente viene confrontata con varie rappresentazioni di oggetti noti, alla ricerca di quelli idonei per il risultato. Tutto questo, avviene attraverso l'IR.

7. Indexing & Retrieval: Testo

Vediamo come avviene, all'interno di appostiti Database, l'indicizzazione e il recupero di dati in formato testuale. Una tipologia di ricerca di dato è la **flat-file**, che consiste sostanzialmente nel ricercare il nostro dato testuale scorrendo l'intero documento (file di testo); questa tipologia è molto poco utilizzata a causa della maggiore inefficienza. Esistono metodi di indicizzazione e recupero automatici o specifici, che analizzeremo singolarmente.

7.1 Inverted File

Per ricercare un dato all'interno di un documento testuale (come una determinata parola), possiamo utilizzare l'inverted file, il quale contiene un insieme di righe di testo (ognuna indicizzata con un determinato ID), ognuna delle quali contenente il **termine** che vogliamo ricercare e un **puntatore a documenti** record che contengono quel termine. Definito inverted in quanto spiega come la ricerca avviene in maniera inversa: prima la chiave da ricercare, e poi il documento che la contiene. Ad esempio, come riportato nell'immagine, le stringhe "butterfly" e "breeze" sono contenute solo nel documento 1



Rispetto alla ricerca su flat file è molto più efficiente, in quanto non dobbiamo scorrere ogni volta l'intero file per trovare la stringa desiderata.

Come possiamo vedere dall'esempio riportato, è anche possibile **raffinare** la ricerca, riportando altre componenti importanti come la frequenza del dato, la sua posizione (indicata nella colonna di sinistra, dopo il puntatore al documento), un grado di rilevanza totale con la nostra ricerca, ecc...

Definiamo alcune fasi di filtraggio del testo al fine di indicizzarlo:

- **Stop Words:** vengono esclusi elementi "insignificanti", come articoli, congiunzioni, punteggiature, pronomi, ecc…

- **Stemming:** se nel testo compaiono più volte parole analoghe (ad esempio delle derivate come "pescare", "pescatore", "pescato"), allora si considera solo il prefisso comune (in questo caso, "pesc"); tuttavia, ne lede il significato.

- **Thesaurus:** usando un vocabolario apposito, c'è la possibilità di sostituire tutti i termini simili e sinonimi, con un termine unico che compare nel testo (come, ad esempio, se sono presenti "lavare", "pulire", "smacchiare", potremmo considerare unicamente il termine "lavare").

- **Weighting:** siccome non tutti i termini che compaiono nel documento hanno la stessa valenza, la loro importanza può essere valutata in base al significato, frequenza ed ordine nelle frasi, mediante la formula: $W_{ij} = tf_{ij} * log\ (N / Df_j)$

  (dove W_{ij} è il peso del termine *j* nel documento *i*, tf_{ij} è la frequenza del termine *j* nel documento *i*, *N* corrisponde al numero di documenti totali nel database e Df_j è il numero di documenti nel database che contengono il termine *j*.)

Introduciamo inoltre due operatori che è possibile utilizzare per gestire i termini da ricercare, quali **WITHIN SENTENCE** (due termini devono essere presenti nello stesso paragrafo) e **ADJACENT** (due termini devono essere adiacenti). Le query, saranno dunque elaborate mettendo insieme i vari operatori messi a disposizione.

7.2 Retrieval con modello spazio-vettoriale

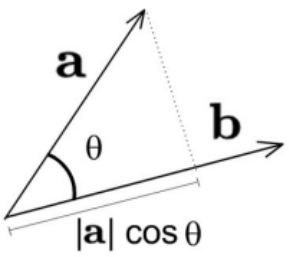
Definiamo *A* come un vettore in uno spazio vettoriale, con componenti a_1, a_2, …, a_n.

Il vettore *A* rappresenta un file, e le componenti che lo descrivono sono le sue caratteristiche. Supponiamo ora di voler estrarre dalla query dei dettagli, che andranno a formare un vettore *B* con componenti b_1, b_2, …, b_n.

A questo punto, la similitudine tra un file e una query è l'**angolo formato dai due vettori**; più l'angolo è piccolo, maggiore sarà la somiglianza. Il coseno dell'angolo è:

- 0: se l'angolo è di 90°
  file e query non si somigliano

- 1: se l'angolo è di 0°
  file e query massima similitudine



$$\mathbf{a} = [a_1, a_2, \ldots, a_n]$$

$$\mathbf{b} = [b_1, b_2, \ldots, b_n]$$

$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2 + \ldots.}$$

7.3 Relevance Feedback

L'idea di questo modello è quella di sfruttare le **valutazioni** degli utenti per raffinare le ricerche future; i termini che si sono rivelati meno rilevanti per gli utenti vengono diminuiti di peso o scartati, mentre query con termini molto ricercati saranno corrette per fornire un risultato migliore.

7.4 Modello probabilistico e Cluster

Esistono altri modelli da utilizzare per il retrieval; analizziamone due in particolare.

Modello probabilistico
Questo modello considera la dipendenza dei vari termini con le loro principali relazioni, quindi calcola la probabilità che, ad esempio, dopo un dato termine se ne presenti un altro specifico; ha avuto scarso successo a causa delle difficoltà nel calcolo delle probabilità.

Modello basato su Cluster
Tale modello si basa sull'idea di definire un insieme di gruppi contenenti al loro interno elementi simili (**cluster**); la query, cercherà proprio all'interno del cluster.

La generazione del cluster può avvenire in due modi:

- **Similarità per coppie:** ogni documento viene rappresentato come un vettore; viene calcolata la similitudine per ogni coppia di documenti e si popola una matrice con le varie distanza. Inizialmente, ogni documento viene posizionato in un cluster (avremo quindi cluster con un singolo documento), poi la coppia meno distante (quindi la più simile), dopo esser stata rimossa dalla matrice delle distanze, sarà sostituita da un oggetto unico, messo in un cluster. Si avanza via via confrontando le distanze del nuovo cluster con quelli già presenti finché nella matrice delle distanze non rimarrà un solo elemento.

- **Clustering euristico:** viene considerato il primo documento, che formerà il primo cluster; successivamente, viene confrontato un altro documento con i cluster generati e collocato nel cluster "più vicino" (se la distanza è eccessiva si forma un nuovo cluster). Si ripete il processo fino ad esaurire i documenti.

Ogni cluster è caratterizzato da un *centroide* (media dei vettori nel cluster); il processo di **retrieval** confronta la query con i centroidi dei cluster, selezionando il più simile (restituendo tutti i documenti del cluster o quello con maggior similitudine).

7.5 Misurazione delle prestazioni
Considerata la quantità immensa di informazioni e dati esistenti, restituirli tutti come risultato di una query (ad esempio, dal web), sarebbe impossibile; pertanto, si cerca di restituire meno elementi per una determinata query, ma con il maggior dettaglio possibile. Le prestazioni di un motore di ricerca si basano principalmente su:

- **Velocità di ricerca:** tempo che passa dalla richiesta (da Client a Server) alla risposta (da Server a Client).

- **Recall:** capacità di recuperare solo le informazioni rilevanti per una query; è il rapporto tra il numero degli elementi rilevanti recuperati e il numero totale di elementi rilevanti presenti nel database.

- **Precisione:** misura l'accuratezza degli elementi recuperati; si definisce come il rapporto tra il numero di elementi rilevanti e il numero totale di documenti recuperati per una determinata query.

Maggiore è il recall, minore sarà la precisione e viceversa (preferibile alta precisione)

$$recall = \frac{|doc\_ret \cap doc\_rel|}{doc\_rel}$$

$$precisione = \frac{|doc\_ret \cap doc\_rel|}{|doc\_ret|}$$

doc_ret = documenti recuperati
doc_rel = documenti rilevanti

7.6 Crawler

Attraverso il web possiamo accedere ad una quantità immensa di istruzioni, tramite i vari motori di ricerca; esistono molteplici tipi di comunicazione tra Client e Server.

Ogni documento inserito in rete, è identificato da un indirizzo univoco (URL); a gestire tutti questi documenti, vi sono i **Crawler** (noti anche *spider*), ovvero dei bot che analizzano il contenuto di una rete in modo del tutto automatizzato, per inserire tali documenti all'interno di un indice; si basano su una lista di URL da visitare fornita proprio dal motore di ricerca e, durante l'analisi di un URL, vengono identificati tutti i link contenuti nello stesso documento e aggiunti alla lista di URL da visitare.

In sostanza, il motore di ricerca seleziona diverse liste di URL da affidare ai vari Crawler, i quali li visitano e indicizzano così i vari documenti caricati in rete (ricordiamo che ad ogni URL è associato un documento).

8. Indexing & Retrieval: Audio

Siccome per l'essere umano è facile distinguere delle tracce audio secondo numerosi fattori, la cosa migliore sarebbe quella di riprodurre "la stessa capacità" su un database, fornendone quindi anche una certa *soggettività*. Bisogna innanzitutto effettuare un primo livello di distinzione tra **parlato**, **musica** e **rumori**; tale distinzione è importante in quanto diversi tipi di audio richiedono diverse tecniche di recupero e indicizzazione, oltre al fatto che assumono tutti diverso significato.

I segnali audio vengono rappresentati nel **dominio temporale** (tempo/ampiezza) e nel **dominio delle frequenze** (frequenza/magnitudine). Ogni rappresentazione è idonea all'estrazione di determinate caratteristiche.

8.1 Dominio Temporale

La rappresentazione in *Time Domain* è molto intuitiva: possiamo rappresentare il silenzio dallo 0, mentre i valori del segnale possono essere positivi o negativi in base alla relazione tra la pressione d'aria generata dall'onda sonora e quella atmosferica in condizioni di silenzio; la rappresentazione, avviene mediante un sistema a 16 bit.

Vediamo alcune caratteristiche derivate dal *time domain*:

- **Average Energy** (energia media)**:** indica la rumorosità del segnale audio, calcolando mettendo in relazione la somma di tutti i campioni al quadrato (in modo da eliminare valori negativi), e il numero totale di campioni.

- **Zero Crossing Rate:** indica con quale frequenza l'ampiezza del segnale cambia di segno, calcolabile mediante complesse relazioni.

- **Silence Ratio:** corrisponde alla quantità di silenzio in una traccia audio (soprattutto in quelle musicali); è il periodo entro il quale i valori di un certo numero di campioni, per un determinato arco di tempo, siano vicini ad una data soglia di ampiezza (molto vicina a 0).

8.2 Dominio delle frequenze

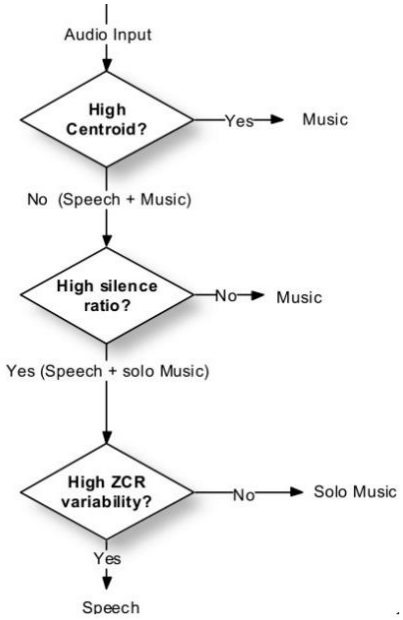
Un segnale può essere scomposto in frequenze (che a loro volta lo compongono), applicando al segnale dal dominio temporale la *trasformata di Fourier*. Il dominio delle frequenze mostra come è distribuita l'energia alle varie frequenze.

Tale rappresentazione, è detta **spettro del segnale**. Dalla trasformata di Fourier derivano diverse caratteristiche:

- **Bandwidth:** range delle frequenze del suono; tipicamente la musica ha un range più ampio del parlato, quindi frequenze inferiori a 7 kHz indicano che il file contiene del parlato (il range è calcolato effettuando una differenza tra la massima e la minima frequenza). Il modo più semplice però per valutare la distribuzione delle frequenze, è osservando lo spettro del segnale.
  Il *centroide spettrale* è il punto medio della distribuzione di energia del suono.

- **Armoniche:** un suono prodotto da un corpo vibrante non è mai puro; gli armonici sono un insieme di suoni più/meno acuti e più/meno intensi, che si accavallano al suono principale (detta *frequenza fondamentale*). Sono essenziali per determinare il timbro di uno strumento e gli intervalli musicali (in genere la musica contiene molte più armoniche di un semplice suono). Per testare se un suono contiene armoniche, si controlla che le frequenze dei componenti dominanti siano multiple di una frequenza fondamentale.

8.3 Metodi di classificazione dell'audio



Le principali fonti di analisi dei segnali audio sono il parlato e la musica; ovviamente, sarebbe possibile fare una classificazione molto più raffinata (aggiungendo varie sotto-categorie). I principali metodi di classificazione audio sono:

- **Step-By-Step:** viene prima valutata la posizione del centroide e, se il valore supera una certa soglia, allora si presume che si tratta di un brano musicale. Successivamente si valuta il *Silence ratio*, tipicamente un basso silenzio conferma che si tratta di un file audio. Infine, si valuta lo *Zero crossing rate*. Se la traccia "passa" tutti i controlli, si tratta di parlato.

- **Su caratteristiche Vettoriali:** i valori di varie caratteristiche della traccia considerata costituiscono le componenti di un vettore (V) che sarà *confrontato* con altri vettori di caratteristiche R_i, che rappresentano i vettori di riferimenti di varie tipologie di audio.

8.4 Riconoscimento del parlato e ASR

Una volta identificato il parlato, viene utilizzato l'**Automatic Speech Recognition** (ASR), che prende la traccia audio e ne genera un testo; ad occuparsi di estrarre e indicizzare i contenuti, se ne occuperà poi l'IR (in quanto ora abbiamo un file testo).

Al sistema dell'ASR vengono fornite componenti di base del parlato (fonemi, parole, frasi, ecc…), e viene istruito mediante campioni audio a riconoscere determinate parole e il timbro con cui vengono pronunciate (*training*). Viene tutto suddiviso in singole unità, confrontate con vettori di features (bandwidth, silence ratio, ecc…).

Purtroppo, vi sono diverse problematiche in quanto molti fonemi, in base alla persona da cui sono prodotti, hanno grandi differenze, oppure vi è una forte presenza di rumore di fondo o ambientale. Il processo di riconoscimento è infatti *statistico*, migliorabile fornendo una maggiore conoscenza del linguaggio.

8.5 Altre tecniche

Il **Dynamic Type Warping (DTW)**, cerca di normalizzare la durata della sequenza audio parlata, così da poterla confrontare con vettori memorizzati in *fase di training*.
Il principale obbiettivo, è quindi quello di avvicinare il più possibile due elementi (in questo caso tracce audio), in modo da confrontarle in maniera accurata.

La **Hidden Markov Models (HMM)** è una tecnica molto usata sia per il riconoscimento del testo che dell'audio (parlato); è costituito da stati, probabilità do transizione e probabilità di generazione dei simboli. La sua costruzione avviene durante la fase di training.

Le **Artificial Neural Networks (ANN)** sono impiegate per il riconoscimento e simulano i processi cognitivi dell'uomo, in particolare la fase di apprendimento, replicando una rete interconnessa da vari link. Vengono utilizzate implementazioni che prevedono l'uso di molti elementi di calcolo detti *Processing Elements* (PE), interconnessi tra loro come neuroni; tali connessioni rappresentano le **sinapsi**.

Ogni PE della rete effettua una **integrazione** (somma pesata) degli input derivati dalle altre sinapsi. L'input passato viene successivamente valutato da una funzione detta **trasformazione** per determinare l'output del singolo PE.

9. Indexing & Retrieval: Immagini

L'indicizzazione e il recupero delle immagini costituisce un settore molto sensibile nel campo dell'informazione, in quanto i risultati ottenuti dalle ricerche sono implementati in molteplici scopi, dai fini più banali a quelli legali; mediamente, i risultati sono impiegati in database di tipo commerciale.

Possiamo seguire diversi approcci per l'indicizzazione e la ricerca, come quelli **tradizionali,** ovvero attraverso l'uso di metadati (nome del file, categoria, ecc…), ma gli attributi possono non descrivere in maniera completa gli oggetti. Un altro approccio è quello **basato sui contenuti**, come per le caratteristiche a basso livello (colori, texture, forma).

Ancora, quelle basate sul **riconoscimento degli oggetti** sono molto valide e sfruttano complessi algoritmi di riconoscimento ed estrazione di features degli oggetti presenti nella scena, tuttavia comprendo molti svantaggi (come la "pesantezza" computazionale impiegata, l'utilizzo di algoritmi estremamente sofisticati...)

9.1 Annotazioni libere (text-based retrieval)

Le immagini vengono descritte da testo libero non controllato; è quindi una tecnica che offre un enorme vantaggio, quale quello di descrivere ed indicizzare concetti di altissimo livello (ad esempio "persona che sorride con un braccio alzato"), ma essendo una tecnica manuale, comporta numerosi aspetti negativi, come la soggettività della descrizione, l'incompletezza/inconsistenza di quest'ultima, e la necessità di un dizionario enorme per riconoscere i vari termini utilizzati.

Le query sono sotto forma di parole chiave concatenate con operatori booleani, confrontate, tramite IR, con il testo descrittivo associato all'immagine.

9.2 Confronto degli istogrammi di colore

Ogni immagine è memorizzata assegnando ai vari pixel tre valori corrispondenti a quelli dello standard RGB, che vanno da 0 a 255; ogni canale è diviso in *m* intervalli di colore (*quantizzazione dei colori*), e il numero di combinazioni diverse è definito **bin**.

Definiamo **istogramma di colore** il vettore $H(M) = (h_1, h_2, \ldots, h_j, \ldots, h_n)$ dove h_1 rappresenta il numero di pixel dell'immagine *M* che appartengono al bin *j.*

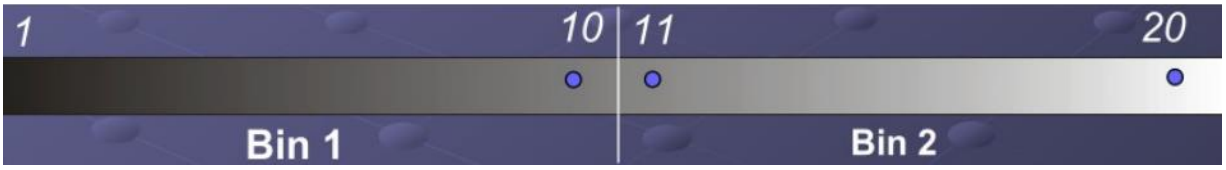
Per indicizzare un'immagini con questo metodo, dunque, si calcola l'istogramma di colore H(M) che verrà utilizzato come indice di quell'immagine. Successivamente, per effettuare la ricerca di immagini simili ad una fornita da query, basta definire una distanza tra i due istogrammi; minore è la distanza, più le due immagini sono simili.
Date due immagini A e B, la loro distanza è:

$$d(A,B) = \sum_{i=1}^{n} |a_i - b_i|$$

dove a_i e b_i: numero di pixel dell'immagine A e B che rientrano nell'i-esimo bin.

Tale metodo è soggetto a numerose problematiche e può fornire risultati incorretti se non si applicano le dovute varianti, necessarie per trattare meglio l'informazione.
Tra le problematiche possibili, analizziamo la seguente:

Innanzitutto, immaginiamo i vari bin come delle "partizioni" del canale; dunque, nonostante possono essere adiacenti, saranno considerati completamente diversi



Così come per i bin, anche per i punti che gli appartengono; secondo questo esempio, il punto 11 è più simile a 20 (perché è nello stesso bin) che a 10,

Per ovviare a questo problema, sono possibili tre soluzioni alternative:

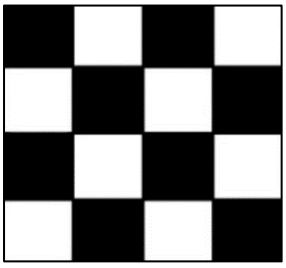
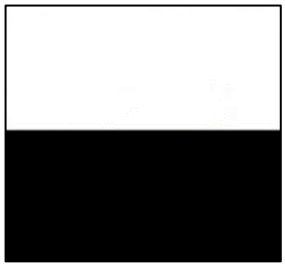
- **Soluzione 1) Distanza tra i bins**
  La differenza tra i bis si definisce come similarità calcolata "bin a bin"; così facendo, abbiamo una misurazione decisamente più precisa.

- **Soluzione 2) Istogramma cumulativo**
  Non considera la distanza tra i bin ma crea delle classi cumulative per aggirare il problema; l'istogramma cumulativo è ottenuto associando a ogni classe cumulativa un valore dato dalla somma dei valori della stessa classe e di quelle che la precedono, in modo da avere un risultato "più spalmato".

- **Soluzione 3) PWH**
  Il Perceptually Weighted Histogram si basa nel cambiare lo spazio di rappresentazione dei colori da RGB a CIELuv, in modo da associare ad ogni valore RGB circa 10 corrispondenze in CIELuv, fornendo un nuovo istogramma più dettagliato.

9.3 Limiti dell'approccio color-based

I sistemi di indicizzazioni basati sul colore, hanno il grave problema di ignorare le relazioni spaziali del contenuto, limitandosi a valutarne il colore; nonostante due immagini siano totalmente diverse, se presentano numericamente gli stessi pixel (con ovviamente gli stessi colori), saranno considerate uguali, come le seguenti:

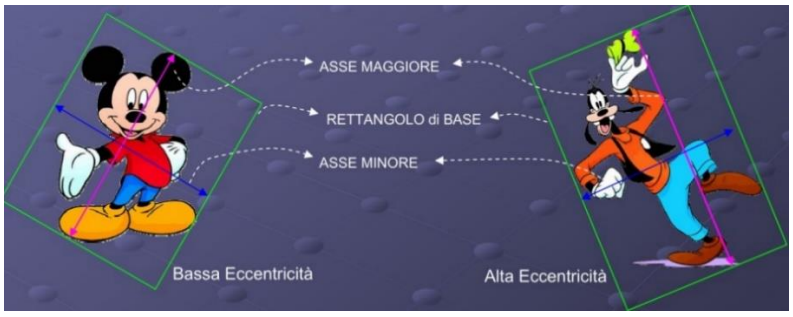
Le due immagini,   nonostante siano disposti diversamente, presentano lo stesso numero di pixel bianchi e neri, pertanto, sono considerate uguali.

Un altro problema è la mancata distinzione di soggetti, background e foreground;
Sono considerate più simili tra di loro due auto totalmente diverse, sia in forma che in colore, ma che si trovano sullo stesso colore di sfondo (quindi si presume che i pixel del background coprano la maggior parte dell'immagine), rispetto a due auto identiche, ma su background di colori diversi. Bisognerebbe segmentare l'immagine, fornendo una distinzione tra soggetti e sfondi, considerandoli separatamente.

9.4 Indicizzazione e ricerca sulla forma

L'immagine viene segmentata in oggetti che la definiscono; bisogna riconoscere e identificare i soggetti anche in seguito a **traslazioni**, **rotazioni** e modifiche **di scala**.

Bisogna confrontare tutti gli elementi che compongono il soggetto, quali un **rettangolo di base**, ovvero il più piccolo rettangolo capace di contenere completamente la figura; un'**asse maggiore**, segmento che congiunge i due punti del soggetto più distanti tra loro; un'**asse minore**, una linea *perpendicolare* all'asse maggiore, e che congiunge i due lati paralleli del rettangolo di base.
L'eccentricità è il rapporto tra la lunghezza dell'asse maggiore e la lunghezza dell'asse minore.



Durante i confronti da una query, immagini con eccentricità diversa verranno scartate a prescindere (pur traslando, ruotando o effettuando modifiche in scala a un'immagine, la sua eccentricità non cambia). Tipicamente, quando un'immagine viene ricercata, si tende ad effettuare una normalizzazione in scala, in modo da confrontarle tutte avendo lo stesso asse maggiore.

Il processo è ben preciso: viene fissata la **CellSize** (dimensione in pixel), si presume che l'asse maggiore sia univoco; vengono effettuate le ulteriori operazioni di rotazione/scala e, infine, per ogni forma si conosce la sua eccentricità.

Il **calcolo della similarità** viene compiuto dopo aver "regolarizzato" le varie immagini; dopo aver determinato le eccentricità (come visto precedentemente) se queste ultime sono simili tra loro, allora sovrapponiamo a ogni immagine una griglia regolare alla forma: si assegna 1 nei pixel in cui è presente la figura, e 0 laddove è presente il background; otteniamo così delle stringhe binarie, che costituiranno l'indice e, insieme al numero di celle occupate, saranno un criterio di confronto per identificare e recuperare le immagini.

9.5 Indicizzazione su Texture

La texture descrive una "percezione" dell'immagine, dividendola in diverse aree caratterizzate da una "trama" comune; possiamo identificare una texture in base a diversi elementi della sua struttura, quali **granularità**, **contrasto**, **direzione dominante**, **regolarità**, ecc…

Tutte queste caratteristiche, costituiranno il "motivo" dominante delle varie aree in cui è possibile suddividere l'immagine, nonché fonte di criterio di indicizzazione.

10. Indexing & Retrieval: Video

Un file di tipo video è un flusso di dati ricco di informazioni, che includono spesso testo (come i sottotitoli), audio parlato e/o musica, immagini (frame riprodotti a velocità costante) e metadati.

Oltre a caratterizzare il file, questi dati sono anche il principale riferimento per l'indexing e il retrieval dei video, in quanto gestiamo le informazioni dei metadati all'interno di RDBMS tradizionali, il testo utilizzando IR standard, l'audio secondo le tecniche precedentemente descritte (in particolare segmentando il parlato e il non parlato tramite *speech recognition* per ricavare il testo su cui applicare l'IR).

Un altro importante aspetto di indicizzazione è quello basato sul contenuto; si distingue in due metodi, entrambi basati sull'analisi dei fotogrammi:

- **Singolo fotogramma:** il video viene considerato come una serie indipendente di immagini indipendenti sulle quali è possibile applicare tecniche (appunto, per immagini). Questo metodo, oltre a causare la perdita delle informazioni temporali, richiede una mole di dati da elaborare molto grande.

- **Gruppi di fotogrammi:** i frame vengono suddivisi in *shots* (gruppi) e l'indicizzazione è basata sul frame più rappresentativo di ogni shot.

10.1 Indexing basato su Shots

Per identificare shots, occorre considerare le condizioni di colore, sfondo e soggetti rappresentati; generalmente frame consecutivi non hanno cambi radicali nelle caratteristiche appena citate, ma quando ciò avviene, vuol dire che i nuovi frame appartengono ad uno shot diverso (basta immaginarsi gli shots come "scene" separate; appena vi è un cambio drastico in colore, sfondo e/o soggetti, allora siamo in uno shot diverso). Uno shot è quindi a tutti gli effetti considerabile una scena.

Per individuare gli shots, dunque, occorre definire una misura quantitativa che catturi le differenze tra coppie di frame: definiamo una certa soglia, se la differenza tra un frame e il suo successivo supera quel valore, allora quel punto è l'interruzione tra due shots differenti. La suddivisione in shots può essere effettuata in due modi:

- **Metodo 1)** si calcola la somma *pixel per pixel* delle differenze tra due frame consecutivi; metodo con scarsi risultati, in quanto, anche se la scena è la stessa, un singolo oggetto in movimento causerebbe un cambiamento radicale nella valutazione dei pixel.
- **Metodo 2)** si calcola la differenza tra gli *istogrammi di colore* di due frame consecutivi; così facendo, il movimento di un oggetto non altera esageratamente l'istogramma di colore tra i frame.

Definiamo inoltre la distanza tra frame *SD* come: $SD_i = \sum_i | H_i(j) - H_i(j) |$
dove $H_i(j)$ è il valore del j-esimo bin dell'istogramma dell'i-esimo frame.

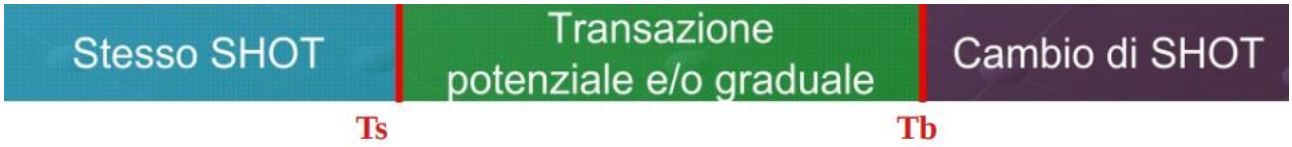
Se SD_i supera la soglia prefissata, allora si verifica un cambio di shot; per una buona **scelta della soglia** per individuare gli shots è consigliato scegliere un valore leggermente più alto della media di tutte le differenze tra un frame e il successivo.

10.2 Segmentazione a due soglie

Quanto espresso fino ad ora ha efficacia se, tra un frame e un altro, ad indicare il cambio di shot vi è un cambio di scena tramite "taglio netto"; se la scena cambia lentamente, con effetti di sfumatura (fade-in / fade-out), ciò risulta un problema, in quanto un cambiamento graduale non viene rilevato per indicare un nuovo shot.

Utilizziamo dunque tecniche di **segmentazione a due soglie** per ovviare il problema; letteralmente, invece di scegliere un'unica soglia (come nei casi precedenti) come metro di paragone tra frame per determinare il cambio di shot, ne utilizzeremo due.

La soglia **Tb** viene utilizzata per i cambi di camera, mentre una soglia più bassa **Ts** per determinare i frame nei quali avviene una transizione graduale.



Se la differenza è maggiore di *Tb*, abbiamo un cambio di shot (mediante taglio netto), mentre se il valore è compreso tra *Ts* e *Tb* abbiamo un *frame di transizione potenziale*; se ci sono più frame di transizione potenziale di fila, la loro distanza viene sommata, e quando si raggiunge la soglia *Tb* allora si genera un cambio di shot.

10.3 Panning/Zoom e Cambiamenti di Illuminazione

Oltre al problema delle transizioni prima citato, a disturbare gli algoritmi di controllo shots, vi sopraggiungono altri problemi:

Problema 1 – Panning e Zoom
Queste due operazioni possono generare dei cambiamenti graduali (come nel caso precedente); per limitare questo problema, si cerca di identificare il movimento di blocchi di pixel nei frame. I vettori di movimento hanno un preciso comportamento: mentre nel Panning vanno tutti verso una stessa direzione (traslano), nello Zoom convergono o divergono dal punto centrale dell'immagine.

Problema 2 – Cambiamenti di illuminazione
Cambi di illuminazione repentini e drastici (come il flash), generano un'alterazione nell'istogramma di colore dell'immagine, portando l'algoritmo a riconoscere uno shot diverso; per ovviare questo problema, la scelta migliore (nonché la più semplice), è quella di *normalizzare i colori* nel frame (in alternativa, è possibile lasciar perdere i colori e passare ad un'analisi del soggetto).

10.4 Utilizzo degli Shot e Frame Rappresentativo

Una volta identificati gli shots che compongono il file video da analizzare, bisogna rappresentarli ed indicizzarli, così da gestirli meglio in fase di ricerca. Per ogni shot, dobbiamo trovare uno o più **frame rappresentativi (R-Frame)**, utilizzati per indicizzazione e ricerca, sfruttando le tecniche già conosciute per le immagini.

Quantitativamente, scegliere **1 R-Frame** per gli shots, potrebbe non catturare completamente il contenuto, e non consideriamo correttamente le differenze di lunghezza tra shots. Scegliere **N R-Frame** potrebbe generare R-Frames sovrabbondanti nel caso di shot più uniformi, oltre al fatto che gestisco la lunghezza e non il contenuto (potrebbe essere troppo variegato e quindi poco rappresentativo)

La scelta migliore è la suddivisione in **Sotto-Shot**, quindi utilizzando le stesse tecniche per individuare gli shots, andiamo a creare "gli shot degli shot", generando poi un R-Frame per ogni sotto-shot.

Ma *come individuare il frame rappresentativo (R-Frame)?*
Nel caso di shot abbastanza statico, è indifferente quale R-Frame scegliere: qualunque frame può essere preso in considerazione come rappresentativo; nel caso invece di shots più movimentati, la scelta dell'R-Frame è fondamentale, e possiamo:

- Considerare come R-Frame il **primo frame** dello shot (o del sotto-shot)

- Fare la **media dei colori** di tutti i frame dello shot (o del sotto-shot), pixel per pixel; il frame con i colori più vicini a tale media sarà l'R-Frame scelto.

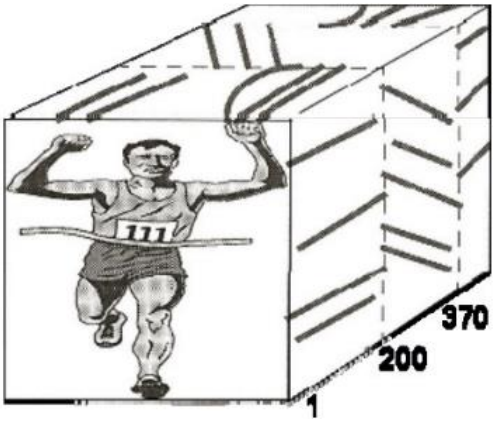
- Calcolare la media di tutti gli istogrammi dei frame di shot (o sotto-shot) e scegliere come R-Frame quello che ha l'istogramma più vicino a quello medio.

10.5 Motion Vectors e Riconoscimento oggetti

Nel caso di shots molto movimentati (oppure in cui ci sono degli spostamenti dell'inquadratura come per Zoom e Panning), possiamo indicizzarli anche mediante le informazioni ricavate dai ***motion vectors***, ovvero vettori che "rappresentano" il movimento (che sia di un oggetto, oppure di "gruppi di pixel").

Così come visto per l'*analisi della forma*, è possibile **identificare oggetti** mediante varie caratteristiche; contrariamente a quanto si possa immaginare, ciò rende ancora meglio nel caso di video, rispetto ad immagini statiche, in quanto il movimento fornisce informazioni aggiuntive sull'oggetto in analisi.

10.6 Rappresentazione MICON



Un modo per rappresentare in maniera compatta gli shots è tramite **Motion Icon** (MICON); questa rappresentazione, utilizza diversi componenti per dare un'idea molto più completa dello shot che si vuole rappresentare.
Innanzitutto, immaginiamolo come un cubo: sulla facciata anteriore, l'R-Frame, la profondità rappresenta invece la durata temporale, mentre i pixel lungo i bordi orizzontali e verticali (le altre facciate del cubo) danno l'idea del movimento all'interno dello shot. Questo cubo non è altro che un "pacchetto" compatto dei vari frame che compongono lo shot (quindi, andando ad esempio a "tagliare" il cubo in un dato punto, otteniamo informazioni aggiuntive).

10.7 Browsing video e Altre rappresentazioni

Come abbiamo visto, le caratteristiche che rappresentano un video analizzate fino ad ora, ci serviranno per effettuarne l'indicizzazione; nella fase di retrieval, dunque, effettuiamo una ricerca mediante **browsing video**, che analizzerà la struttura *gerarchica* del video, seguendo questa precisa discendenza:

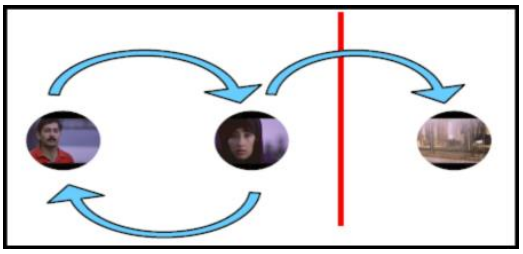
Livello video (titolo) → **Livello scene** → **Livello shots** → **Frames**

Ogni elemento di un livello, racchiude molteplici elementi del livello successivo
(un video ha molteplici scene; ogni scena ha diversi shots e, ogni shot molti frame).

Oltre alla normale indicizzazione tramite shots, possiamo trovare altre rappresentazioni compatte (ma meno performanti di quella tramite shots), come:

lo **storyboard**, una collezione di R-Frame che rappresentano le porzioni più importanti dell'intero video, la **mosaicatura**, ovvero un'unione di frame che descrivono nell'insieme qualcosa di più complesso (come un elemento del video).

Vi è anche lo **Scene Transition Graph (STG)**, una struttura che cattura in modo compatto il flusso temporale: questa struttura, utilizza un grafo orientato per unire tra loro diversi R-Frame in successione temporale.



10.8 Tipi di Query e Vettori

Le features, una volta estratte, sono rappresentate mediante un vettore multidimensionale; similarità e distanza sono calcolate come distanza tra vettori. Lo scopo fondamentale di queste tecniche consiste nel suddividere lo spazio multidimensionale in sottospazi, in modo che solo alcuni di questi debbano essere presi in considerazione per rispondere alle query. Vediamo alcuni tipi di query:

- **Point query:** questo tipo di query ricerca una risposta da oggetti il cui vettore di features corrisponda esattamente a quello della query (exact match).

- **Range query:** la query è rappresentata da un vettore e da un dato range; se il vettore di features recuperato risponde alla query rimanendo in quel range di similarità, purché non esattamente identico, è considerato valido.

- **K-Nearest Neighbor:** la query è rappresentata da un vettore e da un intero (K); saranno dunque recuperati i K oggetti il cui vettore di features ha la distanza minore dal vettore della query.

Per il recupero tramite questo sistema vettoriale, in base alle nostre esigenze, possiamo anche "spezzare" il vettore in sotto-vettori di caratteristiche, in modo da filtrare quali aspetti vogliamo prendere in considerazione per il recupero.

Esistono sistemi di filtraggio basati sul colore medio o sull'istogramma di colore.

Distinguiamo il filtraggio basato sulla **disuguaglianza triangolare**, che consiste nel calcolare la distanza tra i vari file indicizzati, salvando i risultati in una matrice; successivamente, quando eseguiamo un retrieval, sarà inutile calcolare la distanza da un dato oggetto se un suo simile è già troppo distante per la ricerca. Così facendo, si risparmiano molti confronti e velocizziamo le operazioni di recupero.

11. Strutture dati per ricerca di similarità

Come abbiamo visto, la fase di indicizzazione dei dati genera vari vettori di features, ognuno dei quali raggruppa molteplici caratteristiche di dimensioni variabili che caratterizzano il dato; eseguire, nella fase di retrival, un confronto lineare tra i vettori non è neanche minimamente pensabile, in quanto porterebbe ad un dispendio di computazione immenso, oltre ad effettuare una miriade di confronti decisamente inutili; pertanto, adottiamo l'utilizzo di strutture dati da utilizzare per la ricerca.

11.1 Alberi B

Un Albero B è una struttura dati che presenta un ordine *m* (ovvero il massimo numero di figli che un nodo può avere, ad esempio per *m = 2* è un albero binario); corrisponde ad un generico albero di ricerca che ha le seguenti proprietà:

- La **radice** ha almeno due sottoalberi, a meno che non sia foglia (non può quindi essere un albero degenere).
- Ogni **nodo interno** (quindi non radice e non foglia), contiene *k-1* chiavi e *k* riferimenti a sottoalberi, dove $\lceil m/2 \rceil \leq k \leq m$ (dove $\lceil \dots \rceil$ è il tetto).
- Le **foglie** sono allo stesso livello e contengono *k-1* chiavi, con $\lceil m/2 \rceil \leq k \leq m$

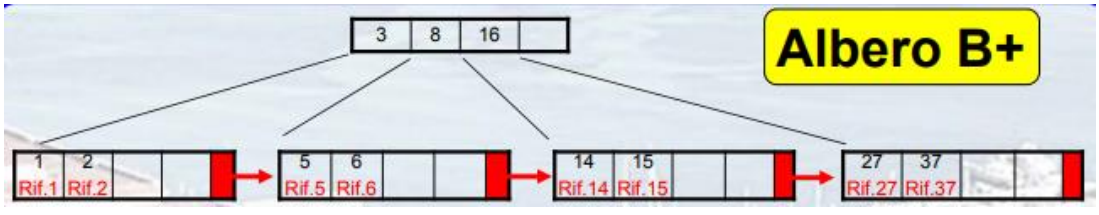
Per definizione, dunque, un Albero B è sempre pieno almeno per metà, ha pochi livelli ed è ben bilanciato; inoltre, le operazioni di ricerca e/o di navigazione dell'albero sono prevedibili. Le fondamentali operazioni effettuabili per Alberi B sono: **Creazione**, **Inserimento**, **Cancellazione**, **Ricerca**

Per l'inserimento di un valore *k*, ad esempio, se la foglia di arrivo *F* ha ancora spazio, inseriamo *k*; se *F* è piena, allora si crea un'altra foglia *G* e si spostano la metà delle chiavi di *F*, in *G.* Se la radice *R* invece è piena, allora si crea una nuova radice *S* e un nodo *M*; da questo momento, *R* ed *M* sono figli di *S.*

11.2 Alberi B+ e MB+

Un albero **B+** è considerabile come un'evoluzione dell'albero B, in quanto i riferimenti ai dati sono contenuti *solo* nelle foglie, anziché esserlo in qualsiasi nodo; le foglie di un albero B+ contengono anche un campo **puntatore** aggiuntivo per navigare le altre foglie, che ricordiamo essere tutte allo stesso livello (questa navigazione può ricordare quella di una linked list).

Vi è anche un'estensione del B+, ovvero il **B+ Multidimensionale (MB+)**, che supporta la *similarity*



query, ovvero query per intervalli e prossimità. Ogni vettore di features è un punto nello spazio 2D, e l'intero spazio è una *bounding box*, ovvero una scatola con dei limiti definiti; la bounding box è divisa in regioni contenenti i vari vettori di features simili per caratteristiche (contiene quindi i vettori, non i dati).
Una volta fatto ciò, viene costruito l'albero, rispettando ancora i criteri del B+.

11.3 Clustering

Come abbiamo già visto, è possibile raggruppare vettori di features simili tra loro in gruppi detti *cluster* in base ad un calcolo sulla similarità; se un vettore è troppo distante da un cluster, allora formerà un nuovo cluster. Ognuno, inoltre, ha un "punto centrale", detto centroide, che si ottiene mediando i vari vettori del cluster; è proprio con il centroide che andiamo a comparare il vettore di features della query, e verranno considerati solo i cluster più simili ad essa; per evitare confronti inutili e/o un dispendio di risorse eccessivo, possiamo raggruppare i cluster in un *super cluster* (quindi un gruppo di più cluster), che avrà un proprio centroide, ottenuto mediando i centroidi dei cluster al suo interno.

11.4 Alberi K-d

Gli alberi K-d sono sostanzialmente un'estensione dei normali Alberi Binari di Ricerca, in quanto ogni nodo presenta una chiave, un pointer a destra e uno a sinistra, oltre a rispettare i criteri descritti da un Albero Binario di Ricerca; tuttavia, la differenza sostanziale sta nel fatto che la key *k* non è rappresentata da un valore ma da un vettore di valori e, per generare l'albero, bisogna regolamentare l'inserimento a sinistra o destra in base al vettore: al primo livello si decide basandosi sul primo elemento del vettore, al secondo livello si decide basandosi sul secondo elemento, e così via; una volta esaurite le componenti, si riparte dalla prima del vettore.

11.5 Grid Files

Struttura dati semplice e molti intuitiva per i vari tipi di indicizzazione e ricerca; consiste sostanzialmente da una griglia multidimensionale, dove ogni cella è referenziata da un puntatore. Per andare ad inserire un vettore di features all'interno della griglia, basta andare a riferirci al puntatore che indicizza l'area predestinata a quel range di valori. Questa struttura, tuttavia, funziona bene quando i vettori nella griglia sono ben distribuiti, in quanto può capitare spesso di avere griglie quasi vuote e altra completamente sovraffollate.

11.6 Alberi R

[MISSING]

12. Watermark

I Watermark sono un insieme di strumenti e metodi per *marcare* un qualsiasi file digitale; i motivi principali per cui vengono usati i watermark, sono ad esempio la protezione con copyright di un file, riconoscerne e manifestarne il proprietario, garantirne l'originalità e la non alterazione del file stesso da parte di terzi, evitare la distribuzione non autorizzata, marcare determinate caratteristiche, oppure segnare il processo di vendita del file utilizzando marchi differenti per utente. Nella maggior parte dei casi, costituiscono (soprattutto nel caso di immagini) una sovraimpressione con delle stampe d'autore (tipicamente il nome del sito/proprietario).

Un watermark può essere **evidente**, come nell'esempio precedente, con scritte ben visibili e non cancellabili, così da evitarne l'uso non autorizzato, oppure **latente**, nascosto cioè all'interno del file, come una filigrana digitale (steganografia).

Una classificazione può essere eseguita considerando le seguenti caratteristiche:

- **Visibilità:** un watermark può essere *visibile*, rendendo note alcune informazioni all'utente, oppure *invisibile*, utilizzato in casi in cui il proprietario voglia distinguere le copie dall'originale (bisogna però "decodificare" il file).

- **Resistenza:** in base a quanto è resistente agli attacchi, può essere considerato *fragile* (facilmente attaccato, distrutto e reso irriconoscibile), *semifragile* (subisce la stessa fine di quello fragile sotto una certa soglia definita dall'autore), e *robusto* (consistente e resistente ad attacchi volti alla sua distruzione, oltre alla capacità di mantenere intatta l'informazione).

- **Autonomia:** un watermark può essere considerato *cieco*, cioè che per verificare la loro presenza non è necessario il documento originale, oppure *non cieco*, per il quale è invece necessario il documento originale (robusto).

- **Dominio:** può essere *privato* (estraibile solo se si conosce il contenuto e possiede il documento non marchiato), o *pubblico* (rilevabile anche se non si conosce il contenuto e senza documento originale, semplice da identificare e da alterare o addirittura rimuovere).

12.1 Proprietà dei Watermark

Vi sono delle caratteristiche comuni che tutti i Watermark devono rispettare, per soddisfare determinate esigenze; il legittimo proprietario o un'autorità di controllo indipendente deve poter estrarre facilmente le informazioni da un watermark, e il suo recupero non deve presentare ambiguità su chi sia l'effettivo proprietario del documento. Deve inoltre essere possibile sovrapporre più watermark, senza distruggere i precedenti e bisogna inserirli all'interno del segnale da proteggere, in modo da avere maggior sicurezza e portabilità.

Per soddisfare tutte queste richieste, un watermark apporta sicuramente un degrado del dato (soprattutto dell'immagine), data la numerosità delle informazioni richieste; è il proprietario a decidere l'impatto del degrado, ricordando che più è marcato (robusto), maggiore sarà la sicurezza. Ogni segnale di Watermark deve essere associato ad una **chiave privata** (sequenza di bit) posseduta dal proprietario del watermark; questa chiave ha la funzione di riprodurre il segnale di

watermark del documento, ma anche per riconoscerlo, ed è privata; solo chi è in possesso della chiave può dimostrare la presenza del watermark nel prodotto originale. Nel caso di watermark pubblici, ognuno è identificato da una chiave.

Inoltre, oltre a dover dare la possibilità di inserire molteplici watermark, e quindi molteplici chiavi, e garantire l'univocità di queste ultime, un watermark deve essere **invertibile**: deve dunque essere possibile, da parte del proprietario, di poterlo rimuovere; questa proprietà va in contrasto con la robustezza.

Ricordiamo inoltre che i watermark possono essere utilizzati in molteplici tipi di file, come per immagini (filigrane visibili o invisibili), video, audio (frequenze particolari), e addirittura per il testo (ad esempio, mediante l'uso di determinati spazi al termine delle righe, oppure tra le parole, in modo da costruirne un codice preciso).

13. GIS - Geographic Information System

[MISSING]

14. GPS - Global Positioning System

[MISSING]