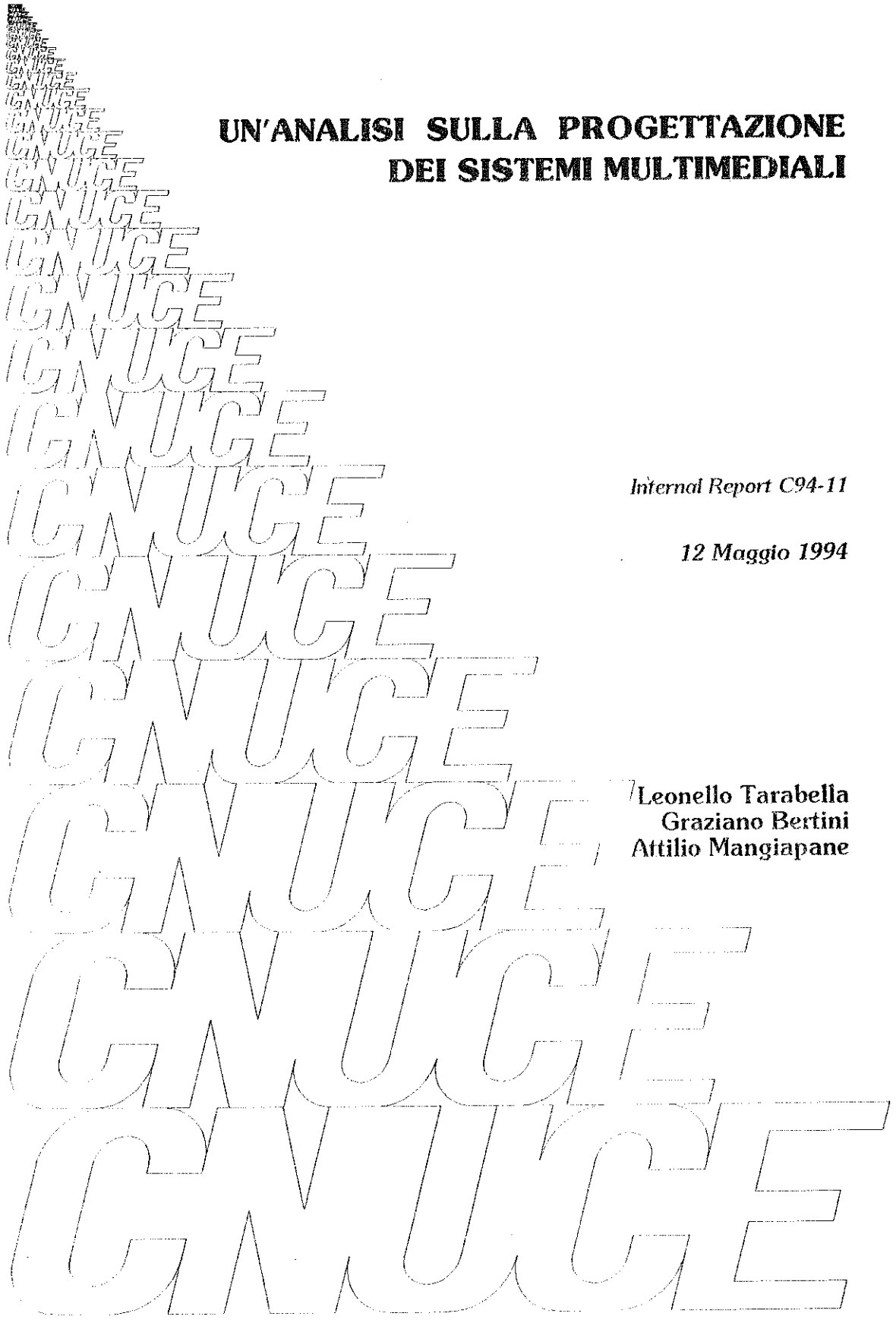
UN'ANALISI SULLA PROGETTAZIONE DEI SISTEMI MULTIMEDIALI



Internal Report C94-11

12 Maggio 1994

Leonello Tarabella
Graziano Bertini
Attilio Mangiapane

Un'analisi sulla progettazione dei sistemi multimediali

Internal Report C94-11

12 maggio 1994

Leonello Tarabella
Graziano Bertini
Attilio Mangiapane

Graziano Bertini è Ricercatore presso l'Istituto Elaborazione dell'Informazione (IEI/CNR)

Attilio Mangiapane ha svolto una tesi di laurea sull'argomento presso il Reparto di Informatica Musicale del CNUCE ed il Reparto Elaborazione Segnali ed Immagini dell'IEI.

INDICE

PRESENTAZIONE

L'obiettivo di questa relazione è quello di esaminare i diversi livelli di supporto, concorrenti alla creazione di un ambiente per lo sviluppo delle applicazioni multimediali su *workstation* che presentino un buon rapporto tra costo e prestazioni. Vengono altresì risaltate le problematiche emergenti a ciascun livello e le metodologie d'approccio ad una loro risoluzione.

A tal proposito è stato posto un accento sul problema del trattamento della sincronizzazione dei flussi di eventi media dal basso all'alto livello, allo scopo di garantire il rispetto dei vincoli e delle relazioni temporali che intercorrono tra i dati.

Eventuali approfondimenti in materia fanno riferimento alla bibliografia presentata a fine lavoro.

Il presente rapporto è stato redatto come compendio ed aggiornamento della tesi di laurea in *Scienze dell'Informazione* dal titolo: *"Un'analisi multilivello su ambienti di supporto orientati ad applicazioni multimediali"*, svolta presso il *Consiglio Nazionale delle Ricerche* di Pisa, sia nel reparto di *Elaborazione di Segnali ed Immagini* dell'*Istituto di Elaborazione dell'Informazione*, che nel reparto di *Informatica Musicale* dell'istituto *CNUCE*.

Si ringrazia il p.i. Claudio ORI del *Consiglio Nazionale delle Ricerche* di Pisa, *Istituto di Elaborazione dell'Informazione - Servizi Tecnici*, sia per l'assistenza tecnica che per la collaborazione prestata.

INTRODUZIONE

Col termine **multimediale** si intende l'applicazione integrata, interattiva e simultanea di differenti mezzi di comunicazione quali audio, video, testi, grafica, animazione su *personal computer* o *workstation*. Per le sue caratterstiche trasversali, il settore multimediale si estende ad una notevole diversità di utenti e coinvolge una larga gamma di tecniche e tecnologie. Queste ultime provengono da molti altri campi quali, ad esempio:

- il trattamento di testi;
- il suono digitale;
- le immagini digitali statiche ed in movimento.

I sistemi per la scrittura e l'elaborazione testi sono stati i primi a diffondersi in maniera massiccia in tutte le attività d'ufficio.

Il suono digitale comprende gli apparati e le tecniche per il campionamento, per la sintesi dei suoni e per la riconversione da digitale ad analogico dei segnali in banda audio.

Il video digitale comprende tutte le macchine che consentono la riproduzione di immagini statiche ed in movimento.

Le macchine multimediali, invece, nascono espressamente per usi **integrati** di testi, suoni ed immagini; tale specifica significa non una semplice somma di tecniche (*criteri*), bensì implica una nuova tecnologia, capace di trattare assieme tutte le problematiche.

E' necessario distinguere subito due aspetti rispettivamente legati alla progettazione ed all'utilizzazione dei sistemi multimediali:

- tecnologie per lo sviluppo;
- campi applicativi.

Per sviluppo si intende l'insieme delle fasi necessarie alla realizzazione dei prodotti multimediali, ovvero i programmi, le macchine e le conoscenze (inglobate nel sistema ad uso dell'utenza). I campi applicativi sono molteplici e stanno sempre più diffondendosi in moltissimi settori: dalla medicina all'ingegneria, dalla grafica alla televisione, etc.

La fase importante dello sviluppo, che riguarda il trattamento dei vari tipi di supporto dell'informazione (**media**), è la produzione. Questa si deve sempre articolare in sequenze di parti che determinano la composizione del prodotto finito:

- Progetto:
  - progetto testi;
  - progetto immagini;
  - progetto musiche;
  - progetto grafica per le pagine video;
  - progetto *software*, etc.
- Redazione testi.
- Realizzazione immagini:
  - statiche;
  - animate, etc.
- Raccolta suoni.
- Montaggio.
- Assemblaggio.

Devono quindi esistere degli appositi strumenti che consentano un'opportuna intercalazione di queste fasi; a tal proposito la progettazione di una stazione multimediale deve essere supportata da un ambiente di calcolo formato da particolari componenti *hardware* e da speciali *tools software* proprio per l'elevata complessità di trattamento degli eventi audio/video interallacciati.

Nascono così le architetture *special-purpose* in grado di trattare insieme immagini e suoni nella loro forma più elementare. In particolare, sono proposti *processors* ad elevate prestazioni che riescono a soddisfare i vincoli temporali imposti dal trattamento in tempo reale dei diversi eventi media.

L'attenzione è polarizzata, pure, sui dispositivi di massa; infatti una stazione multimediale è corredata da supporti ad elevata capacità di memorizzazione ed i cui tempi d'accesso risultino brevi. Ciò è richiesto a causa della grande quantità di dati occorrenti per definire le informazioni audio e video; di conseguenza sono necessari dispositivi che consentano il loro recupero nel più breve tempo possibile, altrimenti si può correre il rischio di non rispettare i vincoli temporali imposti per questo tipo di computazioni.

A fronte di questo problema sono state progettate delle tecniche per la compressione dei dati che tentano di eliminare la ridondanza dei contenuti informativi e ciò allo scopo di ridurre sensibilmente l'occupazione di memoria.

Anche il supporto *software* è ampiamente corredato di *tools* che favoriscono lo sviluppo di applicazioni multimediali.

Si è notato che il modo migliore per poter gestire gli eventi media è attraverso la programmazione orientata al trattamento degli oggetti. Sebbene l'impatto con questa risulti ostico si può affermare che, ad una lenta curva d'apprendimento seguono il potenziamento degli strumenti di programmazione ed un aumento delle prestazioni di gran lunga superiori a quanto ottenibile attraverso le tradizionali metodologie di sviluppo del *software*. Solitamente la programmazione orientata al trattamento degli oggetti viene proposta nel contesto di un ambiente integrato, allo scopo di consentire la progettazione delle applicazioni nel modo più scorrevole, secondo le esigenze degli utenti.

Come già accennato, la gestione degli eventi media comporta la considerazione di problematiche del tipo: efficienza di elaborazione - mole di dati - vincoli temporali. Devono, allora, esistere dei metodi di organizzazione di queste informazioni, magari supportate da basi di dati. Inoltre l'interazione tra *software* applicativo ed *hardware* deve essere mediata tramite l'intervento di particolari *routines* di sistema operativo, proprio per poter sincronizzare i flussi di dati multimediali attraverso i livelli gerarchici della macchina, in base ai vincoli temporali ai quali gli eventi stessi sono soggetti.

Quindi lo sviluppo di applicazioni multimediali, che per loro natura sono orientate allo *user-friendly*, deve avvenire in un ambiente appositamente predisposto, altamente interattivo ed in grado di rispondere efficientemente a successive modifiche.

1 LE ARCHITETTURE DEDICATE DEI SISTEMI MULTIMEDIALI

Un dispositivo che consente un'efficiente elaborazione, presentando un buon rapporto fra costo e prestazioni, sia dei suoni che delle immagini e largamente utilizzato nei sitemi multimediali è il DSP.

I DSP sono microcalcolatori fortemente orientati all'elaborazione dei segnali in tempo reale.

Il DSP è, in genere, *il cuore di una "catena" di dispositivi* che prelevano dal mondo esterno un segnale (suono, immagine, etc.), lo trasducono in un segnale analogico elettrico che viene a sua volta trasformato in segnale digitale grazie a dei convertitori A/D; in tale condizione viene elaborato dal DSP che lo restituisce ad un convertitore D/A, il cui compito è quello di trasformarlo nuovamente in un segnale della stessa natura che aveva prima di essere *passato* attraverso la catena.

Ma affinché la qualità del segnale *restituito* al mondo *reale* rimanga il più possibile inalterata, cioè perché sia mantenuto a fine manipolazione il contenuto informativo del segnale originario, le conversioni A/D e D/A devono essere effettuate in modo tale che il campionamento e la conversione della funzione analogica corrispondente avvenga con una frequenza pari, almeno, al doppio del limite superiore di banda dello spettro del segnale che si vuole campionare (***teorema di Shannon***).

Risulta, dunque, chiaro che un campione del segnale da convertire, permarrà all'interno della catena finché non venga sostituito dal successivo che, detta f_0 la frequenza di campionamento, si presenterà dopo $1/f_0$ secondi.

Da quanto detto segue che, se un segnale, la cui frequenza massima sia al più $f_{max} - f_0$, *attraversa* la catena A/D - DSP - D/A in un tempo minore, od al più uguale, ad $1/f_0$, esce da questa senza *"accorgersi"* di esservi passato; il tempo trascorso dal segnale sotto forma di campione all'interno della catena, non ha modificato la sua *realtà* temporale.

Il DSP è, dunque, in grado di elaborare il segnale in un tempo inferiore ad $1/f_0$, *"restituendolo"* al mondo esterno modificato, certo, ma senza averne interrotta la continuità logica e temporale. In tal senso, si dice che il DSP ha elaborato il segnale in *tempo reale*.

Le tipiche operazioni, eseguite da questo processore, sono la moltiplicazione e l'addizione, che portano la ALU ed il moltiplicatore ad essere due dei suoi componenti *hardware* fondamentali. Riveste una notevole importanza anche la memorizzazione e la rilettura dei risultati intermedi, che deve avvenire in tempi molto rapidi. Ciò significa che il DSP, oltre a disporre dei registri come un qualunque processore *general-purpose*, possiede pure una particolare struttura interna che organizza queste memorie locali in modo tale da favorire al massimo il flusso delle operazioni ed il *pipelining*.

L'architettura ideale per un *Digital Signal Processor*, mostrata in figura 1, è quella denominata *Harvard*, che prevede una separazione completa tra programma e dati, sia per quanto riguarda l'area di memorizzazione, che per le unità di controllo e di elaborazione.

Ciò significa che le locazioni di memoria vengono indirizzate per prelevare le istruzioni ivi contenute attraverso dei *bus* dedicati; così come altri *bus* dedicati vengono utilizzati per l'indirizzamento ed il prelievo dei dati.

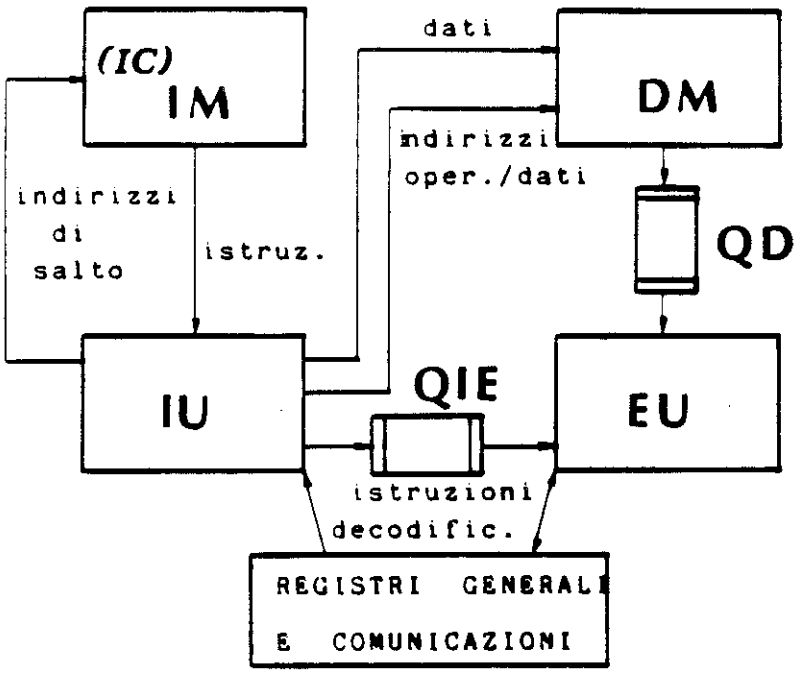


Figura 1

Grazie a tale architettura è possibile sfruttare pienamente il parallelismo nell'esecuzione delle istruzioni, che avviene contemporaneamente all'elaborazione (prelievo da memoria, invio a periferica, etc.) dei dati.

In un solo ciclo d'istruzione, ad esempio, è possibile prelevare dalla memoria i due operandi relativi ad un'operazione aritmetica e depositare in memoria il risultato dell'operazione stessa.

Se si pensa, inoltre, che un ciclo d'istruzione si compie, in media, (per un DSP che lavori ad una frequenza di *clock* di 40 MHz) in 100 ns, è possibile immaginare l'elevata velocità raggiungibile da queste macchine nell'elaborazione continuata di grosse quantità di dati.

Per quanto riguarda, infine, le istruzioni, il DSP è un processore di tipo **RISC**, cioè a *set* ridotto di istruzioni, ma estremamente potente ed orientato al calcolo numerico.

In sostanza, l'intero *set*, che non supera in genere le 50 istruzioni, può essere totalmente descritto come l'insieme di tre classi di istruzioni di calcolo, una classe di *shift*, una di trasferimento, una di salto ed una d'inizializzazione.

Le nuove generazioni dei DSP, possono disporre di un elevatissimo numero di *bits* per il trasferimento delle informazioni e per l'elaborazione aritmetica dei dati (che significa avere alta precisione di calcolo) ed, inoltre, dispongono di molti *bus* operanti in parallelo per l'esecuzione delle istruzioni.

Un'alternativa al DSP, per l'elaborazione dei segnali digitali, consisterebbe nell'utilizzazione del *transputer* che, sebbene presenti un'architettura abbastanza tradizionale e quindi richieda un tempo piuttosto elevato per l'esecuzione di operazioni tipiche del settore, può però sfruttare la sua peculiare caratterisitica di presentare un supporto *hardware* alla concorrenza efficiente e quindi essere utilizzabile un parallelo con altri *transputers*, permettendo di realizzare, così, sistemi con prestazioni interessanti. Nonostante esistano in questo settore dei prodotti che adottano il *transputer*, il rapporto fra costo e prestazioni di un sistema utilizzante più componenti di questo tipo, risulta ancora estremamente sfavorevole rispetto alla scelta di considerare più microprocessori DSP di medio costo.

Alla classe dei processori DSP appartiene anche l'insieme dei dispositivi integrati VLSI di tipo dedicato utilizzati su schede audio progettate per ambiente PC multimediale. Solitamente su ciascuno di questi dispositivi si implementa un particolare algoritmo prepro-

grammato per la sintesi sonora, il più comune fra i quali è quello della FM, od al più una classe ristretta di funzioni specializzate. Ogni scheda contiene inoltre un convertitore A/D - D/A che campiona da 8 *bits* a 16 *bits*, con frequenza di campionamento da 22 kHz a 48 kHz, un'interfaccia per la gestione degli eventi MIDI ed una sezione d'interfaccia verso il *bus* del *computer host* per l'acquisizione dei campioni.

Si deve tenere presente che nella valutazione delle prestazioni di un sistema per la gestione dei segnali audio vanno contemplati i seguenti parametri principali:

- Risoluzione di conversione A/D e D/A.
- Precisione aritmetica, ossia il numero di *bits* sui quali opera la ALU.
- Frequenza di campionamento.
- Velocità di elaborazione e quantità di memoria di lavoro.
- Varietà di tecniche di sintesi disponibili o programmabili.

Viene di seguito descritta, come esempio, un'architettura tipo che supporta l'elaborazione delle immagini correntemente adottata nei sistemi multimediali.

1.1 Dispositivi di restituzione di immagini

La funzione fondamentale di questi dispositivi è quella di convertire le immagini digitali elaborate in corrispondenti immagini analogiche e di visualizzarle sia su supporto fotografico che, soprattutto, sullo schermo di un tubo a raggi catodici. A questa funzione primaria possono venire aggiunte, delegate dal calcolatore ospite, anche funzioni più o meno complesse di elaborazione e di manipolazione selezionate fra quelle meglio definite, più affidabili e fondamentalmente più riccorrenti.

Questa tendenza, favorita dalla necessità di semplificare la gestione delle varie operazioni eseguibili sulle immagini e soprattutto dalla necessità di rendere le esecuzioni sempre più veloci, fa sì che i dispositivi di restituzione siano sempre più dotati di autonomia operativa.

La struttura e l'architettura di una stazione di rappresentazione, o restituzione, può essere variamente organizzata e configurata in funzione della complessità delle operazioni che dovrà svolgere e delle caratterisitche tecniche ed operative richieste dalle specifiche appli-

cazioni. Tuttavia, come è mostrato in figura 2, si può individuare uno schema funzionale che soddisfa le esigenze elaborative generali più ricorrenti. la struttura di un generico dispositivo di restituzione è in genere composta da un controllore od elaboratore locale, da una memoria di immagine, da speciali elaboratori di ingresso e di uscita, da dispositivi di conversione digitale analogica e di generazione del segnale video ed infine da un *monitor* televisivo di rappresentazione.

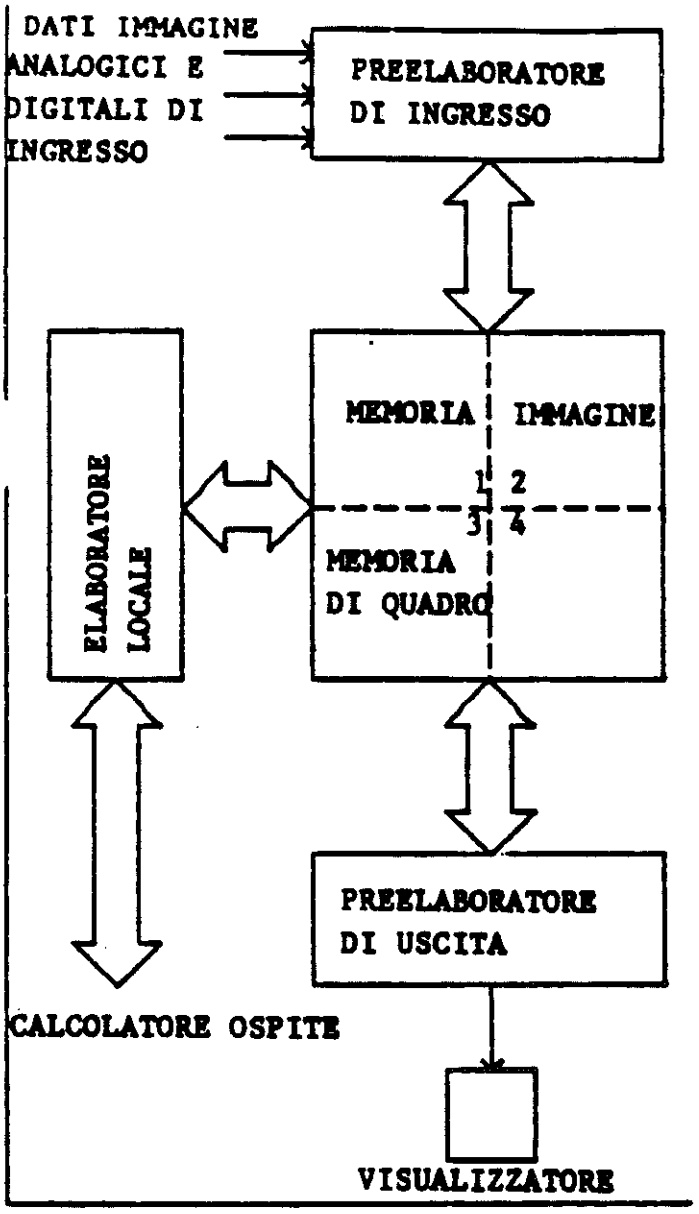


Figura 2

Il segnale digitale od analogico di ingresso viene memorizzato nella memoria immagine dopo una eventuale preelaborazione e quindi inviato al *monitor* dopo un'opportuna elaborazione di uscita che consente di variare il modo di rappresentazione secondo le specifiche esigenze. L'elaboratore locale può svolgere soltanto funzioni di gestione, controllo e temporizzazione delle funzioni interne sotto la supervisione di un calcolatore ospite, oppure può svolgere anche funzioni ed operazioni più complesse di trattamento e di gestione dei dati.

Nel primo caso il controllore locale è in genere basato su un microcalcolatore ad 8 *bits* oppure 16 *bits*, a seconda delle caratteristiche di velocità, precisione e potenza richieste, che interpreta ed esegue, mediante schemi e programmi prefissati, i comandi inviati dal calcolatore ospite al quale è riservata la funzione elaborativa. Nel secondo caso il controllore locale è basato su calcolatori programmabili del tipo *personal*, *mini*, *supermini* a 16 *bits* oppure a 32 *bits* organizzati con architettura a *bus* standard che garantisce le migliori caratterisitiche di versatilità, flessibilità, compatibiltià e programmabilità.

Questo tipo di struttura consente l'esecuzione di gran parte delle funzioni tipiche del trattamento di immagini mentre le operazioni più complesse possono essere affidate al calcolatore ospite che inoltre gestisce le risorse *hardware* e *software* generali.

Come è stato già accennato, i dati di ingresso possono provenire dal calcolatore ospite, dalle eventuali memorie di massa dell'elaboratore locale, oppure da speciali dispositivi di acquisizione (telecamera) direttamente collegati alla stazione di restituzione. In tutti i casi può essere conveniente eseguire una serie di elaborazioni durante la fase di trasferimento dei dati; in particolare se queste elaborazioni sono ben definte e ricorrenti si possono ottenere notevoli vantaggi in termini di velocità di esecuzione. Questo compito è assolto dall'elaboratore di ingresso che sulle immagini può svolgere in tempo reale operazioni di tipo globale (ad esempio, equalizzazioni), puntuale (ad esempio, modifica dinamica), oppure anche se più raramente, di tipo spaziale (filtraggi).

Le operazioni puntuali modificano il valore del singolo *pixel* secondo leggi o regole predefinite ed indipendentemente dai valori dei *pixels* circostanti. Le operazioni globali invece modificano i valori dei *pixels* in funzione della distribuzione che assumono nell'intera matrice immagine; individuata o definita la legge di distribuzione

l'elaborazione globale può essere ricondotta ad una di tipo puntuale. Sia le operazioni puntuali che gran parte di quelle globali possono essere eseguite impiegando specifici e semplici moduli aritmetici (moltiplicatori, sommatori), oppure più frequentemente, utilizzando tavole di conversione (LUT) che consentono di trasformare il dato di ingresso secondo funzioni predeterminate.

Sono disponibili moduli aritmetici estremamente veloci (20 - 50 nsec per eseguire ad esempio il prodotto su 16 *bits*) che consentono una effettiva elaborazione in tempo reale; tuttavia può essere necessario modificare il dato mediante l'esecuzione di una espressione che, comportando la concatenazione di più moduli elaborativi, rallenta eccessivamente il flusso dei dati. In questo caso può essere vantaggioso impiegare le tavole di conversione purché non sia indispensabile una grande imprecisione e sia ben definita la legge di trasformazione.

Le tavole di conversione sono realizzate mediante memorie RAM ad alta velocità di lettura (10 - 20 nsec) organizzate in forma di vettori o, più raramente, in forma di matrici; all'interno di queste memorie vengono trasferiti dall'elaboratore locale i dati di conversione opportunamente calcolati in precedenza secondo specifiche funzioni di trasformazione.

Nella figura 3 è mostrato un semplice esempio di questo tipo di modifica dei dati: il valore V di un generico *pixel* di ingresso $A[i,j]$, che può assumere valori discreti nell'intervallo $[a,b]$, viene utilizzato per indirizzare la v-esima cella del vettore di conversione LUT; il contenuto di $f(v)$ di questa memoria viene inviato e memorizzato nella posizione $A[i,j]$ della memoria immagine del dispositivo di restituzione.

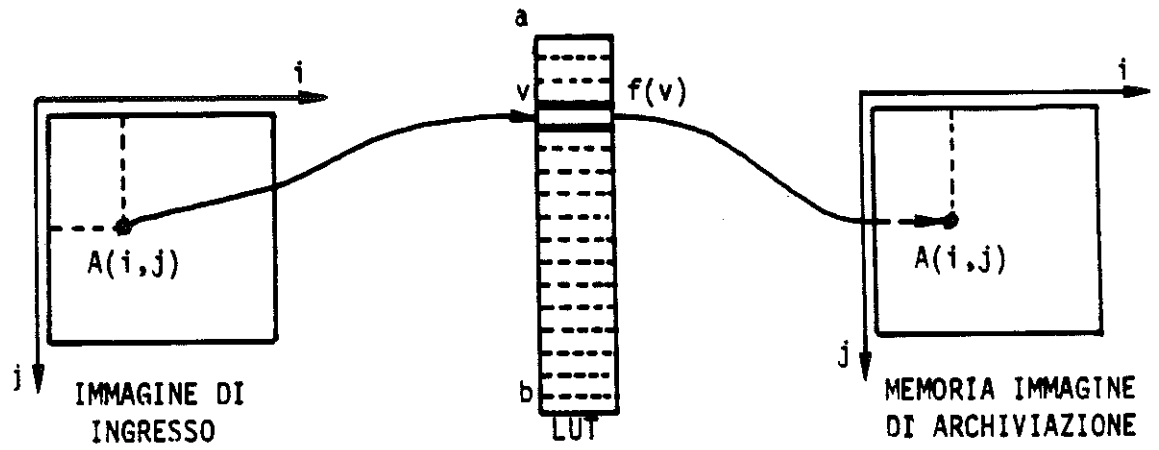


Figura 3

Nella figura 4 è mostrato un esempio di utilizzo di una tavola di conversione a matrice adoperata per eseguire, indirettamente, operazioni puntuali fra *pixels* corrispondenti, appartenenti a matrici immagine diverse. In questo caso i valori V ed U, definiti nello stesso intervallo $[a,b]$, relativi, rispettivamente, ai *pixels* $A[i,j]$ dalla matrice di ingresso M ed A^1 dalla matrice di riferimento MR, individuano nella matrice di conversione LUT la cella $N[v,u]$ il cui contenuto è una funzione definita nei valori V ed U; questo valore viene quindi trasferito e memorizzato nella memoria immagine MI del dispositivo di restituzione.

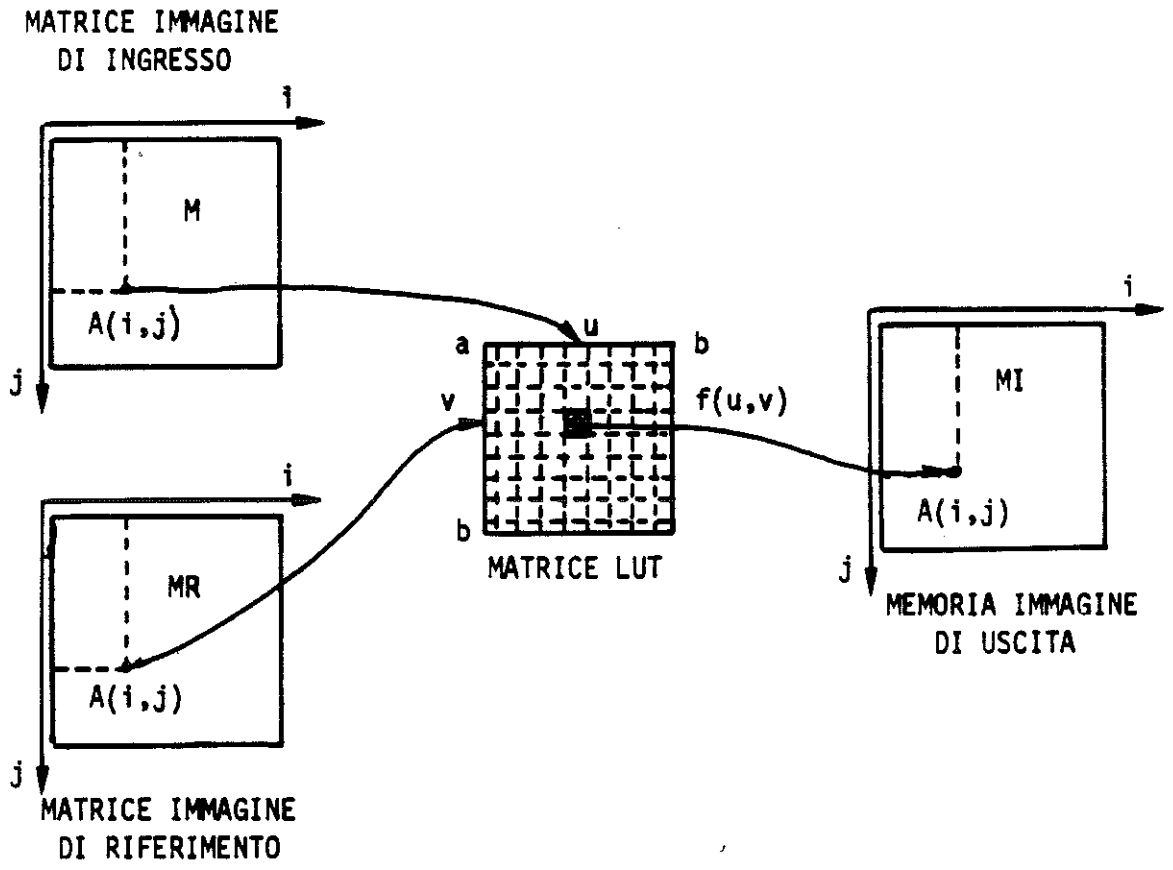


Figura 4

Questo metodo di elaborazione può risolvere efficientemente, soprattutto in termini di velocità, i problemi di elaborazione puntuali in tempo reale fra una matrice ed un vettore, oppure, come nell'esempio di figura 4, fra due matrici purché la dinamica dei valori discreti dei relativi *pixels* sia sufficientemente contenuta altrimenti la dimensione della matrice LUT potrebbe risultare eccessiva ed il procedimento poco conveniente.

Per eseguire operazioni spaziali o locali è invece necessario ricorrere a specifiche unità logico-aritmetiche programmabili mediante le quali è possibile trattare in vario modo i dati di ingresso secondo specifici algoritmi (convoluzioni, filtraggi bidimensionali, analisi statistiche, etc). Infine, specifiche funzioni elaborative particolarmente complesse (trasformazioni ortogonali etc) possono essere eseguite da speciali dispositivi dotati di capacità di calcolo propria ai quali viene delegata l'esecuzione dell'elaborazione.

1.2 La memoria di immagine

La memoria di immagine è il nucleo principale di un dispositivo di restituzione e la sua struttura ed organizzazione può condizionare e caratterizzare sia l'architettura generale del dispositivo che il suo modo di operare.

La memoria, in genere di tipo *refresh*, viene utilizzata sia per memorizzare i dati immagine provenienti dall'elaboratore locale od eventualmente da un dispositivo di acquisizione, sia per rinfrescare l'immagine che viene riprodotta sullo schermo di un *monitor* mediante sequenze di indirizzamento di tipo televisivo. Le dimensioni della memoria dipendono essenzialmente dal tipo di applicazione e dalle caratterisitiche di risoluzione e di dinamica di restituzione; in genere la dimensione standard è di 512*512 *pixels* espressi su 8 o più *bits*, mentre la dimensione massima dipende dalla capacità di indirizzamento del controllore locale ed, in genere, non supera 4096*4096 *pixels*.

La memoria di immagine può essere organizzata in banchi fisicamente o logicamente separati di 512*512 *pixels*, oppure 1024*1024 *pixels*, od ancora in un banco unico suddivisibile logicamente in memorie di quadro di dimensioni compatibili con la capacità di rappresentazione dei *monitors*. Quest'ultimo tipo di organizzazione è il più

utilizzato perché consente la memorizzazione di immagini di grandi dimensioni ed una efficiente e flessibile gestione ed organizzazione della rappresentazione dei dati; è possibile, infatti, ad esempio, rappresentare finestre di quadro che si possono spostare con continuità su immagini di dimensioni non rappresentabili sui *monitors*, oppure organizzare la memoria mediante semplici comandi inviati dal governo locale per consentire la memorizzazione di immagini di formato variabile.

Nello schema funzionale di figura 2 è indicata una memoria immagine suddivisibile in quattro memorie di quadro di 512*512 *pixels*, oppure in due memorie di quadro di 512*1024 *pixels*, od una di 1024*1024 *pixels*; nel primo caso le immagini potranno essere rappresentate su un *monitor* standard, mentre nella seconda e terza configurazione potranno essere rappresentate solo su un *monitor* ad alta risoluzione.

1.3 Elaboratore di uscita

Per visualizzare l'immagine digitale memorizzata nella memoria di quadro, è necessario trasferire i dati secondo sequenze temporali di tipo televisivo e convertirli in un segnale analogico di ampiezza proporzionale ai valori dei *pixels* ed in grado di pilotare il tubo a raggi catodici ed i circuiti di scansione del *monitor* di rappresentazione.

La legge di proporzionalità può essere variata mediante una opportuna elaborazione eseguibile prima della conversione digitale - analogica; a questo scopo sono in genere utilizzate delle tavole di conversione con metodologie analoghe a quelle esaminate per l'elaborazione di ingresso. In questo caso però i dati memorizzati non vengono modificati ma rimangono quindi disponibili nella memoria immagine per ulteriori elaborazioni, mentre viene variato il modo di rappresentarli trasformandoli mediante una LUT ed inviando il segnale così convertito al *monitor*.

Nella figura 5 è rappresentato lo schema di principio del processo di elaborazione di uscita per la rappresentazione mediante *monitor* a colori RGB: il valore V, compreso in un intervallo $[a,b]$, di un generico *pixel* $A[i,j]$ indirizza le v-esime celle di tre distinte tavole di conversione, rispettivamente, per la componente rossa (R), verde (G) e blu (B). I dati trasformati secondo specifiche leggi: $f(R)$, $f(G)$ ed $f(B)$ ven-

gono convertiti in tre distinte sequenze di segnali mediante i convertitori digitali - analogici DAC-R, DAC-G, DAC-B ed inviati ai tre canali *R*, *G* e *B* del *monitor*.

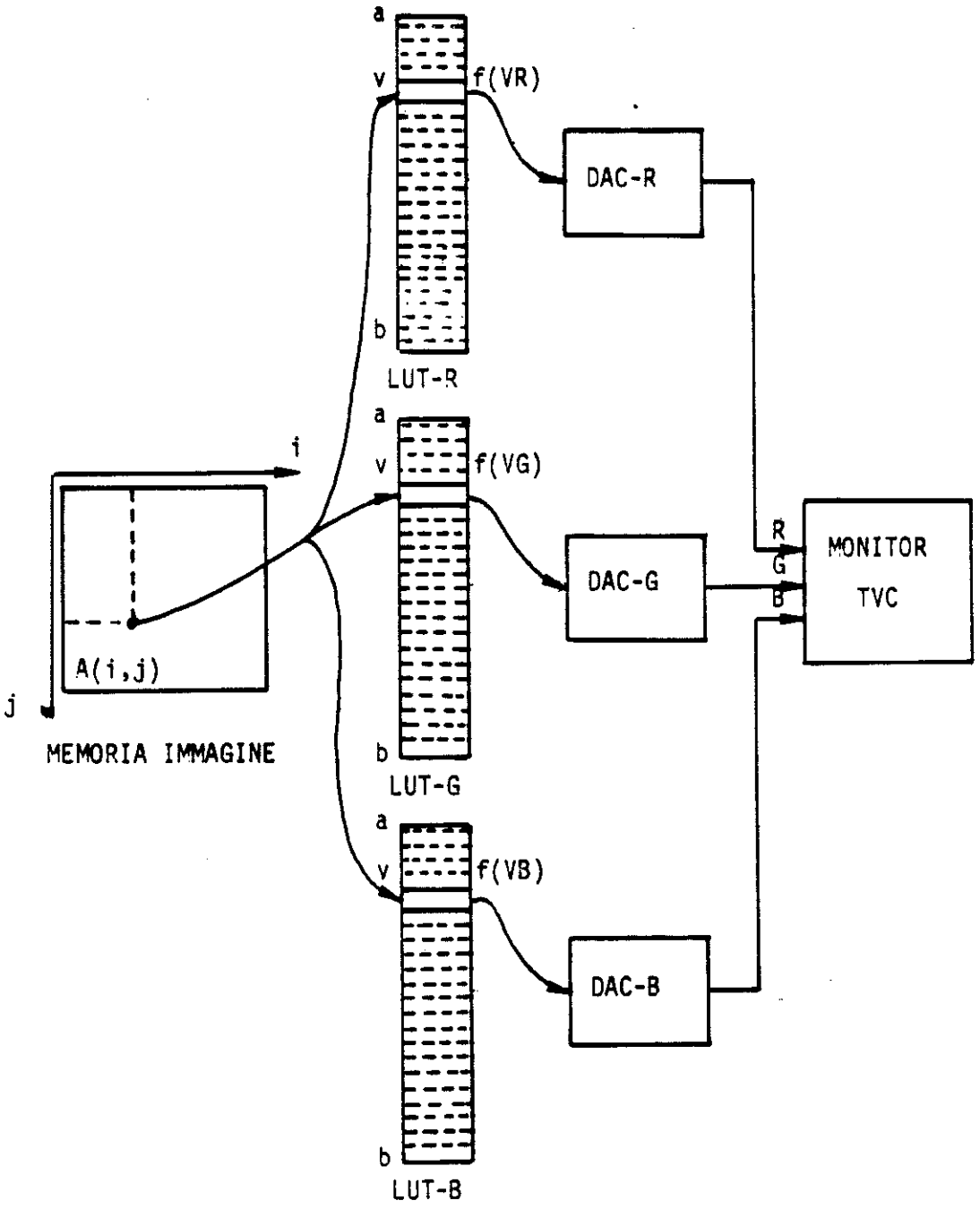


Figura 5

Con questo tipo di organizzazione è possibile, definendo opportunamente le funzioni $f(V)$ di trasformazione, ottenere rappresentazioni acromatiche, oppure policrome a falsi colori: il numero dei valori di grigio e la dinamica di rappresentazione dipendono dal numero di *bits* con cui sono espressi i valori dei *pixels* dell'immagine e dal numero di *bits* delle celle della LUT.

Ad esempio, per *pixels* espressi con 8 *bits* e LUT ancora di 8 *bits* è possibile ottenere 2^8 valori di grigio, oppure colori in una tavolozza di 2^{24} colori possibili.

La figura 6 mostra lo schema di principio per la rappresentazione del *"vero"* colore. In questo caso è necessario che l'immagine sia definita su tre memorie di quadro distinte per le tre diverse componenti R, G e B.

Utilizzando le tavole di conversione è possibile rappresentare l'immagine mediante la combinazione di un numero anche molto elevato di colori primari; infatti nell'esempio sopra citato si possono ottenere 2^{24} colori distinti poiché ogni colore può avere 2^8 possibili valori monocromatici.

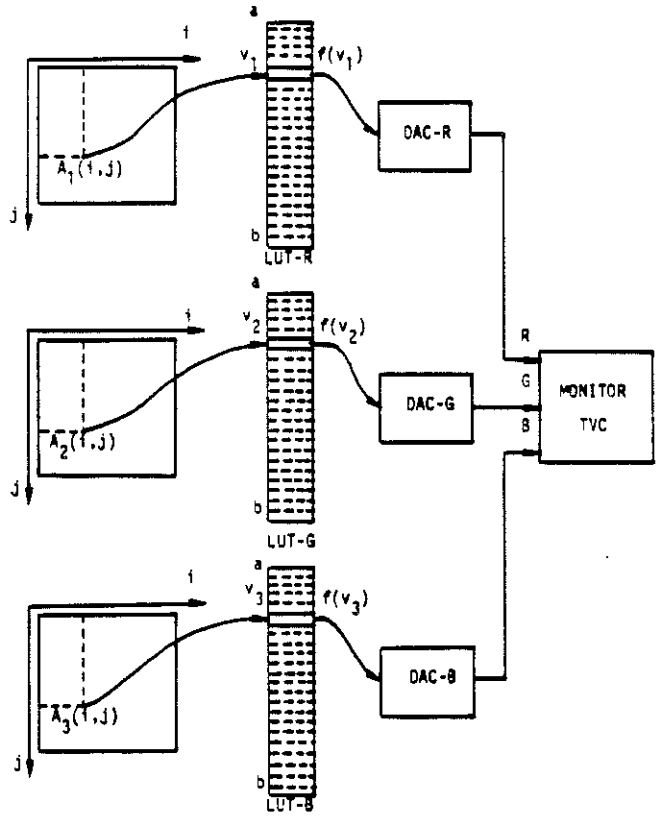


Figura 6

2 LA COMPRESSIONE DEI DATI MULTIMEDIALI: MPEG, UNO STANDARD PER LE IMMAGINI IN MOVIMENTO

Da diversi anni i sistemi per la trasmissione di immagini in movimento hanno assunto una notevole importanza sebbene l'ampia banda necessaria per tali trasmissioni abbia in parte frenato lo sviluppo di sistemi digitali per il loro trattamento. Questo ha favorito l'accettazione di *standards* analogici per la videoregistrazione e diffusione di filmati ed animazioni che, pur non presentando inconvenienti per la diffusione televisiva, non sono di fatto utilizzabili in campo scientifico.

La realizzazione di filmati analogici non consente infatti di archiviare un numero elevato di animazioni e di accedere in maniera efficiente ala singola animazione od al singolo fotogramma. Un filmato, una volta realizzato, non è più modificabile e la sua duplicazione comporta un degradamento significativo della qualità delle immagini. Soprattutto, malgrado le reti di trasmissione abbiano avuto un notevole sviluppo, non è tuttora possibile trasmettere attraverso di esse animazioni e filmati di buona qualità e questo rende difficile la collaborazione fra gruppi di persone in differenti aree geografiche.

Lo standard MPEG consente un'efficiente compressione di una sequenza di immagini permettendone quindi la memorizzazione, in formato digitale, sugli attuali supporti magnetici ed ottici e la trasmissione attraverso reti digitali.

2.1 Sistemi video digitali

I sistemi di trattamento ed elaborazione digitale di un segnale video si differenziano sostanzialmente in funzione della frequenza con cui i vari fotogrammi vengono visualizzati e delle dimensioni, in punti, di tali immagini.

Lo standard CCIR-601, ampiamente adottato da tutti i fabbricanti di apparecchiature video digitali per il mercato USA, prevede che le immagini trattate siano di 720*243 punti e che vengano visualizzate ad una velocità di 60 campi al secondo. Poiché un campo è costituito dalle sole righe pari o dispari di un fotogramma ed i campi vengono alternati ed interallacciati, visualizzando ogni campo per

1/60 di secondo si ottiene una sequenza di immagini alla frequenza di 30 fotogrammi al secondo.

I due canali di *crominanza* vengono decimati con rapporto 2:1 lungo la direzione orizzontale, e ridotti pertanto a 360*243 punti a 60 campi al secondo, essendo ancora visualizzati interallacciati. Questo tipo di decimazione dei canali di *crominanza*, molto usato in campo televisivo, è chiamato 4:2:2 ed indica la frequenza di campionamento utilizzata per Y, U e V espressa in rapporto alla frequenza di riferimento di 3,365 MHz.

Purtroppo per trattare questo genere di segnali occorrerebbe un sistema con una banda capace di trattare un flusso di *bits* dell'ordine di:

720*243*60*(8+4+4) = 160 Mb/sec.

Tale banda supera di diversi ordini di grandezza quella usualmente disponibile su reti di trasmissione digitale, che è dell'ordine di 10 Mb/sec, e, soprattutto, è di gran lunga superiore al *data-rate* dei dispositivi di *input* e di *output*, tipicamente CD, DAT, o dischi magneto-ottici, attualmente dell'ordine di 1,5 - 2 Mb/sec.

2.2 Lo standard MPEG

Il comitato MPEG (*Motion Picture Experts Group*) ha concluso, alla fine del 1991, il *"Committee Draft of Phase 1"*, comunemente chiamato MPEG 1, che stabilisce la strutura di un *bit stream* per audio e video digitali compressi che rientri all'interno del *data-rate* di 1,5 Mb/sec. Un segnale conforme allo standard MPEG 1, può quindi essere memorizzato sugli attuali dispositivi ed essere trasmesso attraverso le attuali reti di trasmissione digitale, sia locali che geografiche. Il *Draft* si divide in tre parti: Video, Audio e Sistemi. Le prime due (ISO 11172-1 ed ISO 11172-2) specificano il tipo di compressione da effettuare sui segnali audio e video, mentre la terza specifica come deve essere effettuata l'integrazione fra i due, consentendo pertanto la sincronizzazione fra audio e video.

Le tecniche che vengono utilizzate per la compressione, decimazione dei canali di *crominanza*, trasformata discreta del coseno, quantizzazione dei coefficienti e compressione di Huffman, derivano in gran parte da quelle proposte dallo standard JPEG (*Joint Photographics Experts Group*) per la compressione di immagini fisse.

Tale standard, per i buoni risultati che si riescono ad ottenere con immagini a colori od in toni di grigio, è molto diffuso nella comunità scientifica ed utilizzato in molte apparecchiature nei campi della fotografia ed elaborazione delle immagini.

Il segnale in ingresso per lo standard MPEG 1, chiamato SIF (*Source Interface Format*), equivale ad un segnale CCIR-601 decimato con rapporto 2:1 in orizzontale, 2:1 lungo l'asse dei tempi ed ancora 2:1 lungo l'asse verticale dei due canali di *crominanza*. Qualche linea viene ulteriormente eliminata per fare in modo che le dimensioni delle immagini siano multiple di 8 e 16 e semplificare così i calcoli successivi. Questo porta ad una risoluzione per il formato SIF di 352*240 punti a 30 fotogrammi al secondo.

In Europa la frequenza di visualizzazione, per gli standard PAL e SECAM, è di 50 campi al secondo. Poiché però il numero di righe in un campo è superiore (288 anziché 243) la risoluzione del SIF europeo è di 352*288 punti a 25 fotogrammi al secondo. Dal punto di vista del *bit-rate* i due formati sono esattamente equivalenti essendo entrambi costituiti dallo stesso numero di punti al secondo (visto che 352*288*50 = 352*240*60). Di conseguenza in seguito consideremo solo il formato SIF US.

Abbiamo visto che nelle immagini a colori in ingresso nel formato SIF, che considereremo già convertite nello spazio YUV, i due canali di *crominanza* (U e V) vengono ulteriormente decimati con rapporto 2:1, ottenendo quindi delle immagini di 352*240 punti per Y e 176*120 per U e V. Questo fatto non comporta un degradamento percepibile delle immagini se queste sono naturali (ad esempio: le fotografie), che purtroppo può essere visibile se le immagini sono generate sinteticamente (ad esempio: immagini ed animazioni generate al *computer*).

La qualità generale dell'immagine non decade comunque in maniera particolare.

Il *bit-rate* di una sequenza di immagini SIF non compresse è quindi:

(352*240+176*120+176*120)*30*8 *bit*/sec = 30 Mb/sec

cioè un valore che è ancora molto al di sopra dell'obiettivo prefissato.

Lo standard prevede quindi l'impiego di alcune sofisticate tecniche per la ridondanza dei dati in ingresso e diminuire il *bit-rate* di almeno un ulteriore fattore 20.

Viene di seguito esaminata la tecnica di *Motion Compensation* per la riduzione della ridondanza temporale e la trasformata discreta del coseno (DCT) per fornire un'addizionale compressione dei dati attraverso la riduzione della ridondanza spaziale.

2.3 *Motion Compensation*

La maggior parte dei fotogrammi di una sequenza è in genere molto simile: le differenze fra un fotogramma ed il successivo sono generalmente dovute solo a traslazioni di alcune sue parti. Ha quindi senso pensare di evitare di trasmettere le parti che non sono cambiate e di trasmettere, per quelle che si sono spostate, solo il verso e l'entità dello spostamento (*motion vector*).

Nello standard MPEG esistono tre tipi di fotogrammi: fotogrammi che vengono codificati singolarmente senza nessun riferimento ad altri (*Intraframes* od *I frames*), fotogrammi che vengono predetti sulla base di un *frame* di tipo *I* (*Forward predicted frames* o *P frames*) e fotogrammi che vengono ottenuti interpolando fra un *frame I* ed un *frame P* (*Bidirectional frames* o *B frames*). In MPEG quindi la predizione di un fotogramma può essere fatta considerando sia la *"storia passata"* (*I frames*) che quella *"futura"* (*P frames*). Tale processo è schematizzato in figura 7.

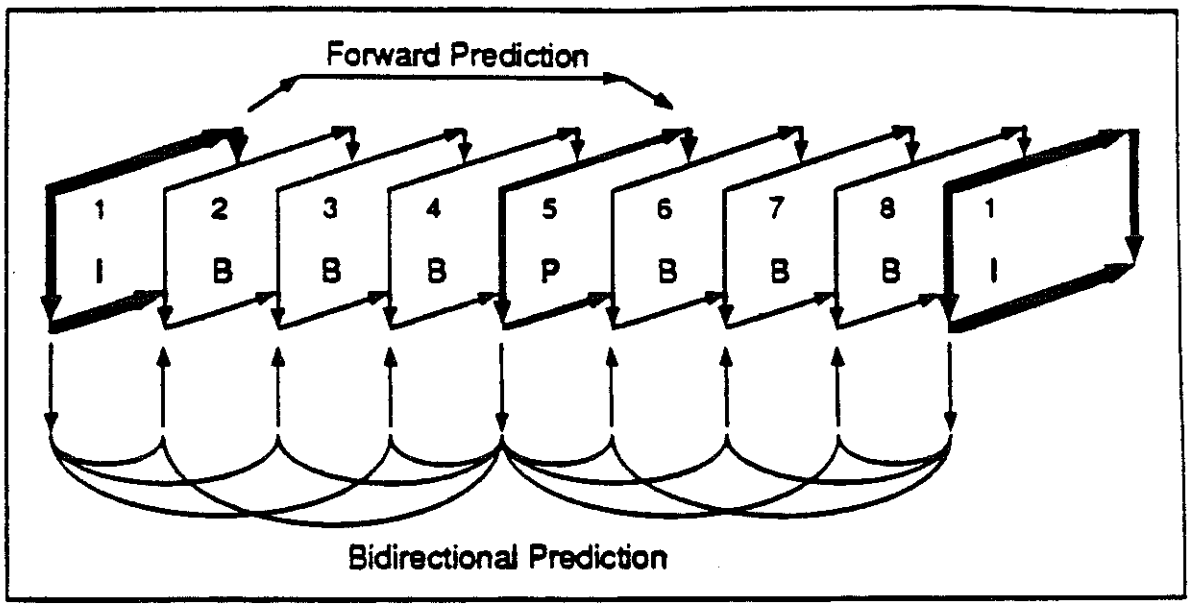


Figura 7

In sostanza come primo passo viene generato un *frame I* considerandolo come una singola immagine fissa. Per il calcolo del *motion vector* e la predizione del *frame P* si considerano i punti all'interno di blocchi 16*16 (*macroblocks*) nel canale di *luminanza* Y e nei corrispondenti blocchi 8*8 nei canali di *crominanza* U e V. Per ognuno di questi blocchi si cerca quello che gli si avvicina di più nell'ultimo *frame I* o *P* inviato; il verso e la direzione fra questi due blocchi identificano il *motion vector*. Se si riesce ad individuare il *motion vector*, per specificare il blocco nel *frame P* che stiamo codificando basterà indicare, oltre ovviamente al *motion vector* stesso, la differenza fra i punti dei due blocchi in esame. Una volta codificato un *frame I* ed uno *P* si possono codificare i *frames B* compreso fra essi.

A questo punto si esaminano i *"macroblocks"* dei fotogrammi compresi fra il *frame I* e quello *P* cercando per ogni blocco quello più simile nel *frame I* (indietro nel tempo), oppure cercando di fare una media fra il blocco più simile nel *frame I* e quello più simile nel *frame P* e quindi sottraendo al valore ottenuto il blocco da codificare.

Se nessuno di questi tre procedimenti dà un risultato soddisfacente si può sempre codificare il blocco come se facesse parte di un *frame I* ovvero senza riferimenti a blocchi precedenti o futuri.

Quindi otteniamo logicamente una sequenza di frrames del tipo:

I B B P B B P B B P B B I B B P B B P B B P B B I . . .

in cui ci devono essere (codifica SIS US) al massimo 12 *frames* fra un *frame* di tipo *I* ed il successivo, mentre la successione di *frames P* e *B* è libera.

Questo permette di poter avere un accesso casuale al flusso di immagini ogni:

$$1/30*12 = 0{,}4 \text{ sec}$$

e di fare in modo che le immagini codificate non divergano eccessivamente da quelle reali.

Naturalmente, poiché per decodificare i *frames B* occorre conoscere già il *frame P* successivo, la sequenza dei *frames* che vengono inviati dopo la codifica è diversa da quella logica ed è come segue:

Tipo I P B B P B B P B B P B B I B B P B B P B B P B B I . . .

Istante 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 2 . . .

0 3 1 2 6 4 5 9 7 8 2 0 1 5 3 4 8 6 7 1 9 0 4 2 3 5 . . .

Quindi le operazioni che si compiranno leggendo un flusso MPEG saranno sostanzialmente queste:

1) Lettura e decodifica del *frame I* (t = 0).
2) Lettura e decodifica del *frame P* (t = 3), visualizzazione *frame I* (t = 0).
3) Lettura e decodifica del *frame B* (t = 1), visualizzazione *frame B* (t = 1).
4) Lettura e decodifica del *frame B* (t = 2), visualizzazione *frame B* (t = 2).
5) Lettura e decodifica del *frame P* (t = 6), visualizzazione del *frame P* (t = 3).
6) Lettura e decodifica del *frame B* (t = 4), visualizzazione del *frame B* (t = 4).
7) Lettura e decodifica del *frame B* (t = 5), visualizzazione del *frame B* (t = 5).
8) Lettura e decodifica del *frame P* (t = 9), visualizzazione del *frame P* (t = 6).
9) . . .

2.4 Stadi di trasformazione

2.4.1 DCT

La codifica delle immagini attraverso il *motion vector* di ogni *"macroblock"* con lo schema appena visto non riesce purtroppo a ridurre la quantità di *bits* necessari per la rappresentazione delle immagini in ingresso. Visto che però, in genere, l'energia associata ai valori delle componenti Y, U e V di un'immagine è distribuita in una banda di frequenze abbastanza ridotta, ha senso applicare una trasformata ed aspettarsi di trovare informazioni spettrali solamente in alcuni coefficienti.

Per ragioni di semplicità si è scelto di dividere l'immagine in blocchi di 8*8 punti ed applicare la DCT (Trasformata Discreta Coseno), un tipo di trasformata spesso utilizzata per la compressione di immagini. La trasformata DCT bidimensionale può essere ottenuta applicando la DCT monodimensionale alle colonne ed alle righe di ogni blocco. La sua espressione formale, in funzione del valore dei *pixels* del blocco $f(i,j)$ e dei coefficienti $F(u,v)$ nel dominio della frequenza, è la seguente:

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

dove:

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & x = 0 \\ 1 & altrimenti \end{cases}$$

La trasformazione inversa della DCT bidimensionale è invece espressa come segue:

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

I coefficienti ottenuti, che sono in numero pari a quello dei punti del blocco, vengono organizzati in una matrice in modo che il valor medio, ovvero la componente continua, si trovi nell'angolo in alto a sinistra. I coefficienti delle componenti alternate, come mostrato in figura 8, vengono invece disposti in ordine decrescente, con i valori più alti vicino alla componente continua ed in modo da avere un numero di riga/colonna che cresce in funzione del valore della frequenza verticale/orizzontale.

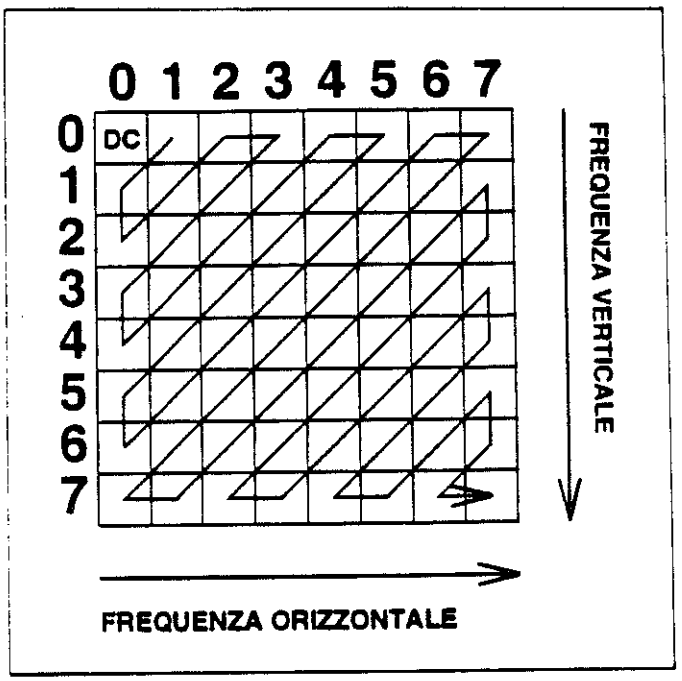


Figura 8

2.4.2 Quantizzazione

Come si è già detto il numero dei coefficienti ottenuti per le varie componenti che hanno un valore nullo è abbastanza elevato. Per aumentarlo, introducendo naturalmente della distorsione e rinunciando a parte dell'informazione originale, gli stessi coefficienti vengono quantizzati. Per semplicità viene usato un metodo di quantizzazione uniforme, usando però un passo diverso a seconda della posizione del coefficiente della DCT e del tipo di *frame*.

Nei blocchi dei *frames* di tipo *I* vengono quantizzate separatamente le componenti alternate e continua secondo le seguenti espressioni:

$$C(0,0) = \left\lfloor \frac{F(0,0) \pm 4}{8} \right\rfloor$$

$$A(u,v) = \left\lfloor \frac{16F(u,v) \pm \frac{Q(u,v)}{2}}{Q(u,v)} \right\rfloor$$

$$C(u,v) = \left\lfloor \frac{A(u,v) \pm Q_F}{2Q_F} \right\rfloor$$

dove $C(u,v)$ sono i coefficienti quantizzati, $F(u,v)$ i coefficienti della DCT, $Q(u,v)$ è il passo del quantizzatore, Q_F è un parametro di quantizzazione ed il segno +/- viene preso positivo per $F(u,v)$ positiva e negativo per $F(u,v)$ negativa.

La quantizzazione inversa si ottiene ponendo:

$$F(0,0) = 8C(0,0)$$

$$\mathrm{F}(u,v) = \frac{C(u,v)Q(u,v)Q_F}{8}$$

Mentre per i blocchi dei *frames* di tipo *P* e *B* il quantizzatore è lo stesso per entrambi i tipi di componente e dato da:

$$A(u,v) = \left\lfloor \frac{16F(u,v) \pm \frac{Q(u,v)}{2}}{Q(u,v)} \right\rfloor$$

$$C(u,v) = \begin{cases} \dfrac{A(u,v)}{2Q_F} Q_F dispari \\ \dfrac{A(u,v) \pm 1}{2Q_F} Q_F pari \end{cases}$$

ed il segno +/- viene preso positivo per $A(u,v)$ positiva e negativo per $A(u,v)$ negativa.

La quantizzazione inversa si ottiene da:

$$F(u,v) = \frac{(2F(u,v) \pm) Q_F Q(u,v)}{16}$$

La quantizzazione è la parte della codifica MPEG in cui si introduce l'errore maggiore dato che il cambiamento di scala di un coefficiente quantizzato comporta errori non recuperabili se il coefficiente in questione non è multiplo del passo di quantizzazione. Questa condizione non è, ovviamente, in genere rispettata per il modo in cui viene fatta la quantizzazione, l'entità dell'errore dipende dalla posizione spaziale del campione; l'errore massimo che si può commettere è pari alla metà del passo di quantizzazione.

Se la quantizzazione viene fatta in modo troppo *"grossa"* le immagini ottenute appaiono segmentate e divise in grossi agglomerati; viceversa, se viene fatta in maniera troppo *"fine"*, si utilizzano più *bits* del dovuto per trasmettere informazione sostanzialmente solo legata a rumore.

2.4.3 Codifica Entropica

Abbiamo a questo punto ottenuto, per ciascuno dei blocchi di ogni *frame*, una matrice nella quale i coefficienti delle componenti spettrali date dalla DCT, sono ordinati a *zig-zag*, in funzione della frequenza. Sappiamo inoltre che, nella maggior parte delle immagini, sono le componenti a frequenza più bassa ad avere i valori maggiori, mentre quelle a frequenza più elevata hanno in genere valore nullo. Risulta pertanto naturale operare una codifica dei coefficienti di tipo RLE (*Run Lenght Encoding*).

I coefficienti delle componenti continue, in genere diversi da zero, sono codificati scrivendo il numero di *bits* significativi nel loro valore seguiti dagli stessi *bits* significativi. I coefficienti delle compo-

nenti alternate, in genere uguali a zero, sono codificati scrivendo il numero di zeri consecutivi ed il primo valore non nullo. I codici che si ottengono dalla codifica RLE vengono ulteriormente compressi con un codificatore di Huffman, scelto per la sua facilità di implementazione in *hardware*, utilizzato in una variante leggermente semplificata (per il funzionamento del codificatore di Huffman si rimanda alla letteratura ed alla bibliografia).

2.5 Prestazioni

Partendo da immagini originariamente in formato SIF e convertendole in MPEG si ottiene un fattore di compressione di circa 30:1. La qualità delle immagini risultanti è sostanzialmente comparabile con quella delle immagini in ingresso. Viceversa se le immagini originarie sono di qualità superiore, ad esempio CCIR-601 od immagini RGB ottenute da un *computer* che poi vengono convertite in SIF, il fattore di compressione è molto più alto, anche 100:1, ma il degrado di qualità delle immagini è molto più marcato.

Ad esempio, dato che ogni fotogramma occupa 352*288*3 = 304128 *bytes*, la memorizzazione di un filmato di 165 *frames* in formato RGB a 352*288 punti richiede circa 50 MB. Lo stesso filmato, compresso in MPEG senza sacrificare troppa qualità, occupa solo 1,1 MB denotando quindi un rapporto di compressione di circa 50:1. La trasmissione dello stesso filmato, supponendo un *data-rate* di 1 Mb/sec, comporterebbe 400 secondi, mentre la versione compressa può essere trasmessa in poco meno di 9 secondi.

Uno dei punti più critici sta nel fatto che al codificatore è lasciata un'ampia libertà di scelta sui tipi di *frames* da utilizzare e di codifica da effettuare sui singoli *macroblocks*. Pertanto a parità di sequenza di immagini in ingresso, ma al variare dei parametri del codificatore, si possono ottenere un gran numero di sequenze diverse, sempre rispettanti lo standard MPEG, ognuna con un diverso valore di compressione e diversa qualità.

Oltretutto, benché si stiano facendo molte ricerche in questo campo, vi è un forte interesse da parte dei costruttori di *hardware* a mantenere riservati i risultati della loro attività di *R & D* per acquisire vantaggi competitivi nella fabbricazione di codificatori su singolo *chip*. Quindi, anche utilizzando codificatori *software*, il *tuning* dei vari

parametri è particolarmente critico e le prestazioni che si ottengono variano molto a seconda della configurazione o delle immagini in ingresso.

In generale, ottenendo un risultato di qualità media, ci si può aspettare che i *frames* di tipo *I* siano compressi di un fattore 10:1, quelli di tipo *P* di un fattore 30:1 e quelli di tipo *B* di 60:1. I filmati MPEG attualmente disponibili hanno rapporti di compressione variabili fra 30:1 sino a 100:1.

2.6 Applicazioni

Le applicazioni della compressione di immagini in movimento sono molteplici e, per la maggior parte, orientate verso l'elettronica di consumo (videogiochi, videotelefono e videoconferenze, educazione a distanza etc). In campo scientifico la possibiltà di codificare filmati ed animazioni in *files* di ridotte dimensioni consente nuove possibilità di collaborazione fra i vari centri di ricerca.

Molti stanno già organizzando delle *"digital galleries"* che rappresentano le ricerche effettuate ed i risultati ottenuti con brevi filmati MPEG. Ad esempio, l'NCSA (*Natonal Center for Supercomputing Applications*) presenta le sue attività nello *"Science Theater"* con una serie di filmati MPEG, accessibili attraverso la rete Internet, che rappresentano lavori svolti nei campi della fluidodinamica, astrofisica e visualizzazione volumetrica.

E' già possibile inviare dei filmati MPEG attraverso i vari sistemi di posta elettronica dato che lo standard MIME (*Multipurpose Internet Mail Extension*) consente di incorporare blocchi di video MPEG all'interno di un messaggio. Inoltre lo standard MPEG è supportato dal sistema WWW (*World Wide Web*) e dal *browser* XMOSAIC, sotto X-WINDOW, per la ricerca ipertestuale di informazioni sulla rete Internet. Sulle USENET *news* sono comunissimi i *postings* di filmati MPEG nei gruppi dove un tempo venivano scambiate immagini fisse in formato GIF o TIFF.

3 SPECIFICHE DI UN SISTEMA OPERATIVO MULTIMEDIALE

Gli eventi multimediali sono legati a vincoli temporali, per cui vengono caratterizzati da relazioni di sincronizzazione atte governare il rispetto di tali vincoli. Riveste quindi un aspetto fondamentale la progettazione di un adeguato sistema operativo che sia in grado di soddisfare, nei termini di tempo imposti, le operazioni di raccolta ed instradamento dei vari flussi di dati che compongono un evento multimediale complesso. Sembra scontato che un sistema operativo in tempo reale possa risultare rispondente alle esigenze alle quali sono sottoposte questo tipo di dati a tempo d'esecuzione.

Inoltre per non appesantire il carico di lavoro di un sistema di elaborazione è auspicabile adottare un processore dedicato alle operazioni di schedulazione dei processi, mentre quello principale si occupa della loro elaborazione; quindi la configurazione ottimale è quella di avere un sistema a processori dedicati.

Sebbene sia configurabile una simile architettura non è, d'altra parte, presente nella maggior parte delle *workstations* un sistema operativo che operi in tempo reale. Per cui attualmente vengono implementati dei supporti alla schedulazione dei processi che possono essere visti di tipo *pseudo tempo reale*, ossia sono presenti in un contesto di tipo tradizionale, ma vengono proposti in maniera da soddisfare le esigenze temporali dei dati e quindi le relazioni di sincronizzazione che intercorrono tra loro.

Allora un buon sistema operativo multimediale deve fondamentalmente disporre di uno schedulatore di processi abbastanza *"flessibile"* (intendendo con ciò la capacità di rispondere a richieste non deterministiche effettuate al sistema), di un *kernel* che sia in grado di supportare un ampio spettro di carico di lavoro e dei meccanismi che consentano la sincronizzazione degli eventi.

Il requisito di flessibilità richiesto per la progettazione dello schedulatore, costituisce uno degli aspetti qualitativi principali da considerare nella formazione di un ambiente multimediale. Ad esempio, la schedulazione in un classico sistema in *time-sharing* non è rispondente alle esigenze dei dati multimediali in quanto non contempla la temporalità insita in questo tipo di eventi, bensì la commutazione di contesto del *processor* è legata a fattori estremamente rigidi e totalmente deterministici.

3.1 La sincronizzazione dei dati multimediali

Per quanto riguarda la sincronizzazione dei dati multimediali questa può avvenire attraverso due modi: da un lato si può fare in maniera che i flussi di dati contengano le opportune informazioni, ovvero siano in grado di *"autosincronizzarsi"*, oppure, da un altro lato, si richiede l'intervento di meccanismi esterni.

Assume importanza il campo d'azione della sincronizzazione, ovvero l'ambito nel quale viene effettuata, nella gerarchia di sistema definita come la sovrapposizione dei livelli: *controllers* dei dispositivi, *kernel* del sistema operativo, applicazioni utente.

Vengono di seguito considerati quattro aspetti che emergono dalla gestione dei processi attraverso la gerarchia di sistema.

a) Classi di sincronizzazione. Vi sono due classi di sincronizzazione in un ambiente multimediale: la classe per la sincronizzazione seriale e la classe per la sincronizzazione parallela.

La prima determina la frequenza secondo la quale gli eventi accadono in un flusso di dati multimediali; considera, ad esempio, la frequenza di elaborazione delle informazioni audio e quella delle informazioni video.

La seconda determina la schedulazione relativa ai flussi di sincronizzazione separati. Nella maggior parte delle applicazioni multimediali non banali, ciascun flusso ha esigenze di sincronizzazioni seriali e parallele, in relazione con altri flussi. Si noti che un caso speciale di sincronizzazione seriale può essere definito in maniera composta, ossia ciascun blocco seriale di dati contiene informazioni per flussi paralleli in *output*. In altre parole, si ha che la sincronizzazione parallela tra blocchi viene forzata in un flusso seriale.

b) Portata della sincronizzazione. Si deve distinguere tra sincronizzazione per punti e continua. La prima richiede soltanto che un punto singolo di un blocco coincida con un punto singolo di un altro. La seconda richiede la sincronizzazione racchiusa tra due eventi a lunghissima durata. In generale, la sincronizzazione per punti può essere gestita al livello delle applicazioni, mentre quella continua avrà la necessità di essere gestita dai *controllers* dei dispositivi, o comunque, da alcuni programmi del sistema operativo.

c) Controllo delle sincronizzazioni. La terza distinzione riguarda il controllo delle entità in un (insieme di) flusso(i). Si devono conside-

rare i tipi di canali di comunicazione utilizzati; ad esempio, alcune volte due canali possono avere uguale importanza, mentre altre volte si ha che un canale funge da *master* e l'altro da *slave*. E' pure possibile che un *clock* esterno giochi il ruolo di sincronizzatore sia per tutti i flussi, che per un sottoinsieme di essi.

d) Precisione della sincronizzazione. Vi sono diversi livelli di precisione. Ad esempio, i canali per il suono stereo devono essere sincronizzati attentamente (entro 1 ms, con uno scarto di 0.1 ms), poiché la percezione degli effetti stereo si basa su minime differenze di fase. La sincronizzazione di una colonna sonora, con relativa sequenza di immagini, richiede una precisione che varia da un minimo di 10 ms ad un massimo di 100 ms. I sottotitoli possono avere un'imprecisione che oscilla tra 0.1 sec ed 1 sec.

4 LA PROGRAMMAZIONE DEGLI EVENTI MULTIMEDIALI

La metodologia di programmazione di tipo multimediale deve principalmente considerare quali devono essere gli strumenti attraverso i quali è possibile effettuare un interfacciamento verso gli eventi, ovvero quale è il tipo di approccio che consente l'interazione e la sincronizzazione fra i vari dispositivi coinvolti in un'applicazione.

Le attuali applicazioni multimediali sono usualmente programmate in linguaggi convenzionali, come il C, potenziati da uno specifico *hardware* per la gestione di appropriate librerie. Alcune applicazioni possono essere prodotte con *tools* che generano codice per interfacciarsi con dipositivi di tipo multimediale. In alcuni casi qualche sostituzione all'equipaggiamento può richiedere maggiori modifiche nei *tools*, nuovi metodi d'interfacciamento ed almeno la rigenerazione del codice eseguibile delle applicazioni.

Le domande che sorgono sono:

1) Perché le applicazioni multimediali sono dipendenti dall'*hardware*?

Le funzioni multimediali per specifici dispositivi, costituiscono le principali ragioni dell'insufficienza dell'attuale programmazione in questo campo. E' diverso ricevere dati da dispositivi multimediali e controllarli, piuttosto che ricevere semplicemente caratteri da una tastiera; vi è un aumento di complessità. Inoltre, diversi dispositivi come, ad esempio, una telecamera, un microfono, un lettore di *compact disc*, differiscono qualitativamente dal modo in cui acquisiscono le informazioni e, quindi, differiscono anche nell'implementazione delle pertinenti funzioni.

2) Come si può superare questo problema?

Tramite l'uso di appositi sistemi operativi ed un adeguato livello d'astrazione della programmazione, si può recuperare l'insufficienza dovuta all'implementazione di funzioni multimediali per specifici dispositivi. Però, non si deve credere che qualsiasi approccio sia universalmente applicabile; tuttavia, non si deve nemmeno dubitare che vi sia soltanto una soluzione. I linguaggi ad alto livello, ad esempio, richiedono sviluppi che coinvolgono alcune astrazioni.

Esistono diversi metodi attraverso i quali è possibile produrre del *software* di tipo multimediale. Ad esempio, possono esistere *routines* contenute in apposite librerie che si occupano della gestione dei dispositivi; un altro approccio consiste nello sviluppo di programmi dove gli eventi multimediali vengono definiti per mezzo di

opportuni tipi di dato. Infine la la programmazione orientata al trattamento degli oggetti viene vista come la forma più naturale tramite la quale è possibile esprimere le azioni e le informazioni che sono coinvolte per applicazioni in questo settore.

Volendo dare una definizione dei tipi di dati astratti, si può dire che questi consistono in interfaccie di specificazione, senza alcuna descrizione di algoritmi interni. Usando tale definizione, s'incontreranno alcune rigide limitazioni: la comunicazione e la sincronizzazione non possono essere espresse direttamente dalla semplice definizione dei tipi di dati astratti.

Invece, negli ambienti orientati al trattamento degli oggetti, la programmazione multimediale è affrontata tramite l'implementazione e l'espansione di gerarchie di classi. A partire da queste, si possono costruire diversi tipi di dati. Un'applicazione che riferisce gerarchie di classi, introduce astrazioni appositamente concepite per il suo ambiente.

I progettisti di applicazioni multimediali interagiscono con un complesso ed incerto ambiente: complesso a causa della varietà dei media e dell'equipaggiamento da supporto; incerto a causa dell'evoluzione degli *standards* e dei mutamenti delle prestazioni dell'*hardware* che sono richiesti dalle continue esigenze di nuovi tipi di applicazioni. Le caratteristiche delle applicazioni multimediali devono rispondere ai seguenti requisiti: devono essere distribuite, eterogenee, soddisfare le esigenze di una svariata utenza, richiedere risposte in tempo reale e gestire flussi di dati multipli. E' importante identificare delle astrazioni generali, che si trovano nelle applicazioni multimediali ed includerle in un *"framework"*, il quale rappresenta un insieme estendibile di classi correlate che fornisce le funzioni di base ed i relativi meccanismi di composizione.

Le proprietà principali caratterizzanti un linguaggio di programmazione ad oggetti sono tre:

- ***Incapsulazione***: è il meccanismo dell'unione dei *records* con le procedure e le funzioni che servono a gestirli, in modo da creare un tipo di dati, l'oggetto.
- ***Ereditarietà***: la definizione di un oggetto (antenato) ed il suo utilizzo per creare gerarchicamente altri oggetti (discendenti) che acquisiscono il patrimonio di dati ed azioni dell'antenato.

• ***Polimorfismo***: ad una azione viene assegnato un determinato nome che viene condiviso da tutta una gerarchia di oggetti definita; ogni oggetto implementa poi l'azione nel modo più appropriato.

Grazie a queste caratteristiche, i linguaggi orientati agli oggetti offrono un maggior livello di astrazione e, soprattutto, un codice più facilmente riutilizzabile e manipolabile, vantaggio questo particolarmente importante quando si debbano sviluppare applicazioni complesse.

Si definisce metodo, una procedura, oppure una funzione, intimamente legata al tipo oggetto a cui si applica. Un metodo può appartenere ad una classe, in maniera esclusiva, oppure può essere condiviso da diverse classi che sfruttino uno stesso tipo di operazione. Le gerarchie di classi offrono la possibilità di parallelismo, attraverso l'esecuzione contemporanea di metodi appartenenti a classi diverse.

4.1 Esempi di oggetti media

E' stato esaminato che, tramite la programmazione orientata agli oggetti, si possono descrivere i dispositivi utilizzati in un'applicazione multimediale, attraverso l'incapsulamento delle loro funzionalità. Un esempio concreto è fornito dal C++, per mezzo del quale s'illustrano, di seguito, alcune implementazioni d'astrazioni precedentemente definite nel corpo di una data applicazione.

Volendo descrivere l'interfaccia ad un lettore di videodisco, si può specificare una classe di rappresentazione nel seguente modo:

```
class VideoDiscPlayer {
    public:
    bool IsCAV( );
    void ForwardPlay ( );
    void ForwardPlay (int speed);
    void ReversePlay ( );
    void ReversePlay (int speed);
    void Still ( );
    void Search (int frame);
    };
```

Invece, la seguente classe specifica l'interfaccia al dispositivo utilizzato per visualizzare un segnale relativo ad un'immagine sul *monitor*:

```
class VideoOverlay {
    public:
    void SetFramePosition (int x, int y);
    void SetFrameSize (int w, int h);
    void SetChromaKey (int rmin, int rmax,
                       int gmin, ...);
    void SetBrightness (int k);
    void FreezeFrame ( );
    .....
    .....
    .....
    };
```

VideoDiscPlayer e **VideoOverlay** sono oggetti media. Ciascuno di questi elabora uno o più flussi media che possono essere, ad esempio, un segnale analogico, oppure una sequenza di dati sincronizzati.

Gli oggetti media si suddividono in tre categorie: quelli che producono, quelli che consumano, e quelli che trasformano i flussi media.

VideoDiscPlayer è un produttore, in quanto costituisce una sorgente di un segnale video analogico, mentre **VideoOverlay** è un consumatore. Un esempio di trasformatore è dato dalla classe **AudioMix** che miscela due o più sequenze audio in una singola.

L'implementazione di un oggetto media incapsula informazioni specifiche sui dispositivi, quindi si viene a creare una situazione di trasparenza con l'utente. Ad esempio, il metodo **VideoDiscPlayer::ForwardPlay**, può inviare un comando su una linea seriale ad un lettore di videodisco.

Si tenga presente che una classe di C++ specifica un'interfaccia per oggetti media simili, mentre un'istanza C++ fornisce l'interfaccia di un particolare oggetto media. Quindi, per quanto possa, ad esempio, esserci soltanto un lettore fisico di videodisco definito attraverso un oggetto, possono esserci diverse sue istanze effettuate da altrettanti processi.

Un aspetto caratteristico degli oggetti media è quello che sono in grado di operare autonomamente.

Ad esempio, si consideri la classe **AudioMix** definita come:

```
class AudioMix{
    public:
    void SetGain (int channel, int gain);
    void Mute (int channel);
    .....
    .....
    .....
    };
```

La classe **AudioMix** contiene il metodo **SetGain**, il cui uso influenza l'operazione di un *mixer* audio, o comunque, di un generico *mixer*, il quale, una volta attivato, sarà in grado di procedere senza l'intervento diretto dell'utilizzatore. Nella programmazione orientata agli oggetti, il *mixer* può essere chiamato oggetto attivo, ovvero si tratta di un oggetto che è nelle condizioni di eseguire *"spontaneamente"* azioni, anche in assenza di un metodo che li invoca. Un ambiente che supporta oggetti attivi, semplifica l'implementazione degli oggetti media. Ad esempio, **AudioMix** può essere implementato utilizzando un dispositivo *hardware*, oppure un processo *software*. In una gestione più avanzata, risulta particolarmente utile un supporto agli oggetti attivi.

In seguito, verranno approfonditi i concetti di *framework* e di media attivi, attraverso una loro definizione formale, allo scopo di estendere le potenzialità della programmazione orientata al trattamento degli oggetti multimediali, formando, così, il più completo livello *framework* che, con maggiore efficienza, riesca a supportare la gestione degli eventi media.

4.2 Introduzione ai media attivi

L'attuale tecnologia, consente, come già osservato, la codifica, la memorizzazione ed il recupero degli eventi multimediali in forma digitale, sui quali possono essere effettuate delle operazioni; queste si considerano come delle azioni indipendenti.

Quando gli oggetti multimediali vengono considerati semplicemente come dati che devono essere elaborati da funzioni già programmate, allora vengono chiamati oggetti passivi, in quanto in essi non è incorporato alcun processo che implementi operazioni di manipolazione dei dati.

Verrà in seguito esaminato il comportamento dinamico degli oggetti multimediali. Si associeranno direttamente i processi a tali oggetti dando loro il significato di oggetti attivi. Ciascuno di questi, interpreta i propri dati multimediali e coopera con gli altri scambiando messaggi. Questi oggetti vengono allora riferiti come media attivi poiché sono in grado di rispondere alle interazioni con l'utente ed alle modifiche dell'ambiente di lavoro.

Vi sono molte ragioni perché la generalizzazione del trattamento degli eventi media venga contraddistinta col nome di **media attivi**. Innanzitutto, l'equipaggiamento utilizzato è basato su tecnologie digitali sempre più avanzate e conseguentemente questo diventa sempre più programmabile. Fino a poco tempo fa, ci si limitava alla sola interconnessione *hardware* della *"strumentazione"* multimediale; invece, adesso si è nelle condizioni di poter progettare *software* in grado di controllare i componenti e ciò gioca un ruolo importante tanto nella produzione, quanto nella trasformazione dei dati multimediali. Si deve entrare nell'ottica che i processi *software*, rivolti al trattamento degli eventi media, vengano considerati come parte integrale dell'ambiente multimediale, piuttosto che considerarli dei meccanismi a sé stanti.

Una seconda ragione, per la quale si considerano i media attivi, è dovuta al fatto che questi risultano gli strumenti *software* più adatti per poter programmare anche le più complesse interfaccie utente, nell'ambito delle applicazioni multimediali.

Un livello *framework*, orientato al trattamento degli oggetti, offre diversi vantaggi. Primo, gli oggetti sono usati per produrre, consumare e trasformare diversi eventi media. In tal senso, gli oggetti incapsulano le caratteristiche dei dispositivi e quindi, attraverso dei meccanismi *software*, vengono mascherati i funzionamenti *hardware*. Secondo, un livello *framework* può essere continuamente esteso, tramite l'aggiunta di nuovi dati, o dispositivi multimediali. Infine, siccome i dispositivi multimediali possono essere coinvolti in diverse forme di interazioni, di connessioni e di miscelazioni, si è definito un modo che consente di ottenere l'implementazione delle funzionalità a basso livello, sfruttando le proprietà della programmazione orientata al trattamento degli oggetti, attraverso l'uso di costrutti ad alto livello.

4.3 Definizione formale dei media attivi

Il punto di partenza è l'uso dei tipi di dati per caratterizzare le informazioni multimediali. Ad esempio, il tipo di dato multimediale CD_{AUDIO} potrebbe consistere in tutti i possibili segnali audio codificati su un *compact disc*, ovvero i suoi valori possono essere sequenze della forma: $campione_i$, dove la rappresentazione dei *bits* utilizzata dai campioni è specificata dal formato di registrazione del CD. Come avviene con gli altri tipi di dati, si possono definire operazioni per CD_{AUDIO}, ad esempio: compressione, filtraggio, etc. Questo suggerisce la seguente definizione per i valori dei dati multimediali.

Definizione: Un valore multimediale v, relativo ad un tipo di dati D, consiste in una sequenza finita di elementi d_i, dove la codifica e l'interpretazione di questi sono governati da D.

In particolare, questo tipo dati, determina come la rappresentazione di v possa essere ottenuta dai d_i. Tale rappresentazione avviene ad una cadenza r_D, ossia la frequenza dati di D. Questo tasso indica il numero dei valori sequenza rappresentati per secondo.

Naturalmente, la frequenza dati, da ora in poi indicata con r_D, varia in base al tipo di dati. Ad esempio, il CD audio richiede 44100 campioni per secondo, mentre, una normale animazione necessita, almeno, di 8 *frames* per secondo. In alcuni casi la frequenza può essere semplicemente una ideale e quindi la rappresentazione avviene ad una diversa (oppure variabile) frequenza con probabile perdita di qualità. Inoltre, vi possono essere tipi di dati dove i d_i contengano esplicite informazioni di sincronizzazione. Il valore da assegnare ad r_D è uguale alla massima frequenza di dati che intercorre tra due rappresentazioni successive di valori multimediali.

Un costrutto appropriato di un linguaggio di programmazione è fornito dall'oggetto attivo.

Gli oggetti attivi possiedono uno stato (*istanza di variabile*) ed un comportamento (*metodo*). Inoltre, ciascun oggetto attivo è associato ad un processo che può essere eseguito anche se non vi sono messaggi inviati all'oggetto. Vi sono molti modi nei quali possono essere aggiunti, ad un linguaggio di programmazione orientato al trattamento degli oggetti, i costrutti per la gestione della concorrenza, in relazione al fatto di supportare oggetti attivi. Qui, si assume un semplice model-

lo dove gli oggetti attivi siano composti da diversi processi e la sincronizzazione sia responsabilità del programmatore.

Utilizzando gli oggetti attivi si possono, quindi, definire oggetti multimediali come segue:

Definizione: Un oggetto multimediale è un oggetto attivo che produce e/o che consuma valori multimediali (di tipi specificati) ad una determinata frequenza dati.

Si osservi che gli oggetti multimediali sono soggetti a vincoli di tempo reale per quanto riguarda la produzione, oppure il consumo di dati. Inoltre, ciascun oggetto multimediale, può essere visto come formato da una collezione di porte. Una porta ha un tipo di dati (multimediale) ed è usata per l'*input*, oppure per l'*output*. Le caratteristiche delle porte di un oggetto multimediale (il numero, il loro tipo di dati e la direzione), possono essere determinate attraverso le classi dell'oggetto. Quindi, gli oggetti multimediali, si possono suddividere in tre categorie: **sorgenti**, **pozzi** e **trasformatori**. Una sorgente produce valori multimediali, un pozzo li consuma, mentre, un trasformatore li produce e li consuma.

E' conveniente utilizzare una notazione grafica per rappresentare gli oggetti multimediali; questi vengono contraddistinti con dei cerchi (si veda figura 9), invece i connettori indicano le loro porte (in fuori per le porte di *output*, in dentro per le porte di *input*).

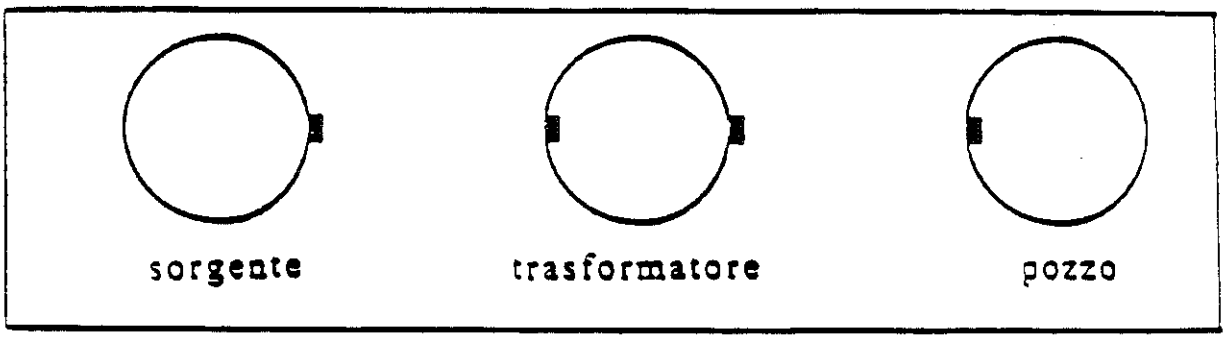


Figura 9

4.4 Composizione temporale

La motivazione per la scelta dell'uso della composizione temporale, proviene dalla necessità di modellare situazioni in cui un certo numero di componenti multimediali vengano rappresentati simultaneamente.

Nell'ambito del *framework* orientato al trattamento degli oggetti già definito, si formeranno multimedia composti attraverso l'uso di oggetti speciali definiti come segue:

Definizione: Un oggetto multimediale composto contiene una collezione di componenti oggetti multimediali ed una specificazione delle loro relazioni temporali e di configurazione.

I due gruppi di relazioni, specificati tramite un oggetto multimediale composto, sono utilizzati per diversi scopi. In particolare:

- le relazioni temporali indicano la sincronizzazione e la sequenza temporale dei componenti;
- le relazioni di configurazione indicano le connessioni tra porte di *input* e di *output* dei componenti.

Come esempio, si supponga di voler costruire un multimedia composto, c_1, che, quando rappresentato, si comporti come segue:

- Attivi al tempo t_0 un oggetto audio ($Audio_1$) ed un oggetto video ($Video_1$).
- Al tempo t_1, trasformi una tonalità di colore partendo dall'oggetto video ($Video_1$), verso un secondo oggetto video ($Video_2$).
- La transizione è completata al tempo t_2, mentre, al tempo t_3 vengono fermati sia $Audio_1$, che $Video_2$.

In questo caso $Audio_1$, $Video_1$ e $Video_2$, sono degli oggetti sorgenti. Per completare il composto, occorrono tre oggetti addizionali: un pozzo audio ($AudioOut$), un pozzo video ($VideoOut$) ed un trasformatore DVE (dve_1), per l'esecuzione del trasformazione della tonalità di colore da $Video_1$ a $Video_2$.

Le relazioni temporali di un oggetto composto, sono facilmente descrivibili tramite un diagramma di sequenze composte dei tempi. Un tale tipo di diagramma contiene una sequenza dei tempi per ciascuna porta di *output* inerente al composto. Il diagramma per il composto c_1 è mostrato in figura 10.

La rappresentazione della sequenza dei tempi mostra la concorrenza dentro un composto, dovuta alla sovrapposizione di un numero

di canali multimediali (ciascuna barra orizzontale nel diagramma della sequenza dei tempi), inoltre identifica dei punti di transizione come gli istanti di tempo in cui le sorgenti partono, oppure si fermano.

Per il composto c_1, vi sono 4 canali e 4 punti di transizione: t_0, t_1, t_2, t_3.

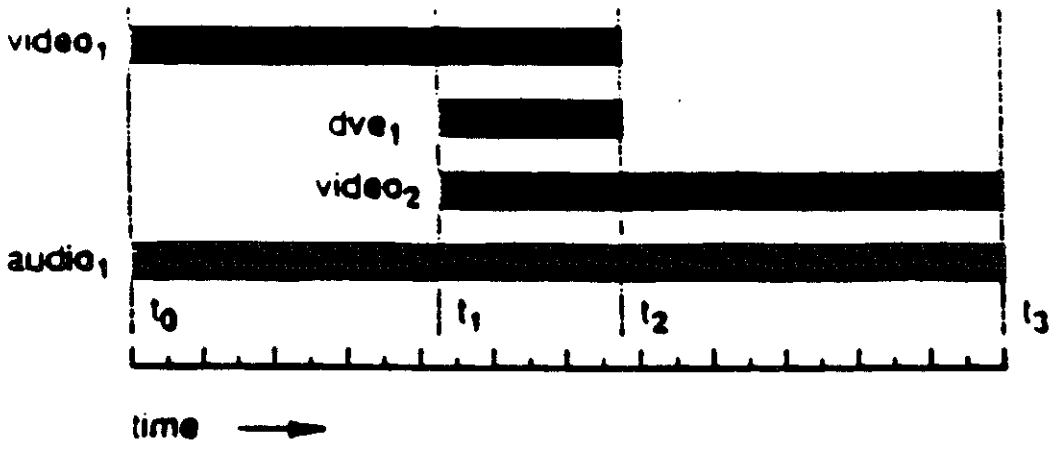


Figura 10

Le relazioni, inerenti alla configurazione di un composto, specificano per ciascuno di tali intervalli, le connessioni tra le porte di *input* e di *output* dei suoi componenti. Questa informazione può essere rappresentata con una rete di componenti, dove i nodi corrispondono ai componenti di un multimedia composto, mentre gli archi corrispondono alle connessioni delle porte. La rete di componenti per c_1, durante l'intervallo $[t_1,t_2]$, è mostrata in figura 11.

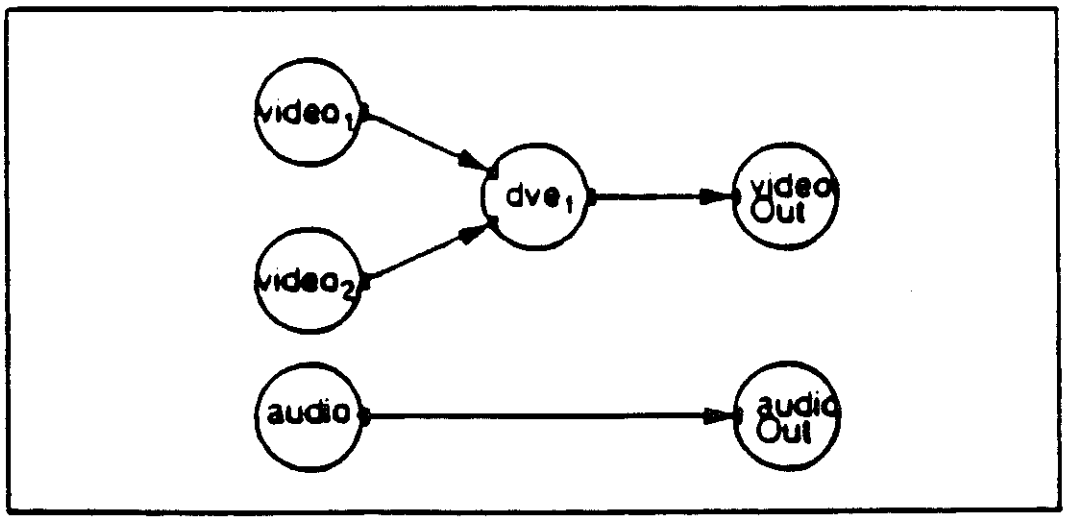


Figura 11

5 I DATI AUDIO/VIDEO

Sono state sviluppate diverse tecnologie per le quali si è ritenuto necessario l'uso dei dati digitali audio e video. L'esistenza di questo tipo di informazioni, comunemente chiamati **dati A V** (*Audio/Video*), ha creato la necessità di una loro gestione attraverso l'uso di *sistemi di basi di dati*, in quanto questi offrono grandi capacità di memorizzazione, possibilità di far riferire un singolo segmento di dati AV contemporaneamente a più applicazioni ed opportunità di recupero di entità, come frammenti audio e video, grafica, immagini, sequenze animate, tramite il metodo dell'interrogazione.

5.1 I sistemi *databases* AV

Un valore AV è costituito da una sequenza di elementi relativi a valori digitali audio, oppure video. Ciascuno di questi valori viene presentato ad una certa *frequenza dati*, la cui determinazione risulta critica in quanto subordinata alle esigenze d'elaborazione.

Esempi di valori AV includono CD audio codificato, sequenze MIDI, video digitale CCIR-601 e vari formati compressi di immagini digitali. Un sistema *database* AV è un'entità *software/hardware* capace di memorizzare un grande numero di valori AV, controlla gli accessi concorrenti e fornisce le risposte in tempo reale alle applicazioni.

Affinché possa essere effettuata la costruzione basilare dei blocchi di dati AV, s'introducono, di seguito, i seguenti concetti.

• *Tipi dati ed Elementi dati.* Un tipo di dati governa la codifica e l'interpretazione di elementi dati; esempi di questi includono i familiari tipi *integer* e *float*. Ciascun elemento dati ha un'ampiezza che può essere intesa come la sua lunghezza in *bytes*. Se tutti gli elementi di un tipo di dati T sono della stessa ampiezza, allora si può dire che T ha elementi ad ampiezza fissata.

• *Tipi AV ed Elementi AV.* Certi tipi di dati sono utilizzati per codificare audio digitale, oppure video digitale; esempi di questi includono i nuovi tipi: CD_{AUDIO} e $JPEG_{VIDEO}$. Un elemento AV, appartiene ad un tipo AV. Per i tipi appena menzionati, gli elementi devono essere, rispettivamente, un singolo campione di CD audio, oppure un singolo *frame* di JPEG video.

5.2 Il tempo ed i dati AV

I dati AV sono entità numeriche dipendenti dal tempo e ciò determina due importanti implicazioni.

1) La manipolazione (recupero, memorizzazione ed elaborazione) dei dati AV è soggetta a vincoli di *tempo reale*.

2) Possono esserci *correlazioni temporali* tra i dati AV.

Quello che è importante, per la modellazione dei dati AV, è l'abilità di specificare le correlazioni temporali. In altre parole, un modello di dati AV deve determinare la sincronizzazione dei dati AV stessi. Si considerano, allora, i seguenti sistemi di riferimento temporali:

- *Valore in tempo continuo*. Un valore in tempo continuo consiste, semplicemente, in un tempo misurato usando alcune convenzioni sulle unità.
- *Sistema coordinato in tempo discreto*. Un tale sistema D è formato dal *mappaggio* dai valori in tempo discreto, al tempo continuo. Il mappaggio è della forma $D_i = (1/f)_i$, dove f rappresenta la *frequenza* del sistema coordinato. Vengono indicati sistemi di tempo specifici tramite D_f; alcuni esempi sono D_{30} per immagini nello standard *nordamericano*, D_{25} per immagini nello standard *europeo*. D_{44100} per CD audio e D_{24} per *films*.
- *Valore temporale*. Un valore temporale è una sequenza finita di *t-uple* della forma: $<e_i, s_i, d_i>$, $i = 1,...,n$. Ciascun valore temporale è basato su un tipo di dati T ed un sistema coordinato in tempo discreto D. In particolare, gli e_i sono elementi dati di tipo T, gli s_i ed i d_i sono valori che si riferiscono al sistema D. Il valore s_i è chiamato *tempo di partenza di* e_i, mentre, d_i rappresenta la sua *durata*. I tempi di partenza e le durate devono soddisfare le relazioni:

$$s_{i+1} \geq s_i + d_i,\ s_{i+1} > s_i \text{ e } d_i \geq 0.$$

Possono essere identificate varie forme speciali di valori temporali, come:

- *valore AV*: T è un tipo AV;
- *valore pienamente coperto*: $s_{i+1} - s_i = d_i$ per $i = 1,...,n-1$;
- *valore uniforme*: il valore è pienamente coperto e $d_i = 1$ per $i = 1,...,n-1$;
- *valore di frequenza costante*: il valore è uniforme e gli elementi hanno la stessa dimensione.

Ad esempio, una sequenza CD audio ha valore di frequenza costante, gli elementi hanno la stessa ampiezza e si presentano uniformemente. Una codifica JPEG, con *frames* immagini NTSC, è uniforme in quanto i *frames* si presentano ogni trentesimo di secondo, ma non hanno frequenza costante poiché le loro codifiche variano in ampiezza. Le rappresentazioni video che evitano la reiterazione di *frames* identici, sono pienamente coperti, ma non uniformi. Infine, le sequenze MIDI, i cui elementi dati corrispondono ad eventi musicali, non sono pienamente coperti.

5.3 Le astrazioni AV

Vengono definite diverse astrazioni che forniscono descrizioni a più alto livello dei valori AV.

• *Fattori di qualità*. Gli algoritmi di compressione dei dati immagine come JPEG, MPEG e quelli usati in DVI, sono soggetti a perdita di informazioni, ovvero la decodifica non genera esattamente l'immagine originaria. Tale perdita di informazioni è intesa come una riduzione della qualità dell'immagine. L'ammontare di informazioni perdute, può essere controllato dalla selezione dei parametri utilizzati durante la codifica. Questi possono non essere visibili al livello della modellazione dei dati, invece, la qualità delle immagini può essere specificata attraverso la descrizione dei *fattori di qualità*.

Un fattore di qualità video consiste in un'espressione della forma: $w*h*d$, che indica una risoluzione video di banda w, altezza di h *pixels* ed una profondità di d *bits* per *pixel*.

• *Tipo entità temporale*. Questo tipo è specificato da un'espressione della forma $N[T, \{Q\}, \{D\}]$, dove N è il nome del tipo entità, T è il suo tipo di dati di base, Q è un fattore di qualità opzionale e D è il suo sistema coordinato in tempo discreto. Non è necessario specificare quest'ultimo se:

1) i valori temporali basati su T usano lo stesso sistema di tempo, (come avviene per i valori del tipo CD_{AUDIO} che usano sempre D_{44100});

2) sono consentite istanze differenti di N per riferirsi a diversi sistemi di tempo.

Un'istanza di un tipo di entità temporale, specificato come sopra, è un valore temporale basato sul tipo di dati T e si riferisce, se presente, al sistema coordinato in tempo discreto D.

Infine, se T è un tipo AV, allora N è un'entità di tipo AV; esempi sono:

- *VideoEntity* [$JPEG_{VIDEO}Type$, 480*640*8, D_{30}];
- *VoiceEntity* [$PCM_{AUDIO}Type$, D_{16000}];
- *MusicEntity* [$CD_{AUDIO}Type$];
- *MIDIEntity* [*MIDIType*];
- *AudioEnvelopeEntity* [*IntegerType*, D_{2205}].

Ad esempio, istanze di *VideoEntity*, sono sequenze di *frames*, codificati JPEG, di buona qualità e disponibili ad una frequenza di 30 *frames* per secondo.

• *Relazioni di derivazione*. Si presentano un numero di relazioni tra entità temporali. Sono di particolare interesse quelle che indicano come un'entità possa essere derivata da un'altra. Così come avviene per le altre relazioni, quelle di derivazione possono essere rappresentate come istanze di tipi di relazione, dove gli attributi del tipo sono parametri che governano la derivazione. Tipi relazione di derivazione che si rivelano utili, includono: *Translation* (gli istanti di partenza per un valore sono traslati uniformemente), *Concatenation* (viene aggiunto un valore alla fine di un altro), *Selection* (viene estratto un *"sottovalore"*) e *Conversion* (viene ricampionato un valore; il caso più semplice è quello di scambiare un valore basato su D_{2n}, con uno basato su D_n, ovvero viene dimezzata la frequenza di campionamento). Generalmente sussistono vincoli sui tipi di entità riferiti dalle relazioni di derivazione. Ad esempio, una *VoiceEntity* può essere concatenata ad una *VideoEntity*, oppure può essere ottenuta da una *VideoEntity* tramite una conversione.

Le relazioni di derivazione sono essenziali per i *database* AV per diverse ragioni. Si riducono i costi di memorizzazione, consentendo modi alternativi per la costruzione dei dati, senza che sia necessario ricorrere ad una loro replicazione. Le operazioni di modifica possono essere eseguite più efficientemente. Ad esempio, per cancellare una sottosequenza di immagini, si dovrebbero copiare e riassemblare i dati *frames*, invece, risulta più efficiente alterare semplicemente una relazione di derivazione. Infine, queste forniscono l'indipendenza dei dati fisici attraverso la separazione dai valori derivati dai dati di base memorizzati. Inoltre, si può osservare che i valori derivati possono essere convertiti, se necessario, in valori non derivati. Ad esempio, un programma di gestione del CD, esegue una conversione in modo tale che i dati vengano opportunamente disposti sul disco.

• *Composizione temporale.* Questa speciale relazione è usata per rappresentare gruppi di entità correlate temporalmente. Alcuni gruppi richiedono rappresentazioni sincronizzate e stanno alla base del multimedia; ad esempio sia la televisione che i *films* consistono di valori audio e video correlati temporalmente. Un'entità semplice, è un'entità temporale (derivata, o non derivata). Un'entità composta, consiste in un gruppo di entità riferite tramite una composizione temporale. Il gruppo può contenere sia entità semplici, che altre entità composte.

• *Diagramma dati AV.* Questo è un grafo diretto, dove i *nodi* corrispondono ad entità temporali (semplici, o composte) e gli *archi* corrispondono alle relazioni di derivazione ed alle relazioni di composizione temporale.

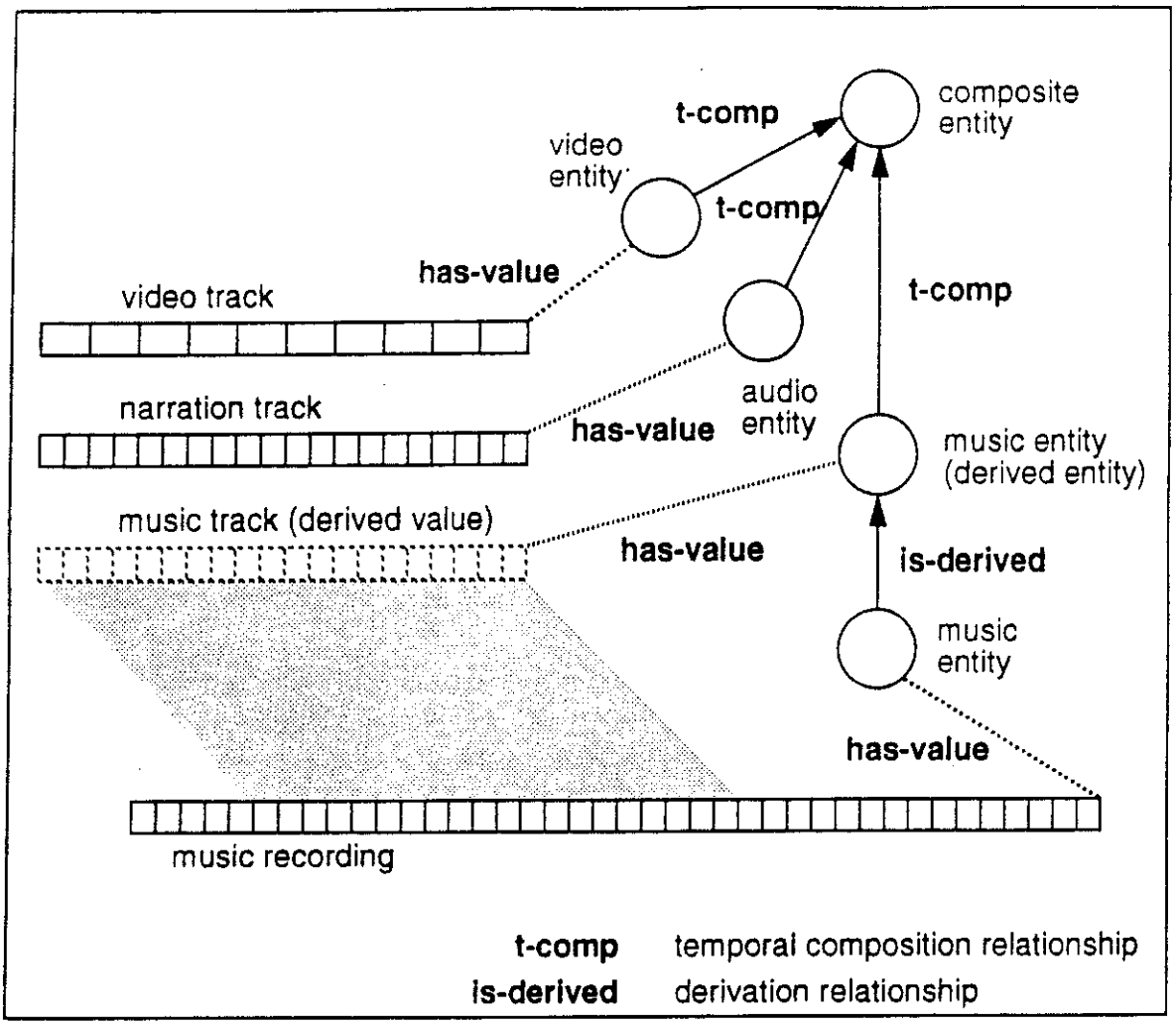


Figura 12

Ad esempio, si consideri un'entità composta formata da due traccie audio (una per la musica, l'altra per la voce) ed una traccia video. La voce viene registrata appositamente, mentre la traccia musicale è derivata da una registrazione preesistente. Il diagramma dati AV, per questo esempio, è mostrato in figura 12; qui, vengono anche indicati i valori entità tramite i *nodi rettangolari* (collegati, con una *linea tratteggiata*, alle corrispondenti entità).

5.4 Le operazioni AV

I dati possono essere visti come se avessero due stati. Il primo è uno **stato passivo**, dove i dati sono memorizzati in qualche formato e gestibili per operazioni di modifica e d'interrogazione. Il secondo, di maggiore interesse, considera i dati AV in uno **stato attivo**; in questa forma è meglio intenderli come delle stringhe. In tal caso, ad esempio, si può trattare di un valore di frequenza associato ai dati e quindi le operazioni su di essi dovranno essere effettuate in base a tale valore.

Alcune operazioni possono essere applicate ai dati AV nello stato passivo, però la maggior parte delle volte le operazioni per un *database* convenzionale come inserimento, cancellazione, modifica ed interrogazione di dati falliscono in questa categoria. Ad esempio, se prendendo un oggetto multimediale viene modificato un suo attributo non sottoposto a vincoli di sincronizzazione, allora quest'operazione può essere compiuta in una base di dati AV così come avviene nei sistemi tradizionali.

Altre operazioni, come la registrazione, il riascolto e la rivisione, interagiscono con i dati AV nel loro stato attivo. Per fornire un supporto a tali operazioni, allora i sistemi di *database* AV devono essere in grado di gestire flussi di dati attivi. Questa esigenza fondamentale ha ripercussioni sugli aspetti inerenti la progettazione di un sistema di *database* AV.

Viene, dunque, intrapreso un approccio che consiste nell'inserimento, dentro le applicazioni, di controlli sui flussi di dati AV attivi, per mezzo della creazione e della manipolazione di istanze delle **classi di attività**.

I sistemi di *database* AV richiedono sia la composizione temporale, che la composizione dei flussi di dati attivi. La composizione

temporale determina quando devono avvenire le operazioni sui valori AV e per quanto tempo devono rimanere attivate.

Si può concludere affermando che, un *database* AV può contenere valori AV *composti temporalmente*, i quali vengono elaborati e trasferiti alle applicazioni, attraverso raggruppamenti di attività connesse, ovvero a ciascuna di queste possono essere associati valori AV. Le richieste effettuate dalle applicazioni di creare, attivare, connettere e sospendere attività, vengono mediate dal sistema di *database* che ha la responsabilità di controllare l'accesso alle risorse condivise.

CONCLUSIONE

I calcolatori oggi sono in grado di acquisire e/o generare testi, immagini e suoni, di correlarli opportunamente tramite determinate applicazioni, di stabilire nuove forme di comunicazione e quindi di instaurare dei metodi di dialogo avanzati con gli utenti. Questa situazione di fatto, ovviamente, non costituisce un punto d'arrivo per la ricerca in quanto il continuo progresso tecnologico consente di apportare al mondo dei calcolatori una maggiore *"espressività"* intesa, fra l'altro, come lo sviluppo delle sue *"facoltà interattive"*.

La ricerca ha evidenziato due situazioni di diverso tipo:

- un'ottima facoltà, da parte dell'uomo, di fare proprie le nuove tecnologie, di adattarle ed integrarle il meglio possibile con le attività espressive tradizionali;
- un certo limite delle tecnologie, rispetto le possibilità intraviste.

Questi aspetti evidenziano una certa *"distanza"* tra il comportamento umano e la macchina; tuttavia costituiscono lo stimolo per il progresso continuo del rapporto tra uomo e tecnologie.

Infatti, considerando i limiti degli strumenti tecnologici, è l'esigenza del loro superamento che genera:

- evoluzione dei meccanismi applicativi e d'uso, per *"aggirare"* il problema tecnologico;
- grosse spinte alla ricerca ed all'innovazione tecnologica.

Negli strumenti e tecnologie multimediali vengono considerati fattori come:

- funzionamento dei *computers*;
- meccanismi di applicazione (i diversi gradi di versatilità);
- natura e struttura dei programmi, filosofia del programmatore;
- filosofie d'uso da parte dell'utente;
- culture derivate dall'uso di stumenti informatici:
  - meccanismi di produzione: industrie e servizi;
  - lavoro intellettuale ed artistico;
  - culture visive e sonore;
- limiti delle tecnologie ed ipotesi di superamento.

Questi temi costituiscono i punti principali che vengono presi in considerazione nell'attività di aggiornamento, ritenuta di fondamentale importanza, degli ambienti multimediali, allo scopo di far evolvere

continuamente la metodologia di comunicazione umana attraverso il mondo dei calcolatori sempre più proteso verso lo *user-friendly*.

BIBLIOGRAFIA

1) *UN'ANALISI MULTILIVELLO SU AMBIENTI DI SUPPORTO ORIENTATI AD APPLICAZIONI MULTIMEDIALI*. Attilio Gabriele MANGIAPANE - Tesi di Laurea in Scienze dell'Informazione. Pisa, febbraio 1994.

2) *DIGITAL IMAGE PROCESSING TECHNIQUES*. M. P. EKSTRONG - Academic Press, 1984.

3) *HYPERTEXT AND HYPERMEDIA*. Jakob NIELSEN - Academic Press, Inc. San Diego, 1990.

4) *IBM. INTERNATIONAL TECHNICAL SUPPORT CENTERS. MULTIMEDIA APPLICATION ENABLERS AND THE IBM PS/2 ULTIMEDIA SYSTEMS*. Document GG24 - 3749 February 1992. IBM Corporation, International Technical Support Center. Boca Raton, Florida.

5) *INFORMATICA E MUSICA*. Leonello TARABELLA, Graziano BERTINI, Annalisa CAIOLI, Andrea GUERRA. Jackson Libri. Milano, 1992.

6) *LECTURE NOTES IN COMPUTER SCIENCE (614). NETWORK AND OPERATING SYSTEM SUPPORT FOR DIGITAL AUDIO AND VIDEO. SECOND INTERNATIONAL WORKSHOP*. Heidelberg, Germany, November 1991. Proceedings. R. G. HERTTWICH (Ed.) - Springer-Verlag, Berlin Heidelberg 1992.

7) *OBJECT COMPOSITION*. Edited by D. Tsichritzis. Université de Genève: Centre Universitaire d'Informatique. Genève, juin 1991.

8) *OBJECT FRAMEWORKS*. Edited by D. Tsichritzis Université de Genève: Centre Universitaire d'Informatique. Genève, juillet 1992.

9) *TEORIA DEI SEGNALI. PARTE PRIMA: SEGNALI DETERMINATI*. Lucio VERRAZZANI. ETS Pisa, 1982.

10) *THE OFFICIAL GUIDE TO CD-I DESIGN FROM PHILIPS INTERACTIVE MEDIA SYSTEMS. THE CD-I DESIGN HANDBOOK*. Philips Ims. Addison-Wesley Publishing Company. 1992 Philips Electronics UK Ltd.

11) *AUDIO REVIEW* N. 113, Febbraio 1992. Michele MARANI - *IL FUTURO DELL'AUDIO SI CHIAMA DSP*. Prima parte.

12) *AUDIO REVIEW* N. 117 Giugno 1992. Michele MARANI - *IL FUTURO DELL'AUDIO SI CHIAMA DSP*. Seconda parte.

13) *COMMUNICATIONS OF THE ACM*. April 1991 Vol. 34 No. 4 Kevin HARNEY, Mike KEITH, Gary LAVELLE, Lawrence D. RYAN, Daniel J. STARK - *THE i750 VIDEO PROCESSOR: A TOTAL MULTIMEDIA SOLUTION.*

14) *PIXEL* N. 10, 1993. Luigi FILIPPINI - *MPEG: UNO STANDARD PER LA COMPRESSIONE DI IMMAGINI IN MOVIMENTO.*

15) *PROCEEDINGS OF IEEE*, vol. 61, pp. 692 - 702, June 1973. Ronald W. SCHAFER and Lawrence R. RABINER - *A DIGITAL SIGNAL PROCESSING APPROACH TO INTERPOLATION AND DECIMATION.*