TESI DI DOTTORATO

UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

DIPARTIMENTO DI INGEGNERIA ELETTRONICA
E DELLE TELECOMUNICAZIONI

DOTTORATO DI RICERCA IN
INGEGNERIA ELETTRONICA E DELLE TELECOMUNICAZIONI

CODIFICA DI SEGNALI VISUALI MULTIDIMENSIONALI.

LUCA CICALA

TUTORE E COORDINATORE DEL CORSO DI DOTTORATO
CH.MO PROF. **GIOVANNI POGGI**

A. A. 2006–2007

A Maria Antonietta,
che, con il suo amore
ed il suo immancabile sostegno,
riempie di gioia le pagine
della mia vita.

Ringraziamenti

Il mio dottorato di ricerca è stato finanziato dal CIRA (Centro Italiano Ricerche Aerospaziali), per cui ringrazio in primo luogo l'Azienda, nella persona del Presidente Prof. Sergio Vetrella, per avermi dato l'opportunità di proseguire gli studi con serenità e sostegno economico adeguato.

Ringrazio il mio tutore, Prof. Giovanni Poggi, per la serietà e la dedizione che ha mostrato nel seguirmi, per tutto ciò che ha saputo insegnarmi con le parole e soprattutto con l'esempio, sia professionalmente che umanamente.

Un doveroso ringraziamento va a tutti coloro che, presso il CIRA, mi hanno aiutato a risolvere i piccoli problemi burocratici, finanziari ed organizzativi, che, in diverse occasioni, hanno potuto presentarsi. In particolare ringrazio il mio responsabile di laboratorio, il Dott. Paolo Leoncini, ed il mio dirigente di settore, il Dott. Pasquale Schiano, per essersi sempre fatti carico personalmente degli eventuali problemi ed aver ogni volta provveduto alla loro risoluzione con fermezza e decisione. Grazie al Dott. Carlo Camerlingo, punto di riferimento del CIRA per le attività di formazione, il quale ha dato prova di massima disponibilità e di impeccabile professionalità tutte le volte che ho avuto bisogno di lui.

Voglio ringraziare inoltre tutti i colleghi ricercatori del CIRA di Capua, dell'Università degli Studi "Federico II" di Napoli e di altre istituzioni, con cui ho avuto occasione di collaborare e da cui ho avuto la preziosa opportunità di apprendere: Marco Cagnazzo, Carlo Di Benedetto, Paolo Leoncini, Francesco Martone, Marco De Mizio, Giuseppe Scarpa, Lorenzo Sorgi, Luisa Verdoliva, Andrea Zinicola.

Un sincero ringraziamento agli amici Angelo ed Ileana, per il loro affetto ed il loro sostegno, soprattutto nel periodo della stesura della tesi.

Un grazie di cuore alla famiglia Gimelli, per la disponibilità e la generosità d'animo dimostrata in innumerevoli occasioni. Se in questi anni ho potuto dedicarmi con serenità agli studi è stato anche merito vostro.

Infine un grazie alla mia compagna di vita ed insostituibile amica, Maria

Antonietta, che anche in questa circostanza, come in tante altre, ha saputo rendersi preziosa.

Grazie a tutti!

Luca Cicala

Indice

Avvertenza

Questo testo, la cui stesura si è conclusa il 30-11-2007, nasce come lavoro di tesi di dottorato. Tuttavia la sua struttura originaria verrà aggiornata e riadattata nei prossimi mesi, in modo da poter includere gli ultimi sviluppi di ricerca. Chiunque sia interessato può trovare il testo definitivo di questo lavoro sul sito del gruppo di ricerca del Prof. Giovanni Poggi.

Luca Cicala

Introduzione

I dati visuali multidimensionali sono collezioni di immagini altamente correlate, ad esempio temporalmente (sequenze video) o spettralmente (immagini multicanale). In questo lavoro di tesi sono affrontati due approcci di codifica con perdita di informazione: mediante analisi multidimensionale multirisoluzione e mediante classificazione.

In particolare, per la prima famiglia di soluzioni, si propone una generalizzazione di una nota famiglia di tecniche di quantizzazione e codifica, gli *zerotree coder*, a cui appartengono i celebri codificatori wavelet EZW e SPIHT. Questi codificatori si basano su strategie alternative a quelle dello standard per la codifica wavelet di immagini fisse, JPEG2000, e, per la loro semplicità e la loro efficacia, sono diventati un importante *benchmark* nell'ambito della codifica wavelet e multirisoluzione. Nonostante questi codificatori siano ampiamente impiegati, in diverse varianti, nelle applicazioni, una completa formalizzazione e generalizzazione degli stessi non è mai stata affrontata. In questo testo diverse nozioni relative agli *zerotree coder* sono raccolte ed organizzate in maniera sistematica. Inoltre vengono aggiunti alcuni strumenti innovativi di interpretazione e di progetto, che permettono in definitiva di modellare gli *zerotree coder* come automi a stati finiti. Infine, si analizzano sperimentalmente diverse configurazioni, allo scopo di mostrare quali scelte progettuali garantiscono più funzionalità e migliori prestazioni tasso-distorsione. Gli esperimenti presentati sono organizzati ordinatamente, riguardano diversi tipi di dato e di applicazione, diversi tipi di trasformata, diversi aspetti progettuali. Alcune direttive progettuali suggerite dalla letteratura scientifica vengono confermate e nuovi criteri vengono esposti.

Per quel che riguarda la *codifica mediante classificazione*, se ne esamina il paradigma generale ed in particolare si articolano considerazioni sulle *trasformate classificate*, adattate cioè alle statistiche della particolare classe di dati. Il problema della codifica classificata è trattato sotto l'aspetto dell'efficienza di compressione piuttosto che sotto l'aspetto delle funzionalità. Si propone un

approccio alla classificazione e al progetto della trasformata di tipo non supervisionato, valutandone accuratamente la fattibilità, la complessità di calcolo e le prestazioni tasso-distorsione.

Sia per il primo che per il secondo argomento, un'attenzione particolare, nel corso della trattazione, è rivolta alla codifica di immagini telerilevate e alle applicazioni nel settore aerospaziale.

Il testo è organizzato in due parti e sette capitoli.

La prima parte, costituita da tre capitoli, è una raccolta di nozioni propedeutiche che mirano a consentire a chi lavora nel settore della scienza e dell'ingegneria dell'informazione, ma non si occupa direttamente di codifica, di dotarsi dei concetti fondamentali ed dei riferimenti bibliografici necessari a comprendere il resto della trattazione.

La seconda parte, di quattro capitoli, è quella che raccoglie i contenuti innovativi. Il cap. 4 ed il cap. 5 in particolare raccolgono rispettivamente le considerazioni teoriche ed i risultati sperimentali sugli *zerotree coder*. Il cap. 6 invece introduce il lettore alla problematica della *codifica mediante classificazione*, raccogliendo concetti e risultati fondamentali dalla recente letteratura scientifica. Il cap. 7 è il cuore del contributo innovativo apportato da questo lavoro alla tematica della codifica classificata, in particolare nel dominio applicativo della codifica di immagini multispettrali telerilevate. L'enfasi è sul progetto dello stadio di segmentazione e di trasformata spettrale orientata alle classi, effettuata nello schema proposto direttamente sui dati da codificare.

Infine, in un paragrafo conclusivo, si accenna alle possibili applicazioni e sono presentati gli eventuali sviluppi futuri.

Parte I

In questa parte del lavoro introduciamo le nozioni fondamentali utili a capire il contesto scientifico e applicativo delle soluzioni innovative proposte. La trattazione ha valenza di una ricapitolazione per chi è già in possesso dei concetti illustrati e di una breve introduzione all'argomento per chi proviene da altri settori della scienza e dell'ingegneria dell'informazione. Alcune nozioni sono considerate propedeutiche: la teoria della probabilità e la teoria dei segnali.
Le basi della teoria dell'informazione di Shannon sono accennate nel primo capitolo e, sempre nello stesso capitolo, si espongono le principali tecniche di codifica con e senza perdita di informazione.
Nel secondo capitolo, invece, si fanno alcuni richiami all'analisi di Fourier per poi passare ad analisi tempo-frequenza più generali, come ad esempio la wavelet. Dopo aver affrontato l'analisi di segnali nel continuo si spiega come derivare le corrispondenti metodologie di analisi nel discreto. Infine, si generalizza l'analisi tempo-frequenza al caso multidimensionale.
Nel terzo capitolo si introducono i segnali visuali monodimensionali, come generalizzazione dell'immagine monocromatica. Sono dunque illustrate le tecniche di codifica più diffuse allo stato dell'arte per immagini multicanale e sequenze video.

Capitolo 1

Compressione dati

Parole chiave

Nozioni propedeutiche

Teoria della probabilità, algebra lineare, fondamenti di teoria dei segnali.

Concetti introdotti nel capitolo

Codifica lossless, codifica lossy, compattazione, compressione, informazione, entropia, tasso di codifica, tasso entropico, primo teorema di Shannon, codice non singolare, codice istantaneo, codice univocamente decodificabile, codice a prefisso, codifica di Huffman, codifica aritmetica, codifica di Lempel-Ziv, entropia condizionata, teoria tasso distorsione, MSE, SNR, quantizzazione scalare, companding, algoritmo di Lloyd-Max, allocazione delle risorse, codifica mediante trasformata, coding gain, KLT, quantizzazione vettoriale, GLA, TSVQ.

1.1 Premessa

Vi sono essenzialmente due metodi per ridurre la quantità di dati necessari alla descrizione di un oggetto. Un metodo consiste nell'impiegare strutture dati opportunamente progettate, che siano le meno ingombranti possibili. Con questa tipologia di tecniche non si altera il contenuto di informazione dei dati, ma si cambia soltanto la loro rappresentazione. Parliamo in questo caso di **compattazione**, ovvero di codifica **senza perdita di informazione** (**lossless**) [1][10] [1].
L'altra famiglia di tecniche, invece, considera la capacità di ricezione delle informazioni da parte del loro destinatario che non sempre è capace di elaborarle tutte nella loro complessità. L'occhio umano, ad esempio, non è capace di dare importanza a tutti i particolari di un'immagine. In questi casi, dunque, il dettaglio informazionale è superfluo e può essere tranquillamente eliminato senza danno alcuno nell'ambito del contesto di impiego. Le tecniche basate su questo principio sono dette tecniche di **compressione** o **con perdita di informazione** (**lossy**) [10] [1] [4].

1.2 Codifica senza perdita di informazione

Lo scopo della *compattazione* è quello di ottimizzare la rappresentazione dell'informazione, la cui completezza deve però essere rigorosamente mantenuta presso il destinatario. Le tecniche di *compattazione* si impiegano spesso per i dati sintetici, cioè prodotti dall'uomo, come le istruzioni per un elaboratore numerico o un documento di testo. In seguito riporteremo alcuni risultati fondamentali della *teoria dell'informazione*, di cui cercheremo di dare soprattutto un'interpretazione intuitiva ed illustreremo delle tecniche di *compattazione* di uso comune, alcune delle quali sono impiegate nello schema di riferimento di questo lavoro.
Useremo nella trattazione termini quale alfabeto sorgente, codice, alfabeto di codice, parole codice. L'**alfabeto sorgente** è l'insieme dei simboli su cui opera il **codice**, una funzione matematica che fa corrispondere a ciascuno dei simboli sorgente una stringa di simboli, detta **parola codice**, appartenenti ad un opportuno **alfabeto di codice**. Nel caso della codifica dei segnali digitali i simboli sorgente sono bit o stringhe di bit, dunque sono già codificati nell'alfabeto binario. Su questa prima codifica se ne opera un'altra anch'essa facente capo allo stesso alfabeto di codice con simboli 0 ed 1; si può parlare dunque di **transcodifica**.

1.2.1 La misura dell'informazione

Fin qui, abbiamo parlato di *informazione* riferendoci ad essa in maniera intuitiva. Tuttavia è opportuno, ai fini del nostro discorso, dare una definizione formale di questa grandezza. In analogia a quanto si fa per le grandezze della fisica, definiamo l'*informazione* in maniera operativa, cioè attraverso la sua misura.
Sia X una variabile aleatoria discreta e sia x una sua realizzazione. L'***informazione*** relativa all'occorrenza x è definita come:

$$I(x) = -\log(p(X = x)) \tag{1.1}$$

dove con $p(X = x)$ si è indicata la probabilità dell'evento x. Questa definizione trova la sua giustificazione in alcune proprietà notevoli:

1. l'*informazione* associata all'occorrenza di un evento è tanto più grande quanto meno probabile è l'evento stesso;

2. per due eventi indipendenti l'*informazione* dell'evento congiunto è la somma delle informazioni degli eventi disgiunti.

Se la forma di rappresentazione elementare dei dati è il codice binario, la base del logaritmo si sceglie in genere pari due e l'*informazione* si misura in bit.
Assume importanza l'*informazione* media di una variabile aleatoria discreta. Essa prende il nome di **entropia**:

$$H(X) = -\sum_{i=1}^{M} p(X = x_i) \log(p(X = x_i)) \tag{1.2}$$

dove con x_i si è indicata la generica realizzazione di X.
Si dimostra che la lunghezza media L della parola codice necessaria a codificare X è maggiore o uguale all'*entropia* $H(X)$ espressa nella sua stessa base:

$$L(X) \geq H(X) \tag{1.3}$$

Questa proprietà rivela il significato profondo di questa grandezza, mettendola in relazione all'ingombro medio minimo, in termini di simboli necessari, della rappresentazione di un'oggetto in codifica.
Il concetto di *entropia* si può facilmente generalizzare al caso di variabili congiunte.

$$H(X_1, .., X_N) = -\sum_{i_1=1}^{M_1} ... \sum_{i_2=1}^{M_N} p(x_{i_1}, ..., x_{i_N}) \log(p(x_{i_1}, ..., x_{i_2})) \tag{1.4}$$

oppure ad un processo discreto di variabili aleatorie discrete, cioè ad una successione delle stesse. Indicato con $\{X_k\}$ il processo aleatorio e con x_{ki} la generica realizzazione di X_k, possiamo definire **tasso entropico**:

$$H(\{X_k\}) = -\lim_{N\to\infty} \frac{1}{N} \sum_{i_1=1}^{M_1} ... \sum_{i_N=1}^{M_N} p(x_{i_1}, ..., x_{i_N}) \log(p(x_{i_1}, ..., x_{i_N})) \tag{1.5}$$

Vale tra le grandezze sopra definite l'importante relazione:

$$H(X_1) \geq \frac{1}{N} H(X_1, .., X_N) \geq H(\{X_k\}) \tag{1.6}$$

La prima maggiorazione significa che sequenze congiunte hanno minor contenuto per campione rispetto ad ognuno dei campioni preso singolarmente. E'facile intuire quindi che anche la loro rappresentazione potrà essere più efficiente. Poiché, come abbiamo visto, la *compattazione* non è altro che una transcodifica, conviene dunque scegliere vettori di simboli di sorgente quanto

più lunghi possibile, compatibilmente con il costo computazionale che si paga durante l'elaborazione. Naturalmente la convenienza sussiste qualora vi sia una dipendenza statistica tra i simboli della stringa da codificare; altrimenti la maggiorazione diventa una semplice uguaglianza.
La seconda maggiorazione, non è altro che l'estensione al caso asintotico del discorso precedente: al limite, se si ha una sequenza infinita di simboli, conviene codificarla nel suo insieme. In questo modo si minimizza il contenuto informazionale da rappresentare.

1.2.2 Il limite alla compattazione: il primo teorema di Shannon

Le precedenti definizioni sono necessarie ad introdurre il teorema fondamentale sulla *compattazione* dei dati, il *primo teorema di Shannon*, di cui qui forniamo un'enunciato poco formale ma sufficiente per i nostri fini introduttivi.
Consideriamo la generica sequenza infinita di simboli $\{X_k\}$ emessa dalla sorgente S. Sia ogni simbolo definito sullo stesso alfabeto sorgente $\{x_1, ..., x_M\}$, di cardinalità M. Possiamo dunque definire l'*tasso entropico della sorgente* come:

$$H(S) = -\lim_{N\to\infty} \frac{1}{N} \sum_{i_1=1}^{M} ... \sum_{i_N=1}^{X_s} p(x_{i_1}, ..., x_{i_N}) \log_{X_s}(p(x_{i_1}, ..., x_{i_N})) \quad (1.7)$$

Il **primo teorema di Shannon** afferma che il rapporto asintotico tra la lunghezza della sequenza rielaborata (L) e quello della sequenza originaria (N) non potrà essere mai inferiore all'*entropia* della sorgente.
In sintesi:

$$R(S) = \frac{L(N)}{N} \geq H(S) \quad (1.8)$$

dove R è detto **tasso di codifica**, nel caso binario **bit rate**, ed esprime la misura della *compattazione*. Il teorema dimostra altresì che esiste una strategia di codifica tale che, asintoticamente, il tasso di codifica raggiunge il limite inferiore costituita dal *tasso entropico*.
Concludendo la *compattazione* dei dati ha un limite invalicabile che è legato alle statistiche della sorgente.

1.2.3 I requisiti di validità di un codice

Vediamo in breve quali sono le caratteristiche di una buona transcodifica su sequenze di dati:

non singolarità : esiste una corrispondenza biunivoca tra simbolo originale e parola codice;

univoca decodificabilità : esiste una corrispondenza biunivoca non solo per i singoli simboli ma anche per tutte le possibili sequenze di simboli e le parole codice associate;

istantaneità : una parola codice deve poter essere decodificata nel momento in cui viene letto il suo ultimo simbolo; non deve essere necessario dover leggere anche uno o più simboli della parola codice successiva.

Si osservi come ogni proprietà di questo breve elenco implichi anche le precedenti.
Una famiglia notevole di codici univocamente decodificabili è quella dei **codici a prefisso**. I codici a prefisso si identificano per la proprietà che nessuna delle parole codice è prefisso delle rimanenti.
Si può rappresentare un codice, in maniera intuitiva, come un albero N-ario, dove N è la cardinalità dei simboli impiegati per la rappresentazione (si veda la fig. 1.1). Tale struttura è detta **albero di codifica**. Ogni foglia dell'albero di codifica rappresenta una parola codice. I rami sono etichettati con gli N simboli dell'alfabeto di codice $\{c_1, ..., c_N\}$. La parola codice C_k, con cui è codificato il simbolo sorgente x_k, dunque si ottiene leggendo le etichette dei rami lungo tutto il percorso che va dalla radice fino alla foglia k-sima, passando per L_k livelli dell'albero, dove con L_k si fa riferimento alla lunghezza della parola codice.

$$C_k = \{c_{liv1}, ..., c_{livL_k}\}$$

Un *codice a prefisso* è tale che tutte le parole codice sono foglie dell'albero. Da questa semplice rappresentazione si può evincere come ci sia un'identificazione tra codici a prefisso e codici istantanei.

1.2.4 La codifica di Huffman

Il **codice di Huffman** [5] è un particolare *codice istantaneo* a lunghezza variabile. Nella nostra rappresentazione ad albero significa che non tutte le codeword si trovano necessariamente alla stessa profondità, cioè l'albero non è necessariamente bilanciato.
La *codifica di Huffman* sfrutta le caratteristiche statistiche della sorgente ed in particolare le differenze di probabilità di occorrenza dei simboli da codificare, per minimizzare la lunghezza media della parola di codifica. Essi, in breve,

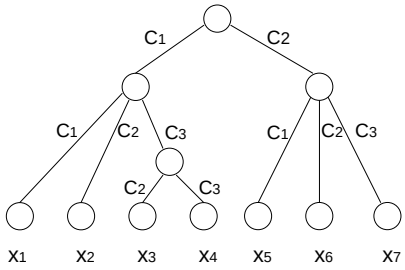


Figura 1.1: Un esempio di codice a prefisso.

assegnano parole corte a simboli molto probabili e parole più lunghe a simboli meno frequenti.

Metodo 1. ***Codificatore di Huffman.***
Data una sorgente costituita da N simboli $\{x_1, ..., x_N\}$ con probabilità decrescente con il generico indice n, possiamo costruire il codice di Huffman costruendo iterativamente un albero binario, osservando i seguenti passi:

1. *si ordinino i simboli in modo che le probabilità degli stessi siano decrescenti e a ciascuno di essi si associ un nodo;*

2. *si considerino dunque i due simboli con minor probabilità e si fondano gli stessi in un unico nuovo simbolo con probabilità pari alla somma delle probabilità dei simboli di partenza;*

3. *al nuovo simbolo si associ un nodo nella struttura ad albero, collegato ai due nodi di partenza mediante una relazione genitore - figli;*

4. *a ciascuno dei due rami provenienti dal nuovo nodo si associno i simboli dell'alfabeto di codifica* 0 *d* 1.

Si iteri dunque fino a quando l'albero di codifica non viene completato.

Un esempio di *codifica di Huffman* in quattro iterazioni è riportato in fig. 1.2. Dentro ciascun nodo dell'albero è riportata la probabilità ad esso relativa, mentre sui rami vi sono i simboli binari dell'alfabeto di codifica. La codifica dei simboli di sorgente è riportata nella seguente tabella:

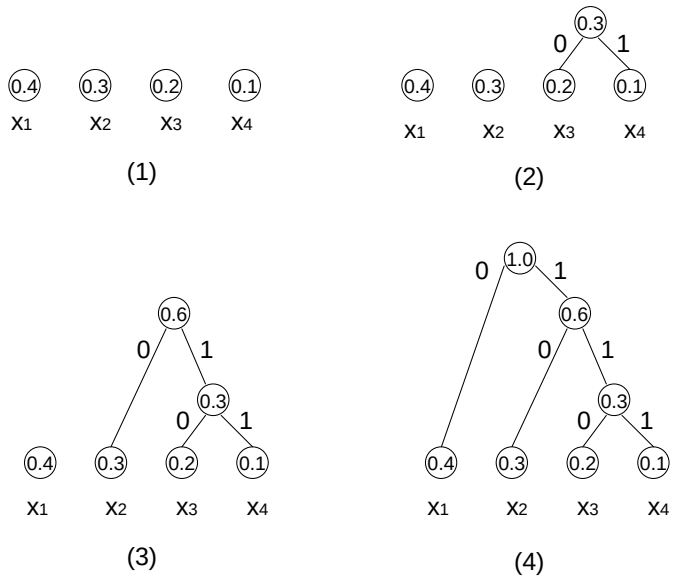


Figura 1.2: Codifica di Huffman.

sorgente:	x_1	x_2	x_3	x_4
probabilità:	0.4	0.3	0.2	0.1
codifica:	0	10	110	111

Tabella 1.1: Codifica di Huffman relativa alla fig.1.2

1.2.5 La codifica aritmetica

Il principale limite della *codifica di Huffman* è che essa assegna parole di lunghezza intera a ciascun simbolo separatmente. Risulta di conseguenza che il bit rate non può essere meno di 1 bit per simbolo a meno che i simboli non vengano codificati congiuntamente. Un altro svantaggio della *codifica di Huffman* è la dipendenza della struttura del codice dalle statistiche della sorgente. Se le statistiche della sorgente cambiano il codice deve essere completamente rioprogettato.
Derivata dalla codifica di Elias la *codifica aritmetica* è subottima, e quindi tecnicamente inferiore a quella di Huffman, ma asintoticamente ottima al crescere del numero di simboli di sorgente codificati congiuntamente. Questo, insieme alla complessità di codice molto ridotta, la rende estremamente interessante.

La ***codifica aritmetica*** [6][7] è una tecnica che non soffre dei limiti della *codifica di Huffman*, in quanto ogni simbolo non deve essere mappato necessariamente in un numero intero di bit. In questo modo può essere raggiunto per un determinato simbolo un numero frazionario di bit rate medio di codifica. In aggiunta il modello statistico della sorgente è visto come un parametro dalla struttura del codificatore, in modo che le statistiche possono essere cambiate dinamicamente senza dover modificare l'algoritmo vero e proprio.
Vediamo ora più nel dettaglio come opera un codificatore aritmetico. Come Huffman, esso associa a stringhe di simboli di sorgente di lunghezza fissa, parole codice di lunghezza variabile. Allo stesso modo inoltre a parole meno probabili assegna parole codice più lunghe.
L'idea di fondo è mappare la sequenza di simboli da codificare in un'unica parola codice. Non è necessario segmentare la sequenza poiché la parola codice può essere determinata ed aggiornata in maniera incrementale. In qualsiasi momento la parola codice elaborata rappresenta univocamente i simboli codificati. Anche se la parola codice finale conta un numero intero di bit può verificarsi per i singoli simboli codificati un'assegnazione media di bit frazionaria.

Metodo 2. ***Codificatore aritmetico.***
Nel generico passo di codifica, allorquando k simboli $\{x_1, ..., x_k\}$ dell'alfabeto sorgente sono stati già codificati, la parola codice è rappresentata da un intervallo semiaperto $[L_k, H_k[\subset [0, 1[$. Qualsiasi numero reale c_k dell'intervallo può rappresentare la codifica dei k simboli fino ad ora processati. Al sopraggiungere del nuovo simbolo x_{k+1}, l'intervallo $[L_k, H_k[$ viene ristretto ad un suo sottointervallo $[L_{k+1}, H_{k+1}[\subset [L_k, H_k[$. Il mensionato sottointervallo viene scelto in maniera che la sua lunghezza sia proporzionale all'occorrenza della nuova sequenza $\{x_1, ..., x_{k+1}\}$. Ne consegue che sequenze meno probabili sono rappresentati da intervalli più corti e quindi necessitano di più bit di precisione nella rappresantazione.

La procedura che abbiamo appena illustrato, presa così com'è, produce un output solo a valle dell'esame dell'intera sequenza da codificare. La codifica può tuttavia avvenire anche in maniera incrementale emettendo di volta in volta quei bit che sono univocamente determinati così come spiegato in [6] e in [7].

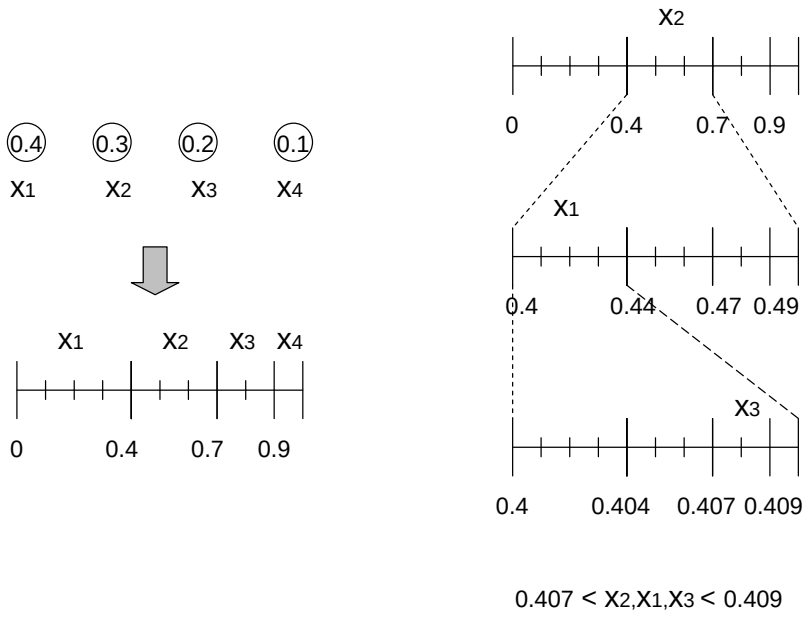


Figura 1.3: Un esempio di *codifica aritmetica*.

1.2.6 La codifica Lempel-Ziv

Mentre i codificatori di Huffman ed i codificatori aritmetici associano a sequenze di simboli di lunghezza fissa parole codice di lunghezza variabile, il **metodo di Lempel-Ziv** [8][9] codifica sequenze di lunghezza variabile mediante parole codice di lunghezza variabile.
Il concetto è quello di estendere l'alfabeto base alle stringhe composte da più simboli. In questo modo si può codificare ciascuno dei nuovi simboli, costituiti da stringhe di simboli dell'alfabeto sorgente, con una singola parola codice. Le prestazioni di questo metodo di codifica sono essenzialmente legate alla dimensione massima del dizionario di stringhe (codebook) e migliorano al crescere di essa. Il limite pratico a queste grandezza è imposto dalla occupazione di memoria del dizionario e dalle elaborazioni necessarie a gestirlo.
Riportiamo in seguito una breve descrizione dell'algoritmo di Lempel-Ziv. Si noti che il dizionario, costituito da coppie indice - parole codice, viene aggiornato in maniera dinamica sia in codifica che in decodifica. All'inizio esso contiene solo i simboli dell'alfabeto base.

Metodo 3. ***Codificatore di Lempel-Ziv.***
Codifica:

1. *Si memorizzi un bit alla volta della sequenza in un buffer, finchè la successione di bit costituisce una parola del dizionario. Quando si è raggiunta la lunghezza massima della parola S_n, nella condizione che essa concatenata al simbolo successivo non sia una parola del dizionario, se ne invii l'indice i_n ad essa relativa.*

2. *Si aggiorni il dizionario con la concatenazione di S_n ed il simbolo x_{n+1} immediatamente successivo.*

3. *Si azzeri il buffer e si memorizzi x_{n+1} nella prima posizione della nuova stringa S_{n+1}.*

Decodifica:

1. *Si legga l'indice i_n relativo alla parola codice S_n da decodificare e si proceda alla decodifica.*

2. *Si aggiunga al dizionario la stringa costituita dalla concatenazione della stringa precedentemente decodificata S_{n-1} e dal primo bit x_n della stringa S_n.*

3. *Si memorizzi S_n al posto di S_{n-1}.*

1.2.7 Sorgenti con memoria

Le tecniche fin qui considerate fanno riferimento a sorgenti senza memoria. Tuttavia quando esiste dipendenza statistica trai simboli è senz'altro conveniente portarla in conto.
Introduciamo le due seguenti definizioni:

$$H(X|Y=y_j) = -\sum_{i=1}^{M_X} p(X=x_i|Y=y_j)\log(p(X=x_i|Y=y_j) \quad (1.9)$$

$$H(X/Y) = \sum_{j=1}^{M_Y} p(Y=y_j)H(X|Y=y_j) \quad (1.10)$$

Di cui la seconda delle quali identifica l'*entropia condizionata*.
Una proprietà notevole dell'*entropia* condizionata è la seguente:

$$H(X|Y) \leq H(X) \quad (1.11)$$

Dove l'eguaglianza vale solo se X ed Y sono indipendenti. Questa relazione si commenta dicendo che i condizionamenti riducono l'incertezza.
Se rileggiamo l'*entropia* come lunghezza media minima di codifica, questa relazione afferma che se ho un'informazione parziale su X (cioè Y) posso codificare X stesso con una stringa più breve. Se ho una sequenza di simboli statisticamente dipendenti posso dunque sfruttare la conoscenza che ho del simbolo attuale (ed eventualmente di quelli precedenti) per migliorare la codifica del simbolo successivo.

Questo principio è alla base dei metodi di **codifica predittiva**. La più semplice tecnica di codifica predittiva è quella che consiste nello stimare il valore del simbolo corrente in base ai valori assunti dai simboli precedenti e nel codificare soltanto la differenza tra il valore reale è quello predetto (**errore di predizione**). Dalla proprietà fondamentale dell'*entropia* condizionata si deduce facilmente per ricorsione che

$$H(X_n|X_{n-1},...,X_{n-k}) \leq H(X_n|X_{n-1},...,X_{n-(k+1)}) \tag{1.12}$$

Ciò significa che il sistema di predizione è tanto più efficiente quanto più lunga è la sua memoria. Naturalmente il costo da pagare è sempre la maggiore complessità computazionale.

Un altro metodo per la *compattazione* efficiente di sequenze di simboli dipendenti è la codifica **run length**. Esso si impiega lì dove sono frequenti lunghe sequenze dello stesso simbolo. Essa consiste nel codificare non i singoli simboli ma la lunghezza delle stringhe di simboli con lo stesso valore. Sono queste lunghezze a rappresentare il nuovo alfabeto sorgente che può essere codificato a sua volta con ciascuno dei metodi visti in precedenza.

1.3 La codifica con perdita di informazione

L'oggetto tipico della *compressione* è un insieme di dati acquisiti, cioè riportati nel mondo digitale a partire da quello analogico.
L' **acquisizione** è, di per sé, un processo *lossy*, cioè con perdita di *informazione*. Infatti bisogna approssimare dei valori esprimibili da infinite cifre decimali, i valori analogici, a valori rappresentati da una stringa finita di simboli binari, valori numerici. Tale processo, più in generale il processo di riduzione della risoluzione della rappresentazione di un valore, va sotto il nome di **quantizzazione**.
Inoltre, bisogna estrarre da una distribuzione continua dei valori nello spazio o nel tempo, una quantità finita di campioni rappresentativi. Ad esempio non possiamo esprimere la luminosità di tutti i punti di una superfice, ma solo di una quantità discreta di regioni in cui è partizionabile la superficie stessa. L'operazione con la quale riduciamo il numero di valori rappresentativi da una quantità continua ad una quantità discreta è denominata **campionamento**. Si usa in realtà il termine campionamento anche nel caso si riduca semplicemente una cardinalità discreta di valori ad una inferiore.
L'acquisizione numerica dei dati tanto è possibile quanto essi non hanno bisogno di una precisione infinita e di una registrazione continua, ma, ai fini del loro impiego, possono essere rappresentati con risoluzione limitata e soltanto da un campionamento dei valori reali.
Se la perdita di informazione è qualcosa di intrinsecamente connaturato ai processi di acquisizione, essa nelle applicazioni avviene anche a valle dell'acquisizione stessa ed in maniera selettiva, con lo scopo ben preciso di diminuire la quantità di dati da gestire. Non sempre infatti con le tecniche di codifica senza perdita di informazione si può ottenere un tasso sufficientemente spinto per soddisfare le specifiche di memoria, risorse di trasmissione o complessità computazionale.
Tecniche elementari di campionamento e quantizzazione non costituiscono tuttavia la soluzione migliore al problema: per questo motivo nascono le tecniche di *compressione* dati.

1.3.1 Teoria tasso-distorsione

Rispetto alla codifica *lossless* la codifica *lossy* presenta degli elementi di novità. A volte le specifiche permettono di eliminare semplicemente l'informazione inutilizzata, altre volte tuttavia la perdita di informazione deve necessariamente riguardare anche dati potenzialmente utili dal punto di vista ap-

plicativo. Mentre per la codifica *lossless* il *tasso di codifica* era sufficiente a determinare la qualità dell'elaborazione, ora entra in gioco un'altro fattore estremamente importante: la qualità dei dati ricostruiti.
Il grado di perdita dell'informazione viene denominato **distorsione**. La distorsione è tanto maggiore quanto i dati ricostruiti, a valle della *compressione*, distano in qualità dai dati originali. E' molto importante fin da ora chiarire che il concetto di qualità non può essere un concetto assoluto, ma esso dipende necessariamente dall'applicazione, e cioè dall'impiego che si fa dei dati ricostruiti.
Fissata la misura della qualità, è valido il concetto che una buona tecnica di *compressione*, stabilito un certo tasso di codifica, deve essere in grado di ridurre al minimo la distorsione. Se D è la misura della distorsione, e R il tasso, ciò che si desidera è una tecnica che per ogni fissato livello di distorsione D codifichi la sorgente al tasso R più piccolo possibile o, al contrario, per ogni fissato *tasso di codifica* R comporti la minima distorsione.
La **Rate-Distortion Theory** è quella branca della *teoria dell'informazione* che si occupa, appunto, di trovare le prestazioni limite teoriche, in termini di curve $R(D)$ e $D(R)$ per assegnate sorgenti e misure di distorsione. Anche per sorgenti piuttosto semplici, tuttavia, esistono pochi risultati in forma chiusa. Inoltre, anche quando le curve limite sono note, i metodi di codifica esistenti forniscono prestazioni lontane da quelle ottime, soprattutto a causa dei vincoli di memoria e complessità computazionale ai quali si deve sottostare.

1.3.2 Misure di distorsione

Dato che la valutazione della qualità della codifica dipende dall'uso che si deve fare dei dati codificati non è possibile definire un metodo di misura universalmente valido. Si deve scegliere allora il metodo che di volta in volta risulta più adatto all'applicazione. Un buon metodo di misura, per poter essere convenientemente utilizzato, deve poi essere trattabile con sufficiente semplicità dal punto di vista analitico e numerico.
Un modo semplice di stimare la degradazione della qualità del segnale può essere quello di valutare la distanza euclidea tra i dati originali e dati compressi riportati nella rappresentazione originaria, cioè decodificati. Per derivabilità analitica, si preferisce impiegare non la distanza ma il suo quadrato, dunque non l'errore assoluto ma l'errore quadratico (**Square Error**).

$$\mathrm{SE} = ||\underline{x} - \underline{\hat{x}}||^2 = \sum_{i=1}^{N} (x_i - \hat{x_i})^2 \tag{1.13}$$

Nella teoria dei segnali esso costituisce una misura di energia.
In molte applicazioni di media l'errore quadratico sul numero di campioni, cioè si adotta come misura l'errore quadratico medio(**Mean Square Error**).

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2 \tag{1.14}$$

che rappresenta una misura di potenza. In realtà abbiamo sostituito la media campionaria a quella che dovrebbe essere una media statistica, sotto l'ipotesi di ergodicità.
Si può definire anche un rapporto segnale rumore (**Signal to Noise Ratio**) definito come:

$$\mathrm{SNR} = \frac{P}{MSE} \tag{1.15}$$

dove P è la varianza del segnale originario.
Esso non è altro che il rapporto tra la potenza del segnale originario e quello del rumore introdotto con la *compressione*. L'SNR, a differenza dell'MSE, non è un indice della degradazione bensì un vero e proprio indice della qualità.
Nelle applicazioni non sempre tutti i simboli della struttura dati originaria hanno la medesima importanza. In questo caso si può adottare un errore quadratico pesato (**Weighted Square Error**), definito nel seguente modo:

$$\mathrm{WSE} = (\underline{x} - \underline{\hat{x}})^T \cdot \underline{\underline{W}} \cdot (\underline{x} - \underline{\hat{x}}) \tag{1.16}$$

Dove $\underline{\underline{W}}$ è la matrice dei pesi. La definizione di WMSE segue banalmente.

1.3.3 Metodi di codifica lossy

I metodi di codifica *lossy* si basano essenzialmente sulla quantizzazione dei dati. I dati da codificare vengono rappresentati cioè con una precisione minore ed in questo modo è possibile risparmiare bit.
Rispetto ad una quantizzazione bruta dei dati tuttavia la teoria della codifica con perdita di informazione si pone i seguenti problemi:

- Vi sono rappresentazioni più idonee alla rappresentazione dei dati?
- A quale delle informazioni va assegnata una maggiore precisione?
- Come misurare le prestazioni di un codificatore *lossy*?

Il progetto di un codificatore *lossy* risponde alle precedenti esigenze prevedendo:

- un modello efficace dei dati (*modello dei quantizzatori scalari o vettoriali, transform coding*);
- una tecnica di assegnazione della precisione alle diverse parti dell'informazione da codificare (*rate allcocation*);
- una misura della qualità dei dati ricostruiti che possa guidare le scelte che regolano il funzionamento del codificatore (*misura della distorsione*).

Nei paragrafi successivi ci occuperemo di tutti questi aspetti, illustrandone i principi fondamentali.

1.4 La quantizzazione scalare

In molte applicazioni di codifica *lossy* bisogna rappresentare tutti i possibili valori assunti dai simboli sorgente, con un ben più ristretto numero di indici. Tale processi è detto **quantizzazione**. Gli input e gli output di un quantizzatore possono essere vettori oppure scalari. Consideriamo dapprima il caso più semplice di quantizzatore scalare.
Un **quantizzatore scalare** [4], come tutti i quantizzatori, è caratterizzato da una funzione di codifica EQ e da una di decodifica DQ. La prima opera associando ad ogni elemento x dell'insieme $\Re$ dei numeri reali un indice dell'insieme $I \equiv \{1, ..., N\}$.

$$EQ : x \in \Re \rightarrow i \in I \tag{1.17}$$

La seconda invece associa ad ognuno degli indici i, che praticamente rappresenta una particolare porzione dell'asse dei numeri reali, un numero y_i (parola codice) anche esso reale.

$$DQ : i \in I \rightarrow y_i \in Y \subset \Re \tag{1.18}$$

Nei casi di interesse pratico la cardinalità N di I è finita. Dalla composizione delle funzioni di codifica e di decodifica si ottiene la vera e propria funzione di quantizzazione:

$$Q : x \in \Re \rightarrow y_i \in Y \subset \Re \tag{1.19}$$

Nel seguito restringeremo l'attenzione ai soli **quantizzatori regolari**, per i quali le celle o regioni di quantizzazione $R_i = \{x|EQ(x) = i\}$ sono segmenti adiacenti dell'asse reale

$$R_i = \{x|x_{i-1} < x \leq x_i\} \tag{1.20}$$

individuati dalle soglie di decisione $x_0 < x_1 \ldots < x_{N-1} < x_N$, con $x_0 = -\infty$ ed $x_N = +\infty$. Inoltre le parole codice sono interne sono interne alle rispettive celle di quantizzazione, per cui risulta, in definitiva,

$$x_0 < y_1 < x_1 < y_2 < x_2 \ldots < x_{N-1} < y_N < x_N \tag{1.21}$$

Un quantizzatore scalare di largo impiego è quello uniforme, cioè quello per in gli intervalli interni $[x_1, x_2], ..., [x_{N-2}, x_{N-1}]$ sono tutti di egual misura.

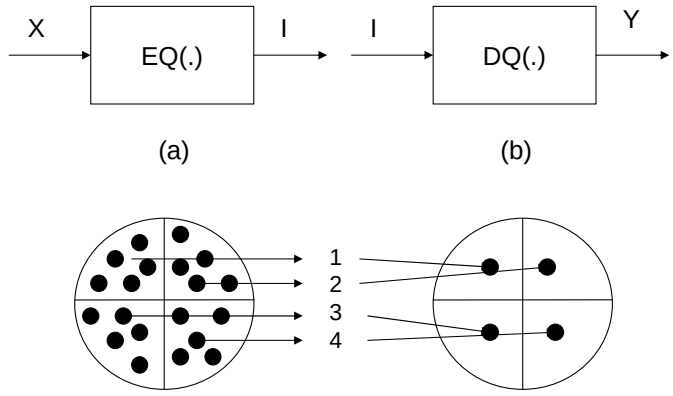


Figura 1.4: Principio di funzionamento di un generico quantizzatore. (a) Codifica. (d) Decodifica.

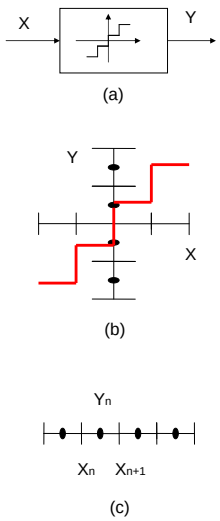


Figura 1.5: (a) Simbolo di un quantizzatore scalare regolare (codifica + decodifica). (b) Caratteristica di un quantizzatore scalare uniforme. (c) Sintesi grafica della caratteristica di un quantizzatore scalare.

1.4.1 Il quantizzatore scalare ottimo

La *quantizzazione scalare* in genere viene impiegata al fine di approssimare gli elementi di un insieme R_i con il valore quantizzato y_i. Si tratta dunque di una codifica con perdita d'informazione, e se ne possono valutare le prestazioni in termini di distorsione (errore quadratico medio):

$$D = E_X[(X - Q(X))^2] \tag{1.22}$$

o rapporto segnale rumore:

$$SNR = \frac{E_X[X^2]}{E_X[(X - Q(X))^2]} \tag{1.23}$$

dove in entrambi i casi abbiamo indicato con $E_X[.]$ l'operatore di media statistica valutata sulla variabile aleatoria X.

Non sempre il quantizzatore uniforme dà buoni risultati in distorsione. E'ovvio infatti che il valore dell'errore quadratico medio dipende dalla pdf (probability density function) f_X di X, per cui si può massimizzare l'SNR scegliendo le parole codice y_i in funzione delle statistiche di X, cioè progettando un quantizzatore non uniforme. La realizzazione di un tale quantizzatore può essere abbastanza impegnativa, per cui spesso viene sostituito con la cascata dei tre sistemi, più semplici, mostrati in fig. 1.6, dove

1. $G(\cdot)$ effettua una opportuna trasformazione non lineare e senza memoria, detta per motivi storici *compression*, della variabile aleatoria X;

2. Q_U è un quantizzatore uniforme a N livelli nel range $[-B/2, B/2]$ che opera sulla variabile trasformata $Z = G(X)$;

3. $G^{-1}(\cdot)$ effettua la trasformazione inversa a $G(\cdot)$ detta *expansion*.

Si dimostra che, scegliendo opportunamente i tre blocchi è possibile realizzare un qualsiasi quantizzatore non uniforme. Tale sistema prende il nome di *compandor*, dalla contrazione di compressor ed expandor, e si parla conseguentemente di tecnica di **companding**.

E' evidente che le caratteristiche del quantizzatore non uniforme equivalente sono dettate dalla funzione $G(\cdot)$. Per capire come conviene scegliere questa funzione, facciamo l'ipotesi di voler progettare un quantizzatore per $N \to \infty$ livelli, nel quale tendono di conseguenza a zero le ampiezze degli

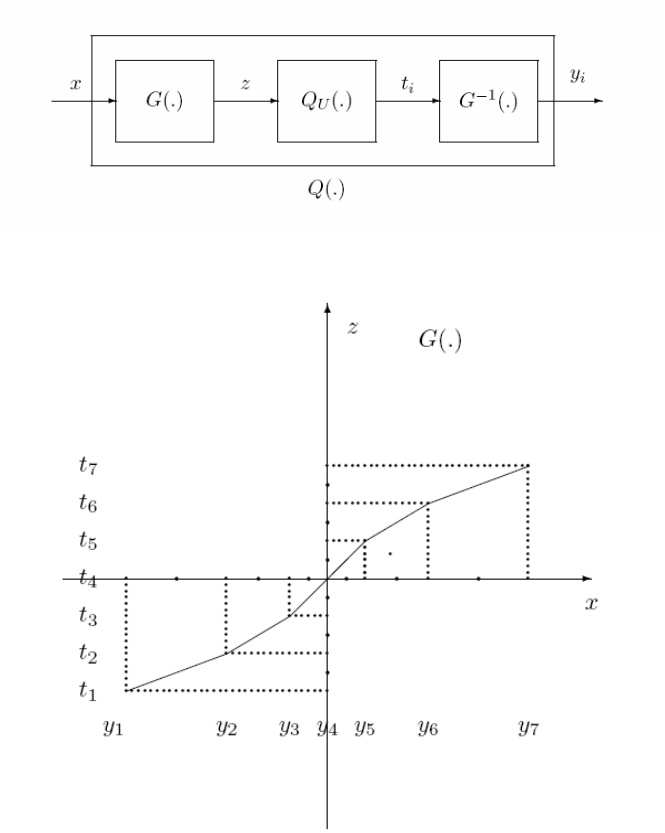


Figura 1.6: Schema a blocchi della tecnica del companding e caratteristica di un compandor.

intervalli di quantizzazione (ipotesi di alta risoluzione). Supponiamo inoltre di fissare il range finito B del quantizzatore uniforme del compandor.

Dato che la funzione $z = G(x)$ proietta celle di quantizzazione di X nelle corrispondenti celle di Z, deve risultare

$$G'(x) = \lim_{N\to\infty} \frac{\Delta_Z(x)}{\Delta_X(x)} \tag{1.24}$$

dove con Δ_X e Δ_Z si sono indicate le dimensioni delle celle di quantizzazione per i punti x e $z = G(x)$ rispettivamente, mentre con $G'(\cdot)$ si è espressa la derivata prima di $G(\cdot)$. Dal momento che poi il quantizzatore che opera su Z è uniforme, varrà $\Delta_Z = \frac{B}{N}$ in ogni punto. Di conseguenza

$$G'(x) = \lim_{N\to\infty} \frac{B}{N\Delta_X(x)} = B\lambda(x) \tag{1.25}$$

dove si è introdotta la funzione $\lambda(x)$ che assume il significato di concentrazione normalizzata delle celle di quantizzazione nell'intorno del punto x. In altri termini, vale la relazione

$$\frac{N[a,b]}{N} = \int_a^b \lambda(x)dx \tag{1.26}$$

dove $N[a,b]$ indica appunto il numero di celle di quantizzazione nell'intervallo $[a,b]$.

Passiamo a questo punto a esplicitare la distorsione come

$$D = \lim_{N\to\infty} E_X[(X - Q(X))^2]$$

$$= \lim_{N\to\infty} \sum_{i=1}^{N} E_{X\in R_i}[(X - y_i)^2] \cdot \Pr(X \in R_i)$$

e osserviamo che per $N \to \infty$ (alta risoluzione)

- le ampiezze degli intervalli di quantizzazione $\Delta_X(x) = \frac{1}{N\lambda(x)}$ tendono a zero;
- la pdf si può considerare costante in ogni cella, per cui la distorsione in ogni intervallo si può approssimare come $\frac{\Delta_X^2(x)}{12}$;
- la probabilità che X appartenga all'intervallo R_i si può approssimare con $\Pr(X \in R_i) = f_X(y_i)dx$.

L'integrale di distorsione diventa quindi

$$D = \lim_{N\to\infty} \sum_{i=1}^{N} \frac{1}{12N^2\lambda^2(y_i)} f_X(y_i)dx$$

e in definitiva si ha:

$$D = \frac{1}{12N^2} \int_{\Re} \frac{f_X(x)}{\lambda^2(x)} dx \tag{1.27}$$

Non ci resta dunque che ricavare la funzione $\lambda(\cdot)$, legata alla $G(\cdot)$ del compandor, che minimizza la distorsione. A questo scopo si ricorre alla disuguaglianza di Holder

$$\int u(x)v(x)dx \leq [\int u(x)^a dx]^{1/a} \cdot [\int v(x)^b dx]^{1/b} \tag{1.28}$$

con a e b che soddisfano la relazione $1/a + 1/b = 1$. Ponendo $u(x) = [f_x(x)/\lambda^2(x)]^{1/3}$ e $v(x) = \lambda^{\frac{2}{3}}(x)$, e ricordando che $\int_{\Re} \lambda(x)dx = 1$ si perviene alla relazione

$$\int_{\Re} f_X^{1/3}(x)dx \leq [\int \frac{f_X(x)}{\lambda^2(x)} dx]^{1/3}$$

che, sostituita nell'espressione di D, permette di determinare il valore minimo possibile per la distorsione:

$$D_{opt} = \frac{1}{12N^2} (\int_{\Re} f_X^{1/3}(x)dx)^3 \tag{1.29}$$

corrispondente alla scelta

$$\lambda(x) = \frac{f_X(x)^{1/3}}{\int_{\Re} f_X(\xi)^{1/3} d\xi} \tag{1.30}$$

L'espressione di $G(\cdot)$ che garantisce la minima distorsione è quindi

$$G(x) = G(x_0) + B \frac{\int_{x_0}^{x} f_X(\xi)^{1/3} d\xi}{\int_{\Re} f_X(\xi)^{1/3} d\xi} \tag{1.31}$$

Più avanti vedremo che la tecnica del companding ci sarà utile per quantizzare vv.aa. di tipo Laplace, la cui pdf ha espressione analitica

$$f(x) = \frac{1}{\sqrt{2}\sigma} \cdot e^{-\frac{\sqrt{2}}{\sigma}|x-\mu|} \tag{1.32}$$

con $\mu = E[X]$ e $\sigma^2 = E[(X - \mu_X)^2]$. Con la tecnica del companding si ricavano facilmente le espressioni per le parole codice e per le soglie di decisione del quantizzatore ottimo $Q(\cdot)$

$$x = \mu - \frac{3\sigma}{\sqrt{2}} \ln(1 - \frac{2|z|}{B}) \text{sign(z)} \tag{1.33}$$

dove con x si sono indicati i punti notevoli di $Q(\cdot)$ e con z i corrispettivi del quantizzatore uniforme $Q_U(\cdot)$ a media nulla e range B usato nel compandor.

All'atto pratico questi tipi di quantizzatore non si basano sulla pdf del segnale, che non è mai perfettamente nota, ma su una sua stima, e quindi le prestazioni sono subottime rispetto a quelle presentate. Il loro impiego è particolarmente efficace se si impiega come modello una pdf definita completamente da pochi parametri statistici, valutabili direttamente sul segnale da quantizzare. Così facendo è possibile progettare rapidamente il quantizzatore ottimo in maniera adattativa.

1.4.2 Progetto mediante algoritmo di Lloyd-Max

Un metodo alternativo di progetto di un quantizzatore non uniforme si basa su un calcolo implicito delle statistiche. Si fa operare cioè un algoritmo di apprendimento su un segnale simile a quello da quantizzare (*caso supervisionato*) o sul segnale stesso (*caso non supervisionato*). Anche quando si opera direttamente sul segnale d'interesse, spesso si fa un opportuno campionamento, in modo tale da usare solo parte dei dati a disposizione e di rendere più semplice e rapido il progetto dei quantizzatori.
L'algoritmo più diffuso in letteratura è quello di Lloyd-Max.

Metodo 4. ***Lloyd-Max.***
Fisso ad N il numero di intervalli del quantizzatore scalare. Inizio fissando N valori d ricostruzione $\{y_n\}$. Poi svolgo i seguenti passi.

1. *Calcolo gli estremi di decisione* $x_n^{(k)} = \frac{y_{n+1}^{(k)} + y_n^{(k)}}{2}$.
2. *Calcolo la distorsione* $D^{(k)} = \sum_{n=1}^{N} \int_{x_n^{(k)}}^{x_{n+1}^{(k)}} \left(x - y_n^{(k)}\right)^2 f_X(x)dx$.
3. *Se* $\frac{D^{(k-1)} - D^{(k)}}{D^{(k-1)}} < \epsilon$ *mi fermo; altrimenti continuo.*
4. *Computo i nuovi valori di ricostruzione* $y_n^{(k+1)} = \frac{\int_{x_n^{(k)}}^{x_{n+1}^{(k)}} x f_X(x)dx}{\int_{x_n^{(k)}}^{x_{n+1}^{(k)}} f_X(x)dx}$.

C'è da sottolineare che tramite l'algoritmo di Lloyd si fa un'ottimizzazione alternata, e non congiunta, del codificatore EQ (ovvero delle regioni R_i) e del decodificatore DQ (vale a dire delle parole codice). Per questo motivo non è detto che il risultato raggiunto sia un minimo globale per la distorsione.
Dell'algoritmo di Lloyd esistono diverse varianti, che spesso barattano le prestazioni in termini di distorsione con la velocità di elaborazione o di codifica.

1.5 Allocazione delle risorse

Un metodo semplice di codifica *lossy* è la suddivisione della sorgente in diversi flussi statisticamente omogenei e la quantizzazione di ciascuno di essi con un diverso quantizzatore scalare opportunamente dimensionato. Resta da capire però, una volta modellato ogni singolo quantizzatore per la particolare sottosorgente, quale risoluzione assegnare a ciascun quantizzatore per minimizzare, ad un certo bit rate, la distorsione sulla sorgente originaria.
Come gestire la quantità limitata di risorse disponibili (per intenderci, il numero di bit) al fine di ottenere la minore distorsione sui dati, è uno dei problemi fondamentali della codifica *lossless* e prende il nome di problema dell'**allocazione delle risorse** (**Rate Allocation**). Conoscendo le statistiche della sorgente, esso è risolvibile in forma chiusa per un banco di quantizzatori scalari [10].

1.5.1 Soluzione analitica per quantizzatori ad alta risoluzione

Supponiamo di avere un insieme di flussi di dati statisticamente indipendenti. Il k-esimo flusso è una sequenza di S_k campioni, modellati come variabili aleatorie indipendenti e identicamente distribuite con pdf nota f_k. Supponiamo che, in generale, gli S_k siano diversi fra loro, e parliamo quindi, con un'analogia idraulica, di "flussi a differente **portata**".

Ci poniamo a questo punto il problema di quantizzare i dati, attraverso un opportuno banco di quantizzatori scalari $Q_1, \cdot, Q_M$, in modo da minimizzare la distorsione per le risorse assegnate. Questo è un problema molto frequente nella codifica *lossy*, e la sua soluzione è ben nota in letteratura per flussi con uguale portata: qui la generalizziamo al caso di portate differenti.

Dato che conosciamo le pdf di ciascun flusso, possiamo costruire per ognuno di essi il quantizzatore ottimo, fissati i livelli di quantizzazione N_k. Nell'ipotesi di alta risoluzione, avremo quindi una distorsione, valutata in termini di errore quadratico totale (non medio), pari a

$$D_k = S_k \cdot \frac{\sigma_k^2}{12 N_k^2} \, (\int_{\Re} \tilde{f}_k(x)^{1/3} dx)^3 \tag{1.34}$$

dove abbiamo fatto comparire sotto l'integrale la pdf normalizzata $\tilde{f}_k(x)$, cioè avente la stessa forma di $f_k(x)$ ma varianza unitaria, e portato quindi in evidenza la varianza σ_k^2. Operando adesso le seguenti sostituzioni

$$h_k = \frac{1}{12} (\int_{\Re} \tilde{f}_k(x)^{1/3} dx)^3 \tag{1.35}$$

e

$$N_k = 2^{b_k} \tag{1.36}$$

(dove b_k sono i bit corrispondenti agli N_k livelli di quantizzazione ed h_k è il **fattore di forma** della distribuzione normalizzata $\tilde{f}_k(x)$ possiamo riscrivere la distorsione nel seguente modo:

$$D_k(b_k) = S_k \cdot h_k {\sigma_k}^2 2^{b_k} \tag{1.37}$$

La distorsione totale sarà dunque pari a

$$D_{tot}(b_1, ..., b_M) = \sum_{k=1}^{M} S_k \cdot h_k \cdot {\sigma_k}^2 \cdot 2^{-2b_k} \tag{1.38}$$

Non ci resta che minimizzarla sotto il vincolo

$$\sum_{k=1}^{M} S_k b_k = B_{tot} \tag{1.39}$$

dove B_{tot} è il budget di bit disponibili per l'intera codifica. Possiamo impiegare a tal scopo il metodo dei moltiplicatori di Lagrange. Scriviamo la funzione di costo di Lagrange come

$$J(b_1, \ldots, b_M, \alpha) = D_{tot}(b_1, \ldots, b_M) + \alpha(\sum_{k=1}^{M} S_k b_k - B_{tot}) \tag{1.40}$$

Derivando J rispetto a $b_1, \ldots, b_k$ e rispetto ad α, e ponendo a zero le derivate parziali, si ottiene l'allocazione ottima dei bit per ogni componente [3], pari a

$$b_k = \frac{B_{tot}}{S_{tot}} + \frac{1}{2}\log_2 \frac{\sigma_k^2}{\prod_{i=1}^{M} \sigma_i^{2S_i/S_{tot}}} + \frac{1}{2}\log_2 \frac{h_k}{\prod_{i=1}^{M} h_i^{S_i/S_{tot}}} \tag{1.41}$$

essendo $S_{tot} = \sum_{i=1}^{M} S_i$. In questa espressione

- il primo termine rappresenta il numero di bit ottimo in caso di identica pdf e di identica varianza per ciascun flusso di dati;
- il secondo termine tiene conto della differenza tra le varianze dei diversi flussi e si annulla completamente se esse sono uguali;
- il terzo termine valuta invece la differenza di forma delle pdf e si annulla per flussi identicamente distribuiti.

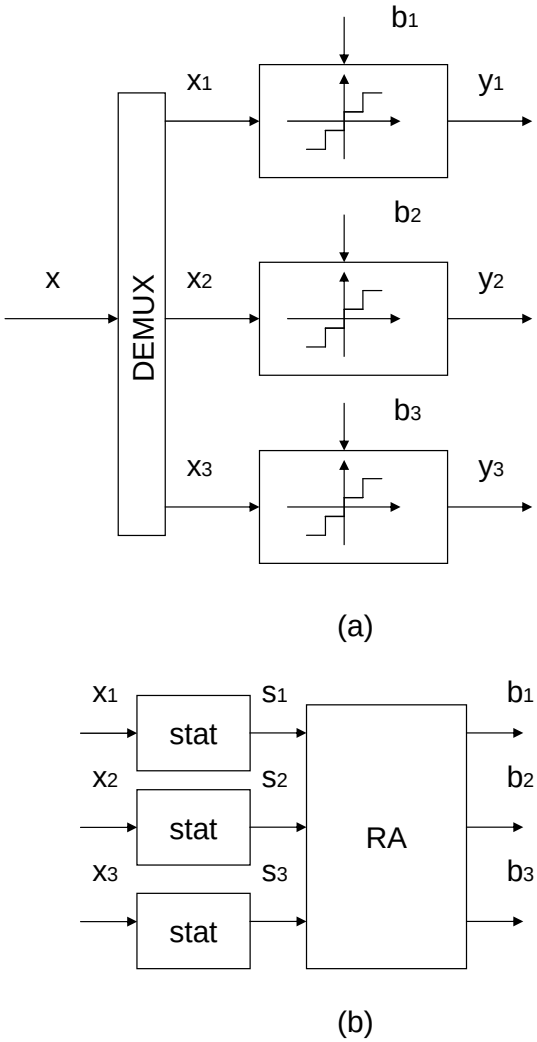


Figura 1.7: (a) Banco di quantizzatori scalari. (b) Rate allocation a partire dalle statistiche della sorgente.

Si noti come, per uguali portate, i termini a denominatore all'interno del logaritmo rappresentino delle medie geometriche, mentre per portate differenti sono delle medie geometriche pesate secondo l'esponente S_k/S_{tot}.

La soluzione trovata in questo modo rappresenta un minimo assoluto per la distorsione. Tuttavia nelle applicazioni ci si scontra con l'inconveniente di soluzioni non intere o negative. E' dunque necessario modificare leggermente l'algoritmo di assegnazione per trovare soluzioni fisicamente accettabili. Per eliminare l'influenza delle soluzioni negative, ad esempio, si possono escludere i flussi con assegnazioni negative e riformulare ad oltranza il problema finchè non si hanno solo soluzioni positive. Anche per assegnazioni di meno di un bit si può iterare lo stesso procedimento. Per quel che riguarda l'assegnazione non intera o il riscontro di un residuo di bit non assegnati, si possono seguire varie strade. Nell'ultimo capitolo offriremo una nostra soluzione, tra le tante possibili, per risolvere questi inconvenienti empirici.

1.5.2 Soluzioni greedy al problema dell'allocazione

Al problema dell'*allocazione delle risorse* si possono dare anche semplici soluzioni localmente ottime, basate su algoritmi miopi o, con termine inglese, **greedy** (avide). Si tratta di algoritmi iterativi basati su opportune euristiche con una prospettiva di ottimizzazione limitata all'iterazione in corso. Se l'euristica, cioè la modalità di scelta in base ad un'opportuna funzione di costo, è selezionata in maniera oculata, si possono ottenere dei buoni risultati. Ad esempio, nel caso di insiemi di diversa cardinalità, si può scegliere come funzione di costo l'errore quadratico e si può assegnare ad ogni iterazione un bit in più a quel quantizzatore che introduce al momento l'errore quadratico maggiore. Si possono impiegare sia quantizzatori standard, costruiti tramite companding, sia quantizzatori progettati tramite un algoritmo simile a quello di Lloyd.

Un modello molto semplice di allocazione greedy delle risorse è la **codifica per bitplane**. Nella codifica per bitplane i valori da quantizzare sono rappresentati in forma binaria, con virgola fissa, in segno e modulo. La cifra n-sima di ciascuna rappresentazione binaria è detta **bitplane**.
Se i valori da quantizzare hanno distribuzione uniforme vale quanto segue: l'energia decresce esponenzialmente al crescere del bitplane. Per questo motivo, sotto questa ipotesi, si può pensare di allocare le risorse assegnandole semplicemente ad un bitplane alla volta fino al loro esaurimento. Ne risulta una codifica progressiva che presenta il notevole vantaggio di essere anche **scalabile in qualità**: il flusso dati codificato può essere troncato in qualsiasi punto in modo tale che il primo segmento rappresenti i dati con più alta

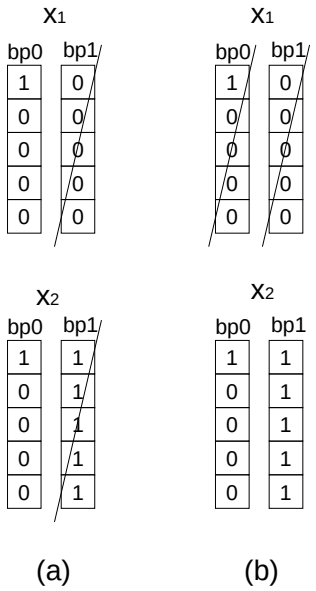


Figura 1.8: Subottimalità della codifica per bitplane per l'allocazione illustrata con un semplice esempio.

distrorsione rispetto al flusso completo. La decodifica scalabile risulta utile allorquando il flusso codificato è indirizzato diversi decodificatori, ognuno dei quali può gestire una diversa quantità di informazione (ad esempio per diverse risorse computazionali o di canale) .
Si noti come la codifica per bitplane sia una strategia subottima se le statistiche della sorgente non sono uniformemente distribuite. In fig. 1.8 ad esempio, come conseguenza di statistiche non uniformemente distribuite per le sorgenti x_1 ed x_2, nel bitplane 0 vi sono molti meno bit 1 che nel bitplane 1. Sotto il vincolo di 10 bit di risorse disponibili ed usando come misura l'MSE, la strategia di codifica (a) ($MSE = 9$) per bitplane è subottima rispetto alla strategia (b) ($MSE = 8$).

1.5.3 Codifica ottima per troncamento

La strategia migliore di allocazione delle risorse è un'analisi a posteriori. I dati, una volta partizionati, vengono codificati ad un bit-rate molto più alto di quello desiderato. Di ciascun insieme della partizione è possibile a questo

punto tracciare le reali curve tasso-distorsione. Il problema diventa adesso una semplice minimizzazione vincolata della funzione convessa di distorsione.

1.6 Codifica mediante trasformata lineare a blocco

Mettiamoci nel caso particolare in cui la sorgente originale è suddivisa in M flussi di informazione $\{x_i\}$ di medesima portata e con lo stesso fattore di forma.
In questo caso particolare la distorsione totale potrà essere riscritta nel seguente modo:

$$D_{tot} = Mh(\prod_{i=1}^{M} \sigma_i^2)^{\frac{1}{M}} 2^{-2\frac{B_{tot}}{S_{tot}}} \tag{1.42}$$

L'idea alla base della **codifica mediante trasformata** (per adesso ci riferiamo al sottocaso più semplice di trasformata *lineare a blocco*) o **transform coding** è quella di effettuare una trasformazione lineare sul vettore costituito dai coefficienti omologhi di ciascun flusso, in modo che l'*allocazione delle risorse* abbia prestazioni migliori sui nuovi coefficienti trasformati $\underline{t} = \{t_1, ..., t_M\}$ che sui dati originali $\underline{x} = \{x_1, ..., x_M\}$.

La matrice $\underline{\underline{A}}$ associata alla trasformazione lineare spesso è scelta in modo da essere ortonormale. La trasformazione ortogonale $\underline{t} = \underline{\underline{A}} \cdot \underline{x}$ infatti conserva l'energia e, se la misura della distorsione è l'MSE, vale che la distorsione sui coefficienti trasformati $\underline{t}$, è uguale a quella sui coefficienti originali $\underline{x}$.

1.6.1 Trasformata di Karhunen-Loeve

Si può dimostrare [4] che nel caso che i flussi $x_1, ..., x_M$ siano processi aleatori gaussiani, l'ipotesi di linearità della trasformata porta ad una soluzione ottima di progetto per l'*allocazione delle risorse*. La trasformata ottima è detta di **Karhunen-Loeve** (KLT) [12].

Sia $\underline{\underline{A}}$ la matrice ortonormale della trasformata. La matrice di correlazione dei coefficienti trasformati sarà:

$$\underline{\underline{R}}_t = E[\underline{t} \cdot \underline{t}^T] = \underline{\underline{A}} \cdot E[\underline{x} \cdot \underline{x}^T] \cdot \underline{\underline{A}}^T = \underline{\underline{A}} \cdot \underline{\underline{R}}_x \cdot \underline{\underline{A}}^T \tag{1.43}$$

La KLT è quella trasformata che decorre perfettamente x. Ciò significa che se decomponiamo spettralmente $\underline{\underline{R}}_x = \underline{\underline{U}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{U}}^T$, dove $\underline{\underline{\Lambda}}$ è la matrice diagonale degli autovalori di $\underline{\underline{R}}_x$ ed U la matrice che ha per vettori colonna gli autovettori, $\underline{\underline{R}}_t$ sarà proprio pari a Λ ed $\underline{\underline{A}}$, la matrice di trasformazione, non

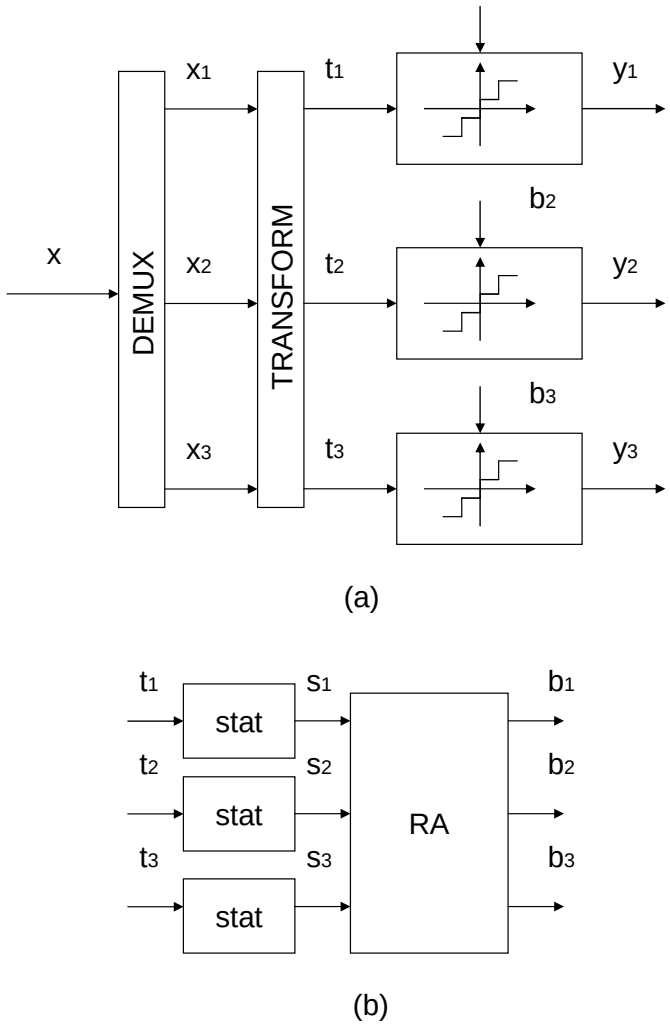


Figura 1.9: Principio di funzionamento della codifica mediante trasformata. (a) Lo schema di fig. 1.7 è modificato inserendo un operazione di trasformata tra le sottosorgenti. (b) La rate allocation avviene sui coefficienti trasformati.

sarà nient'altro che $\underline{\underline{U}}^T$.
La KLT ha dunque la forma:

$$\underline{t} = \underline{\underline{U}}^T \cdot \underline{x} \tag{1.44}$$

dove

$$\underline{\underline{R}}_x = \underline{\underline{U}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{U}}^T \tag{1.45}$$

Se effettuiamo l'*allocazione delle risorse* a valle della KLT, si può calcolare per la distorsione totale la seguente espressione:

$$D^t_{tot} = Mh(\prod_{i=1}^{M} \lambda_i)^{\frac{1}{M}} 2^{-2\frac{B_{tot}}{S_{tot}}} \tag{1.46}$$

Dove i $\{\lambda_i\}$ sono gli autovalori di $\underline{\underline{R}}_x$.
Poiché gli $\{x_n\}$ sono processi gaussiani, la trasformazione lineare A non ne altera la natura gaussiana. Per questo motivo il fattore di forma h ha lo stesso valore che nell'espressione 1.42 ed è quello che si ottiene da una pdf gaussiana normalizzata.

Si può dimostrare che per una generica matrice di covarianza R_x di variabili aleatorie x_i a media nulla e varianza σ_i^2, vale che:

$$\prod_{i=1}^{M} \lambda_i \leq \prod_{i=1}^{M} \sigma_i^2 \tag{1.47}$$

dove il segno di eguaglianza si può avere solo se $\underline{\underline{R}}_x$ è diagonale (variabili aleatorie incorrelate), cioè se i σ_i^2, a meno di una questione di ordinamento, sono uguali ai λ_i.
Da questa osservazione, confrontando le espressioni 1.42 e 1.46, si evince l'ottimalità della KLT.

1.6.2 Prestazioni delle trasformate

Per misurare le prestazioni di una generica trasformata si usa il concetto di **coding gain**:

$$CG = \frac{D^t_{tot}}{D_{tot}} \tag{1.48}$$

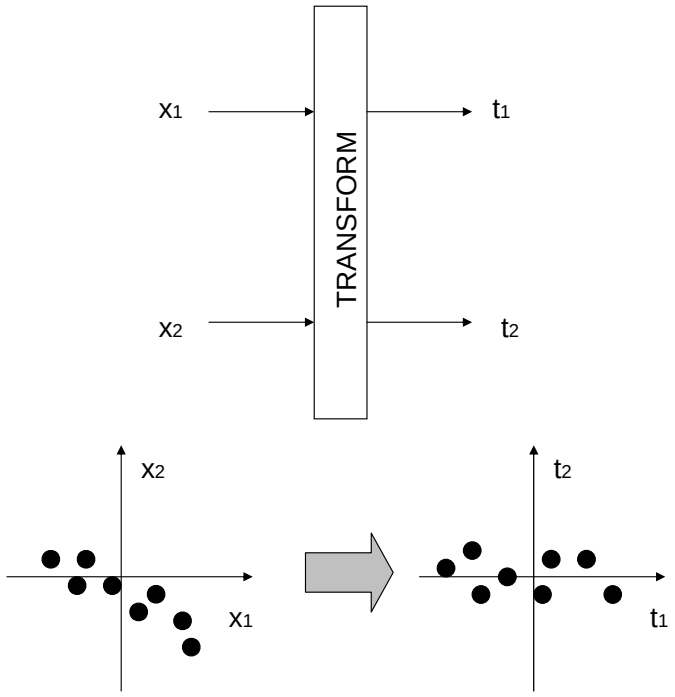


Figura 1.10: Esempio di trasformata decorrelante su due sorgenti correlate.

Sotto l'ipotesi di sorgenti gaussiane:

$$CG = \frac{det(\underline{\underline{R}}_y)^{\frac{1}{M}}}{det(\underline{\underline{R}}_x)^{\frac{1}{M}}} \tag{1.49}$$

avendo ricordato che la produttoria delle varianze è uguale al determinante della matrice di covarianza.

Si noti come non vi sia un teorema generale che affermi che quantità incorrelate possano essere quantizzate più efficientemente che variabili incorrelate. E' l'ipotesi gaussiana che permette di scrivere il *coding gain* secondo l'espressione 1.49. Tuttavia è pur vero che nelle applicazioni si può generalmente osservare che, a parità di energia, quanto più incorrelate ed indipendenti sono le variabili da codificare, tanto più efficiente risulta la codifica.

Si noti anche come la KLT, tra tutte le possibili trasformazioni lineari, è quella che, dopo aver ordinato i coefficienti in ordine di energia crescente, concentra la maggiore energia nei primi k ($k < N$). Ciò si evince dall'osservazione che, fissata l'energia totale e dunque anche la media aritmetica delle energie, la KLT minimizza la media geometrica. In generale la media geometrica, sotto l'ipotesi di media aritmetica vincolata, ha un massimo quando i valori mediati sono uguali e decresce man mano che i valori sono invece tra loro sbilanciati.
Sebbene ancora una volta queste considerazioni siano a rigore valide solo nel caso della KLT e di variabili gaussiane, ancora una volta dall'esperienza si può trarre che quanto più sbilanciate sono le energie delle sorgenti, a parità di energia complessiva, tanto più efficace è la rate allocation.

Detto ciò lo svantaggio della KLT non è tanto la sua mancata universalità quanto la complessità di calcolo delle matrici di trasformazione, che devono essere ricavate dalle statistiche del segnale.
Allo scopo di aggirare questo ostacolo, spesso si adottano altre trasformate lineari, subottime ma indipendenti dalle statistiche.

1.7 La quantizzazione vettoriale

Il metodo più generale di quantizzazione è la **quantizzazione vettoriale** [4], attraverso la quale vengono mappati nello spazio degli indici dei vettori e non degli scalari.
Abbiamo già visto per la codifica *lossless* che codificare sequenze di simboli è più efficiente che codificare simboli isolati. Analogamente la *quantizzazione vettoriale* può portare ad una distorsione ben più bassa di un banco di quantizzatori scalari che opera sulla stessa sequenza.
Nel paradigma della *quantizzazione vettoriale* la sorgente è raggruppata in blocchi. Ciascun blocco costituisce un vettore v da quantizzare. Un dizionario (**codebook**) che associa univocamente indici k a vettori di ricostruzione (**codeword**) w_k è presente sia al codificatore che al decodificatore. All'atto della codifica ciascun vettore v della sorgente è messo in corrispondenza iniettiva con un indice k. Tutti i vettori associati all'indice k si dicono appartenenti alla regione R_k. In decodifica, se un vettore è stato codificato con l'indice k sarà ricostruito con la codeword w_k. La distorsione risultante dalla singola codifica sarà $d(v, w_k)$, dove d è la metrica scelta.
Un quantizzatore vettoriale ben proggettato sarà tale da minimizzare la distorsione complessiva:

$$D = \sum_k \int_{R_k} d(v, w_k) f_V(v) dv$$

1.7.1 Progetto di un quantizzatore vettoriale

In letteratura non esiste un metodo che permetta di ricavare l'ottimo assoluto di un problema di *quantizzazione vettoriale*. L'algoritmo di progetto più impiegato è la generalizzazione dell'algoritmo di Lloyd proposto in [13]. L'algoritmo è noto come GLA, (**Generalized Lloyd Algorithm**).

Metodo 5. ***Lloyd-Max Generalizzato.***
Fisso ad N il numero di intervalli del quantizzatore scalare. Inizializzo fissando N valori d ricostruzione $\{\underline{w}_n\}$. Poi itero i seguenti passi.

1. *Calcolo le regioni di decisione* $R_n^{(k)} = \{\underline{v} | d(\underline{v}, \underline{w}_n^{(k)}) < d(\underline{v}, \underline{w}_i^{(k)}), i \neq n\}$.

2. *Calcolo la distorsione* $D^{(k)} = \sum_{n=1}^{N} \int_{R_n^{(k)}} \left(\underline{v} - \underline{w}_n^{(k)}\right)^2 f_{\underline{V}}(\underline{v}) d\underline{w}$.

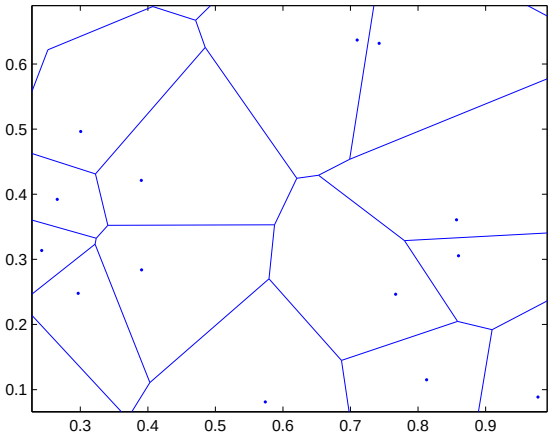


Figura 1.11: Una tassellazione di uno spazio vettoriale a due dimensioni ottenuta con il GLA.

3. *Se* $\frac{D^{(k-1)}-D^{(k)}}{D^{(k-1)}} < \epsilon$ *mi fermo; altrimenti continuo.*

4. *Computo i nuovi valori di ricostruzione* $\underline{w}_n^{(k+1)} = \frac{\int_{R_n^{(k)}} \underline{v} f_{\underline{V}}(\underline{v}) d\underline{v}}{\int_{R_n^{(k)}} f_{\underline{V}}(\underline{v}) d\underline{v}}$.

1.7.2 Quantizzatori vettoriali strutturati

Per semplificare computazionalmente il progetto dei quantizzatori, rinunciando a pretese di spinta ottimizzazione, è possibile imporre dei vincoli di struttura.
Un tipo di struttura molto diffusa è la struttura ad albero TSVQ (**Tree Structured Vector Quantization**) [4]. In seguito illustriamo l'algoritmo nel caso di albero binario.

Metodo 6. ***TSVQ.***
All'inizio l'insieme dei vettori da quantizzare viene rappresentato solo da due codeword. Il quantizzatore siffatto viene ottimizzato tramite GLA. A questo punto solo la regione a più alta distorsione viene mappata a sua volta in due regioni. Di tutte e tre le regioni così ottenute si valuta quale sia quella a più alta distorsione e così via, fino a che non si raggiunge il numero di codeword o la distorsione desiderata.

Lo spazio di partenza viene in questo modo suddiviso secondo una gerarchia di regioni innestate. E' possibile associare a tale gerarchia una struttura ad albero così come mostrato in fig.
E' da notare che la TSVQ può fornire una rappresentazione a qualità progressivamente crescente dei dati. Ogni indice può essere rappresentato in forma di stringa binaria, come sequenza delle suddivisioni della regione di partenza. In questo modo decodificando solo i primi bit dell'indice si può ottenere una ricostruzione dei dati con un più spinto livello di approssimazione.

1.7.3 Quantizzazione scalare di dati vettoriali

La scelta più semplice di *quantizzazione vettoriale* si riduce ad un banco di quantizzatori scalari. Anche se si perde dal punto di vista dell'efficienza di codifica, dal punto di vista della velocità computazionale è più semplice quantizzare un vettore componente per componente, piuttosto che integralmente. Naturalmente in questo modo bisogna prevedere un codebook per ogni componente, la qual cosa aumenta la dimensione della memoria necessaria alla tabella di transcodifica. Per evitare di trasmettere i codebook, e per migliorare la semplicità degli algoritmi, si adotta spesso una pre-modellizzazione statistica della sorgente. In questo modo si risolve a priori il problema dell'*allocazione delle risorse* per ogni componente grazie ad una formulazione analitica, alleggerendo l'onere di calcolo. Fissando a priori la distribuzione inoltre a destinazione si ha già a disposizione la tabella di transcodifica.

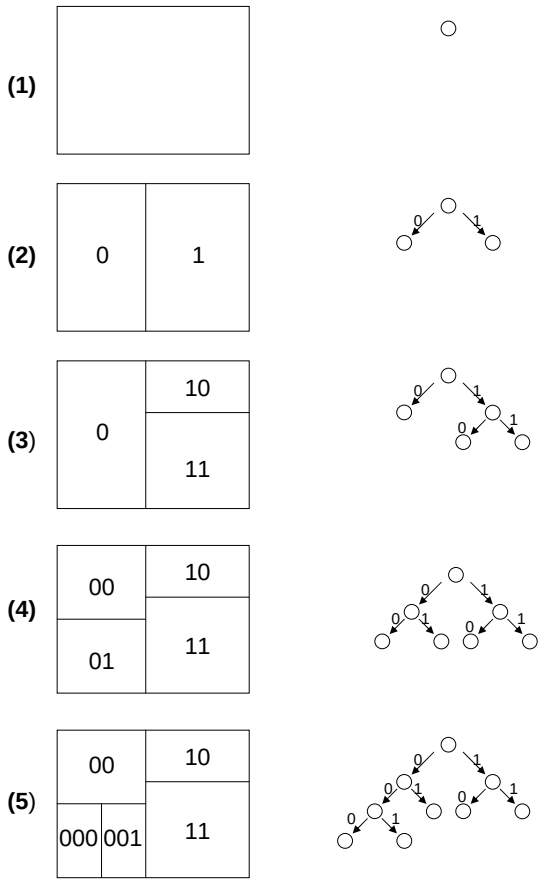


Figura 1.12: Principio di funzionamento della TSVQ con codifica binaria degli indici.

Bibliografia

[1] T. Cover, J. Thomas “Elements of information theory”, *Wiley*, 1991.

[2] K. Sayood “Introduction to data compression”, *Morgan-Kaufmann*, 2000.

[3] D. Salomon “Data compression - the complete reference”, *Springer*, IV Ed., 2007.

[4] A. Gersho, R. M. Gray “Vector quantization and signal compression”, *Kluwer*, 1992.

[5] D. A. Huffman, “A met for the construction of minimum redundancy codes”, *Proc. IRE*, vol. 40, pp. 1098-1101, 1952.

[6] I. H. Witten, R. M. Neal and J. G. Cleary, “Arithmetic coding for data compression”, *Commun. ACM*, vol. 30, pp. 520-540, 1987.

[7] F. Rubin, “Arithmetic stream coding using fixed precision registers”, *IEEE Tran. Inf. Theory*, vol. 25, 672-675, 1979.

[8] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression”, *IEEE Trans. Inf. Theory*, vol. 23, pp. 337-342, 1977.

[9] T. A. Welch, “A technique for high-performance data compression”, *Computer*, vol. 17, pp. 8-19, 1987.

[10] J. Huang and P. Schultheiss, “Block quantization of correlated Gaussian random variables”, *IEEE Transaction on Communication Systems*, vol. 11, pp. 289-296, 1963.

[11] L.Cicala, “Ottimizzazione delle risorse nella classificazione e codifica di immagini multispettrali.”, Tesi di Laurea, Univ. degli Studi Federico II di Napoli, 2003.

[12] Karhunen and Loeve.

[13] X. Lin, A. Buzo and R. M. Gray, “An algorithm for vector quantization design”, *IEEE Transaction on Communications*, vol. 28, pp. 84-95, gennaio 1980.

Capitolo 2

Analisi tempo-frequenza

Parole chiave

Nozioni propedeutiche

Algebra lineare, spazi vettoriali, spazi di Hilbert, trasformata di Fourier (FT), distribuzione di Dirac, trasformata di Fourier tempo discreta (DTFT, DFT).

Concetti introdotti nel capitolo

Trasformata di Fourier finestrata (STFT), trasformata wavelet a parametri continui (CPWT), decomposizione diadica, analisi multirisoluzione, banco di filtri, trasformata wavelet a parametri discreti (DPWT), trasformata coseno tempo discreta (DCT), trasformata wavelet tempo discreta (DWT), analisi multirisoluzione, codifica delle sottobande, lifting scheme, lifting scheme generalizzato, trasformate multidimensionali, trasformate separabili.

2.1 Tecniche di analisi della sorgente

Una generica sorgente, prima di essere quantizzata per la codifica lossy, può essere opportunamente scomposta in sottosorgenti. Al progettista si presentano varie alternative di scomposizione.
Una prima alternativa è l'impiego di un classificatore che possa selezionare all'interno della sorgente principale diverse sottosorgenti statisticamente omogenee. Si parla, dunque, di **codifica mediante classificazione**.
Una seconda alternativa è, invece, suddividere la sorgente originaria, tramite tecniche di campionamento, in tante sottosorgenti tra di loro correlate. A questo punto si può effettuare una trasformata sui campioni di ciascuna sottosorgente per decorrelare i diversi flussi di informazione. Si parla di **codifica mediante trasformata generalizzata**.
Naturalmente queste due tecniche possono essere eventualmente combinate nella stessa strategia di codifica.

Nelle due categorie di tecniche accennate ricadono i casi più semplici rispettivamente della *quantizzazione* e della *codifica mediante trasformata lineare a blocco*, già descritte nel capitolo precedente. I due approcci, che andremo ad approfondire in questo lavoro di tesi, tuttavia hanno un carattere più generale per i seguenti principali motivi.
Nella *codifica mediante classificazione* il sistema di classificazione dei dati può essere diverso da una semplice quantizzazione, in quanto, ad esempio, si può tenere in conto di altre informazioni esterne rispetto al singolo dato da classificare (informazione di *contesto*).
Per quel che riguarda la *codifica mediante trasformata generalizzata*,

- in primo luogo essa può essere effettuata non solo sui campioni omologhi di ciascun flusso ma eventualmente anche su campioni precedenti o successivi (*trasformata con memoria o dispersiva*);
- essa può essere non lineare;
- i flussi considerati possono essere campionati con un passo diverso (*analisi multirisoluzione*).

Nel seguente capitolo si illustrano i principali strumenti impiegati per progettare trasformate. L'enfasi è sulle rappresentazioni tempo-frequenza (o spazio-frequenza a seconda dal campo di applicazione). In particolare, dopo un rapido richiamo alla *trasformata di Fourier*, si espongono i principi fondamentali di una trasformata di ampio impiego sia nell'ambito dell'analisi di

segnali in generale, sia nello specifico settore della codifica di segnali visuali: la *trasformata wavelet*. Infine si espone la tecnica dei *lifting schemes* di fondamentale importanza sia per l'*analisi multirisoluzione* mediante trasformata wavelet sia per *analisi multirisoluzione* di tipo non lineare.

2.2 Analisi tempo-frequenza nel continuo

In questo paragrafo verrà in primo luogo effettuato un richiamo alla trasformata di Fourier [1][1] per metterne in evidenza l'ottima capacità di localizzazione frequenziale a scapito di quella temporale; successivamente si presenterà così la STFT (*Trasformata di Fourier Finestrata*) quale strumento per migliorare il dettaglio temporale della FT (*Trasformata di Fourier*) . In seguito, verrà presentata la CPWT (*Trasformata Wavelet a Parametri Continui*) come una generalizzazione della STFT.

2.2.1 Trasformata di Fourier

La maggior parte dei segnali reali fisici è di tipo non stazionario, cioè con caratteristiche variabili nel tempo. Per esempio, una combinazione di sinusoidi con diversi parametri (ampiezze A_i, tempi di inizio t_i, frequenze istantanee f_i, fasi iniziali ϕ_i e coefficienti di smorzamento a_i) rappresenta molto bene un tipico segnale non stazionario.

$$x(t) = \sum_{i=0}^{N} A_i e^{-a_i(t-t_i)} cos(2\pi f_i t + \phi_i) u(t - t_i) \tag{2.1}$$

Se su questo segnale si effettua una trasformata di Fourier, si ottiene una funzione $X(f)$ il cui modulo dice qualcosa di evidente sulla presenza delle componenti armoniche f_i e delle rispettive ampiezze A_i . Le informazioni relative agli altri parametri sono inglobate nella fase di $X(f)$; il problema è che tali informazioni, che hanno una chiara denotazione integrale, sono completamente illeggibili dal punto di vista puntuale. In altre parole, la *trasformata di Fourier* (**Fourier Transform**: **FT**) scompone il segnale nelle varie sinusoidi di diversa frequenza che lo compongono evidenziando la presenza delle componenti armoniche ma non permettendo di ricavare facilmente informazioni su quando e come tali frequenze siano effettivamente presenti. Il motivo di tutto ciò è intrinseco nella definizione della trasformata:

$$FT[x](f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft} dt = < x(t), e^{j2pift} > \tag{2.2}$$

dove il con $< \ . \ >$ si è indicato il prodotto scalare nello spazio $L^2(\mathbb{R})$ delle funzioni quadrato sommabili e con j l'unità immaginaria.

Sebbene la *trasformata di Fourier* sia in grado di localizzare perfettamente un tono sinusoidale, perde completamente traccia di un impulso temporale,

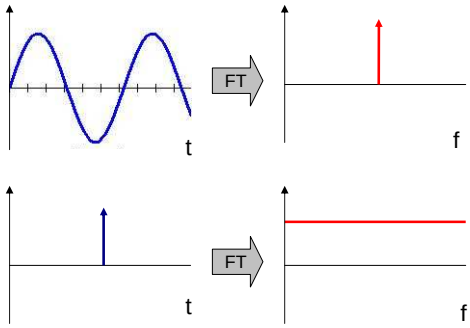


Figura 2.1: Analisi mediante trasformata di Fourier continua di una sinusoide e di un impulso.

il cui modulo nel dominio della frequenza è rappresentato da una funzione costante sull'asse delle frequenze. In altri termini, questa trasformazione, seppure adatta all'analisi di segnali stazionari, perché permette di evidenziare fenomeni ricorrenti nel dominio temporale e dunque localizzati in frequenza, non si presta invece all'analisi di segnali non stazionari, i quali si caratterizzano per eventi puntualmente circoscritti nel dominio del tempo e dispersi nel dominio della frequenza (vedi fig. 2.1).

2.2.2 Trasformata di Fourier finestrata

Per superare i limiti esposti nel paragrafo precedente, negli anni '40 è stata introdotta la *Trasformata di Fourier a finestra* (o **Short-Time Fourier Transform**: **STFT**) [1] la quale prevede di sfruttare una funzione a supporto compatto per mappare il segnale in una funzione sia del tempo che della frequenza. Come precedentemente osservato infatti per i segnali non stazionari occorre inserire nella trasformazione una certa dipendenza dal tempo, in modo da poter individuare le non stazionarietà. Il modo più immediato consiste nel rendere locale la *trasformata di Fourier* non eseguendola su tutto il supporto del segnale ma solo su porzioni di esso. La segmentazione del segnale originale avviene moltiplicandolo per una finestra che trasla nel tempo. L'espressione risultante della STFT è la seguente:

$$STFT[x](\tau, f) = \int_{-\infty}^{+\infty} x(t)g^*(t-\tau)e^{-j2\pi ft}dt =< x(t), g(t-\tau)e^{j2pift} > \tag{2.3}$$

La STFT trasforma dunque un segnale 1D $x(t)$ in una funzione 2D del tempo e della frequenza (τ rappresenta il punto di applicazione della finestra). Con questa nuova trasformazione si ottiene lo spettro del segnale all'interno della finestra temporale definita dalla funzione $g(t-\tau)$ ottenendo quindi un'informazione sul suo contenuto armonico solo in un intorno fissato dell'istante di tempo τ. In genere non si utilizza la finestratura rettangolare, perché determina delle discontinuità artificiali, non esistenti cioè nel segnale originario. Spesso invece si sceglie $g(t) = \alpha e^{-\beta t^2}$ (finestra gaussiana), e si parla in tal caso di *trasformata di Gabor*, il quale fu il primo ad introdurre la STFT nel 1946.

Il vantaggio evidente della STFT, rispetto alla FT, è dunque quello di essere in grado di localizzare il contenuto frequenziale in un intervallo di durata Δt; d' altra parte, come altra faccia della medaglia, essa non è in grado di discriminare due toni puri la cui distanza risulti essere inferiore a Δf. Si può dimostrare che tra indeterminazione temporale ed indeterminazione frequenziale esiste una relazione di proporzionalità inversa (**principio di indeterminazione**) [1], per cui necessariamente una trasformata con minore indeterminazione temporale possiede anche maggiore indeterminazione frequenziale.

La capacità di localizzazione può essere rappresentata in un piano tempo-frequenza (si veda la fig. 2.2). Si osservi come le celle di inteterminazione della SFT abbiano le stesse dimensioni per ogni coordinata temporale e frequenziale.

La trasformata STFT è dunque inadeguata per l'elaborazioni di alcuni segnali naturali, come le immagini, dato che sarebbe utile localizzare con diversa risoluzione spaziale e temporale fenomeni che evolvono rapidamente (*anomalies*) e fenomeni che invece evolvono lentamente (*trend*). Dato che segnali ad alta frequenza operano in intervalli temporali brevi (al contrario di quanto succede per le basse frequenze) sarebbe importante che lo strumento di analisi disponesse di celle tempo-frequenza di dimesioni adattative rispetto alla frequenza, con maggiore indeterminazione temporale alle frequenze di *trend* e minore alle alte frequenze in cui si verificano le *anomalie*.

Una scelta opportuna di funzioni base flessibili può permettere il raggiungimento di questo obbiettivo. Per capire come ciò possa avvenire bisogna però introdurre una nuova trasformata: la *trasformata wavelet*.

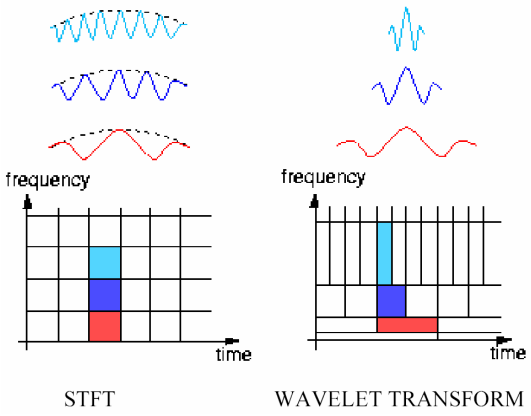


Figura 2.2: Rappresentazione spazio-frequenza per STFT e DPWT diadica.

2.2.3 Trasformata wavelet a parametri continui

La trasformata di Fourier gode della seguente proprietà di *scaling* [1]:

$$FT[x(\frac{t}{a})](f) = |a|FT[x(t)](af) \tag{2.4}$$

Ciò significa che se nella STFT impieghiamo funzioni $g(t)$ scalate possiamo avere spettri espansi, o meglio diminuendo il supporto (e dunque l'indeterminazione) temporale con una semplice operazione di scaling abbiamo una conseguente espansione dell'indeterminazione frequenziale.
Questo principio è alla base della trasformata wavelet, la cui espressione è la seguente:

$$CPWT[x](a,b) = \int_{-\infty}^{+\infty} x(t)\frac{1}{\sqrt{a}}\psi^*(\frac{t-b}{a})dt =< x(t), \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) > \tag{2.5}$$

dove a e b sono rispettivamente la scala e la collocazione temporale a cui fa capo la specifica rappresentazione. Il prototipo $\psi(.)$ prende il nome di *wavelet madre*. Poiché i valori assunti da a e b sono numeri reali qualsiasi, questa trasformata prende il nome di **Continous Parameters Wavelet Transform** (**CPWT**) cioè *trasformata wavelet a parametri continui* [1].
Rispetto alla trasformata di Fourier, la wavelet è una proiezione su una differente famiglia di funzioni generatrici dello spazio $L^2(\mathbb{R})$. La CFT impiega infatti la famiglia delle frequenze armoniche complesse $e^{j2\pi f}$, che

sono funzioni indipendenti e quindi costituiscono una propriamente base in $L^2(\mathbb{R})$, la CPWT invece sfrutta le *ondine* $\psi_{ab} = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}$ che sono funzioni interdipendenti e dunque un insieme generatore ridondante dello spazio delle funzioni quadrato sommabili.

Si noti che la $\psi(.)$ effettivamente non può essere una funzione qualsiasi ma deve essere appunto un'**ondina** (o **wavelet**). Devono sussistere cioè entrambe le seguenti condizioni:

$$\Psi(0) = \int_{-\infty}^{\infty} \psi(t)dt = 0 \tag{2.6}$$

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(f)|^2}{|2\pi f|^2} df < \infty \tag{2.7}$$

che impongono alla $\psi(.)$ una media nulla ed un decadimento rapido in frequenza all'infinito. Queste due proprietà messe insieme comportano per lo *spettro di Fourier* $\Psi(.)$ della $\psi(.)$ un comportamento di tipo *passa banda*. La trasformata wavelet dunque finestra il segnale sia nel tempo che in frequenza, così come la STFT. La differenza sta nel fatto che tramite la wavelet possiamo mappare lo spazio tempo-frequenza con celle di dimensione appropriata rispetto al segnale da analizzare, diverse a seconda della frequenza e del tempo presi in considerazione. Le mappature sono possibili e sovrapponibili, data la ridondanza dell'insieme generatore.

2.2.4 Trasformata wavelet a parametri discreti

Tra le tante mappature possibili, una particolarmente utile nelle applicazioni è quella *diadica* mostrata in figura 2.2. Dell'intera famiglia $\psi_{ab}(.)$ viene scelto un particolare sottoinsieme di funzioni, cioè di parametri a e b, in modo tale che esse siano una base di $L^2(\mathbb{R})$. La scelta che viene effettuata è la seguente:

$$a = {a_0}^{-m} \tag{2.8}$$

$$b = nb_0a_0^{-m} \tag{2.9}$$

con $a_0 > 1$ e $b_0 \geq 1$. Se $a_0 = 2$ e $b_0 = 1$ si parla appunto di **decomposizione diadica**.
Poiché ora le funzioni di base possono essere indicizzate mediante gli indici discreti m ed n, la trasformata wavelet non ridondante appena introdotta

prende il nome di **Discrete Parameters Wavelet Transform** (**DPWT**) cioè *trasformata wavelet a parametri discreti*.
Si noti che con questa operazione abbiamo discretizzato le funzioni dell'insieme generatore ottenendo un'opportuna base di $L^2(\mathbb{R})$

$$\psi_{mn}(t) = 2^{\frac{m}{2}}\psi(2^m t - n) \tag{2.10}$$

ma non abbiamo ancora realizzato una versione della trasformata per segnali discreti.

2.3 Analisi tempo-frequenza nel discreto

All'atto pratico non si ha a che fare con segnali continui indefiniti nel tempo ma con sequenze discrete temporalmente limitate.

2.3.1 Analisi di Fourier nel discreto

Per quel che riguarda l'analisi di Fourier, il passaggio da tempo continuo a tempo discreto avviene attraverso i seguenti passaggi [1]. Una sequenza discreta $x(n)$ di lunghezza N è vista come una successione di impulsi di Dirac $x(t) = x(n)\delta(t-n)$ nel continuo, equidistanziati di 1 sull'asse temporale. Dal segnale limitato temporalmente viene generato un segnale periodico, con periodo pari alla durata N del segnale originale. La trasformata di Fourier di questo segnale periodico, non è altro che una sequenza di impulsi in frequenza (**Fourier Series** : **FS**) distanziati di $\frac{1}{N}$ sull'ascissa delle frequenze. Poiché il segnale nel dominio del tempo è una successione di impulsi di Dirac equidistanziati, nel dominio della frequenza la rappresentazione sarà periodica di periodo 1. In definitiva possiamo associare ad N campioni di una successione temporale, N campioni in frequenza nel seguente modo:

1. dalla sequenza ricaviamo una successione di impulsi di Dirac;
2. replichiamo il segnale con periodo N;
3. effettuiamo la trasformata di Fourier;
4. consideriamo gli N valori trasformati partendo dalla frequenza 0 alla frequenza $\frac{N-1}{N}$, con passo $\frac{1}{N}$.

La trasformata risultante è detta **Discrete Fourier Transform** (**DFT**) cioè trasformata di Fourier tempo discreta, ed ha la seguente espressione:

$$DFT[x](k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{k}{N} n} \tag{2.11}$$

dove $x(n)$ è la sequenza da trasformare ed $X(k)$ la sequenza trasformata in funzione della frequenza discreta k.
Con un ragionamento analogo è possibile ottenere una versione discreta anche della STFT.

E' interessante notare come la DFT possa essere interpretata, nel paradigma dell'algebra lineare, come una trasformata ortonormale tra i vettori $\underline{x}$ ed $\underline{X}$. La matrice di trasformazione ha come generico elemento:

$$c_{kn} = e^{-j2\pi \frac{k}{N} n} \tag{2.12}$$

La DFT in realtà non è impiegata per la compressione, in quanto, come per la FT, i coefficienti trasformati sono numeri complessi. Per ogni valore della sequenza originaria ne abbiamo dunque due per la sequenza trasformata, la qual cosa non è certo un buon presupposto di efficienza. Risulta utile dunque introdurre trasformate capaci di realizzare un'analisi in frequenza a valori reali. A questo scopo viene ad esempio introdotta la *trasformata coseno discreta*.

2.3.2 Trasformata coseno discreta

Una delle più impiegate trasformate lineari ortogonali è la DCT. La *trasformata coseno discreta* (**Discrete Cosine Transform**, **DCT**) [10] è una trasformazione lineare derivata dalla DFT che, a differenza di quest'ultima, utilizza come vettori di base funzioni coseno piuttosto che esponenziali complessi.
Il processo dal quale la DCT è ricavata a partire dalla FT è quasi identico a quello per ricavare la DFT. La differenza sta nel fatto che il segnale generatore della replica sulla quale viene effettuata la FT non è il segnale di partenza $x(t) = x(n)\delta(t-n)$, ma una sua versione simmetrica costruita nel seguente modo:

$$x(t) = \begin{cases} x(n)\delta(t-n-\frac{1}{2}) & \text{se } t > 0 \\ x(n)\delta(t+n+\frac{1}{2}) & \text{se } t < 0 \end{cases}$$

Dato che la trasformata di Fourier di un segnale simmetrico è reale [1], viene soddisfatto il fondamentale requisito che vi sia un solo valore trasformato per ciascun valore originale.

Anche la DCT può essere rappresentata mediante una trasformazione lineare. L'elemento generico della matrice di trasformazione (supponendo le componenti di $\underline{x}$ numerate da 0 ad $N-1$) è

$$c_{ij} = a(i)\cos(\frac{\pi(2j+1)i}{2N}) \tag{2.13}$$

dove

$$a(i) = \begin{cases} \sqrt{\frac{1}{N}} & \text{se } i = 0 \\ \sqrt{\frac{2}{N}} & \text{se } i > 0 \end{cases}$$

Essa può essere applicata, nel caso di dati bidimensionali, a matrici $N \times N$, usando la sua versione 2D applicata direttamente a blocchi quadrati di dimensioni $N \times N$.
In quest'ultimo caso i coefficienti della matrice quadridimensionale sono espressi da:

$$d_{ih,jk} = c_{ij} \cdot c_{hk} \tag{2.14}$$

Ciò implica che la trasformata è **separabile**, cioè la trasformata bidimensionale si può realizzare operando prima per righe e poi per colonne.
La DCT gode delle seguenti proprietà:

Ortogonalità. La matrice di trasformazione è unitaria, cioè la matrice inversa coincide con la sua trasposta. Questo implica che la trasformata è *isometrica*, cioè conserva la norma nel dominio trasformato, per cui se si misura la distorsione con l'MSE si può operare direttamente nel dominio trasformato.

Bassa complessità computazionale. Date le particolari proprietà di simmetria può essere implementata con un algoritmo veloce ed efficiente. Ad esempio può essere ridotta ad una trasformata veloce di Fourier (*Fast Fourier Transform*, FFT) semplicemente riordinando i termini da trasformare.

Compattazione dell'energia. Pur non essendo ottima è dotata, in genere, di buone proprietà di *compattazione* dell'energia nei suoi primi coefficienti. Questo perché, solitamente i segnali naturali hanno una distribuzione di energia decrescente con la frequenza, cosicchè i vettori base della trasformata coseno sono spesso assai simili a quelli, ottimi dal punto di vista concentrazione dell'energia, della KLT.

Per le sue notevoli proprietà molto spesso la DCT è utilizzata come strumento sub-ottimo al posto della KLT.

2.3.3 Trasformata wavelet tempo discreta

Una *trasformata wavelet tempo discreta* può essere ricavata a partire dalla *Discrete Wavelet Transform*. Essa viene spesso indicata con l'acronimo **DWT**, che sta per **Discrete Time** (Discrete Parameters) **Wavelet Transform.**
Mentre per la *trasformata di Fourier* per una rappresentazione discreta abbiamo dovuto supporre la limitatezza temporale del segnale, nel caso della

trasformata wavelet bisogna imporre una condizione di **risoluzione limitata**. Introdotta la funzione $\phi(t)$, detta *di scaling*, il segnale $x(t)$, rappresentativo della sequenza $x(k)$ nel continuo è definito come segue [6]:

$$x(t) = \sum_{n=-\infty}^{+\infty} x(k)\phi(t-k) \tag{2.15}$$

Il segnale $x(t)$ può essere rappresentato come proiezione sulla sua base DWT $\{\psi_{mn}(t)\}$. La condizione di *risoluzione limitata* (assieme a quella di *limitatezza temporale*) permette tuttavia di rappresentare il segnale attraverso la proiezione su una base di cardinalità finita (composta da wavelet e funzioni di scaling) anziché una base di cardinalità infinita (composta da wavelet soltanto). A causa della risoluzione e dell'estensione limitata $x(t)$ giace infatti in un sottospazio di $L^2(\mathbb{R})$ di dimensione finita. Ciò consente in definitiva la rappresentazione del segnale $x(n)$ con una quantità finita di coefficienti trasformati. I dettagli di quanto descritto saranno chiari al termine del paragrafo successivo dopo aver sviluppato il concetto di *analisi multirisoluzione*.

2.3.4 Nota storica sugli acronimi

Gli acronimi qui introdotti per le varie trasformate wavelet non sono quelli più impiegati in letteratura. Di solito la CPWT è indicata semplicemente come CWT (Continous Wavelet Transform) mentre la DPWT è indicata con l'acronimo DWT (Discrete Wavelet Transform) [1]. La DPWT tempo discreta è indicata a volte con l'acronimo DTWT [6], altre volte anche essa con DWT [1], generando una certa confusione quando il medesimo acronimo è usato anche per la trasformata tempo continua (come in [1]).

2.4 Analisi multirisoluzione

Definizione 1. *Siano gli insiemi $\{V_m\}$ con $m \in \mathbb{Z}$ sottospazi di $L^2(\mathbb{R})$ tali che valgano le seguenti proprietà:*

- $V_m \subset Vm + 1$ *;*
- $\bar{\cup}_{j\in\mathbb{Z}} = L^2(\mathbb{R})$*;*
- $\cap_{j\in\mathbb{Z}} = \emptyset$*;*
- $x(t) \in V_m \Leftrightarrow x(t - n2^m) in V_m$*;*
- $x(t) \in V_m \Leftrightarrow x(\frac{t}{2}) in V_{m-1}$.

*Il sistema $\{V_m\}$ è detto **analisi multirisoluzione** [1][6] di $L^2(\mathbb{R})$.*

Una base ortonormale del sottospazio V_0, è costituita da tutte le traslazioni $\phi_{0n} = \phi_0(t - n)$ di $\phi_0(t)$. $\phi_0(t)$ è detta **funzione di scaling**. Vale anche che, per ogni m, gli insiemi di funzioni $\{\phi_j(t - \frac{n}{2^m})\}$, dove le $\phi\frac{(t-n)}{2^m} = \frac{1}{\sqrt{2^m}}$ costituiscono delle basi per i rispettivi sottospazi V_m.
Se inoltre definisco come W_m il sottospazio tale che $V_{m+1} = V_m \oplus W_m$, dove con $\oplus$ ho indicato l'operatore di somma diretta, esisterà per W_m una base $\{\psi_j(t - \frac{n}{2^m}) = \frac{1}{\sqrt{2^m}}\psi\frac{(t-n)}{2^m}$ ortonormale tale che $\psi(.)$ è una *wavelet*. In realtà questa ipotesi può essere rilassata e si possono considerare sottospazi W_m e V_m non ortogonali, essendo V_{m+1} semplicemente la loro somma lineare; si parlerà allora di **analisi multirisoluzione biortogonale**. Si parla invece di **analisi multirisoluzione ortogonale** se V_m e W_m sono tra loro ortogonali e V_{m+1} è la loro somma diretta.
L'analisi multirisoluzione permette in pratica di rappresentare un certo segnale $x(t)$ come la sovrapposizione di una sua versione ad una certa risoluzione m^* e le proiezioni nei diversi sottospazi wavelet $m > m^*$:

$$x(t) = \sum_{n\in\mathbb{Z}} < x, \hat{\phi}_{m^*n} > \phi_{0n} + \sum_{m=m^*}^{\infty} \sum_{n\in\mathbb{Z}} < x, \hat{\psi}_{mn} > \psi_{mn} \qquad (2.16)$$

In pratica il segnale è visto come una sua versione a bassa risoluzione più un raffinamento successivo con dettagli a risoluzione via via crescente.
Tra le funzioni di base delle diverse risoluzioni di scaling e dei diversi spazi wavelet esiste una relazione ben precisa.

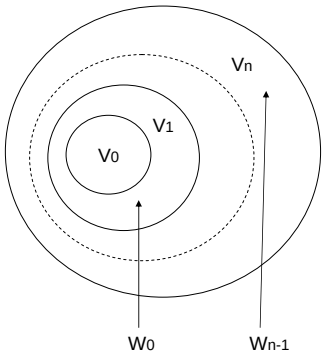


Figura 2.3: Analisi multirisoluzione.

Teorema 1. *Si può dimostrare che valgono le seguenti fondamentali equazioni.*

$$\phi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} h_0(n)\phi(2t-n) \tag{2.17}$$

$$\psi(t) = \sqrt{2}\sum_{n\in\mathbb{Z}} h_1(n)\phi(2t-n) \tag{2.18}$$

dove $h_0(n) =< \frac{1}{\sqrt{2)}}\phi(\frac{t}{2}), \hat{\phi}(t-n) >$ *e* $h_1(n) =< \frac{1}{\sqrt{2}}\psi(\frac{t}{2}), \hat{\phi}(t-n) >$ *con* $\{\hat{\phi}(t-n)\}$ *base duale di* $\{\phi(t-n)\}$.

2.4.1 Implementazione della DWT mediante banco di filtri

Le equazioni della multirisoluzione permettono di ricavare i coefficienti trasformati che rappresentano il generico segnale $x(t)$, ad una certa risoluzione m della funzione di *scaling*, a partire dai coefficienti che rappresentano lo stesso segnale ad una risoluzione di *scaling* $m+1$ immediatamente superiore .

Teorema 2. *Sia*

$$x(t) = \sum_n < x, \hat{\phi}_{0n} > \phi_{0n} + \sum_{m=0}^{\infty}\sum_{n\in\mathbb{Z}} < x, \hat{\psi}_{mn} > \psi_{mn} \tag{2.19}$$

$$= \sum_{n\in\mathbb{Z}} c(0,n)\phi_{0n} + \sum_{m=0}^{\infty}\sum_{n\in\mathbb{Z}} d(m,n)\psi_{mn} \tag{2.20}$$

Vale per i coefficienti della trasformata la seguente relazione [4]:

$$c(m,n) = \sum_k h_0(k-2n)c(m+1,k) \quad (2.21)$$

$$d(m,n) = \sum_k h_1(k-2n)c(m+1,k) \quad (2.22)$$

Questa importante relazione permette di ricavare i coefficienti alle diverse risoluzioni attraverso un **banco di filtri di analisi** così come mostrato in fig. 2.4 (A). Relazioni analoghe permettono di ricostruire i coefficienti $c(m,n)$ relativi ad una risoluzione di *scaling* superiore, partendo dai coefficienti di *scaling* $c(m-1,n)$ e dai coefficienti *wavelet* $d(m-1,n)$ relativi alla risoluzione di un livello immediatamente inferiore. E' possibile dunque passare da una risoluzione più bassa ad una più alta attraverso un opportuno **banco di filtri di sintesi** così come mostrato in fig. 2.4 (S). Iterando diverse volte la struttura del banco di filtri è possibile realizzare l'analisi multi-risoluzione desiderata (vedi fig. 2.5.

Da quanto detto già nel paragrafo 2.3.3 si comprende come in questo modo si possa realizzare la *wavelet tempo-discreta*. Associando alla sequenza $x(n)$ di lunghezza N una funzione a risoluzione limitata $x(t)$ con coefficienti di *scaling* $c(0,n) = x(n)$ è infatti possibile ottenere, attraverso un opportuno banco di filtri, N coefficienti trasformati che rappresentano l'analisi multi-risoluzione di $x(n)$.

Resta da chiarire il legame tra i filtri h_A ed g_A di analisi ed i filtri h_S e g_S di sintesi. E' possibile definire questo legame nel dominio di Fourier imponendo la condizione di prefetta ricostruzione a valle della cascata dei banchi di analisi e di sintesi. Il legame è stabilito dal seguente teorema.

Teorema 3. *La condizione di **perfetta ricostruzione** per i filtri di analisi e di sintesi, è stabilita dalle seguenti due condizioni nel dominio della frequenza:*

$$H_S(f)H_A(f) + G_S(f)G_A(f) = 2 \quad (2.23)$$

$$H_S(f)H_A(f+\frac{1}{2}) + G_S(f)H_A(f+\frac{1}{2}) = 0 \quad (2.24)$$

Scelta la funzione di scaling vi sono diversi gradi di libertà nella scelta della wavelet affinché la perfetta ricostruzione sia possibile, nel caso più generale di *analisi multirisoluzione biortogonale*. Vi sono meno possibilità di

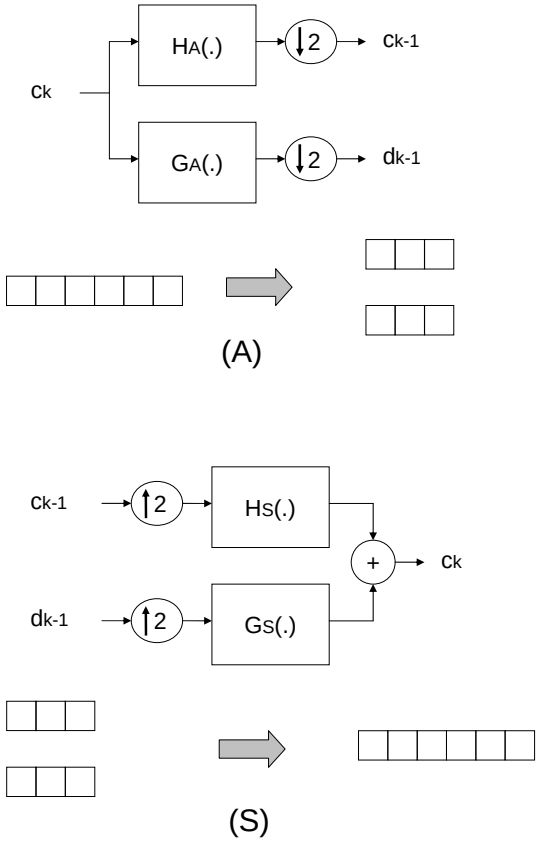
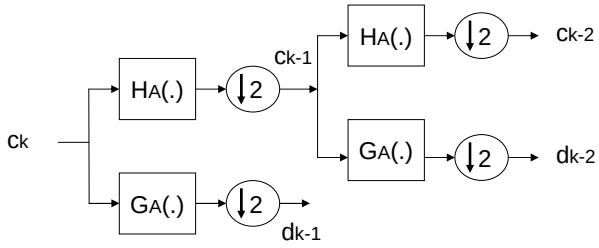
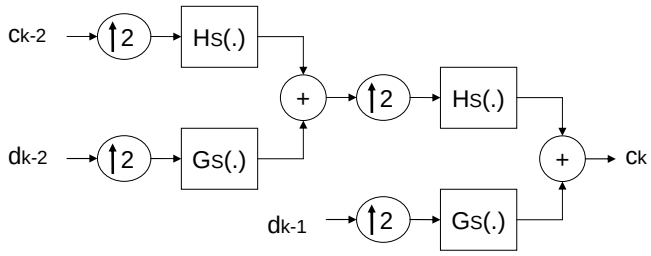


Figura 2.4: Banco di analisi (A) e banco di sintesi (S) per un livello di decomposizione wavelet.



(A)



(S)

Figura 2.5: Banchi di analisi (A) e banco di sintesi (S) per due livelli di decomposizione wavelet.

scelta nel caso in vece di *analisi ortogonale*. Il progetto dei filtri per l'analisi multirisoluzione trascende comunque dallo scopo di questa breve trattazione introduttiva.

2.5 Lifting scheme

Un approccio differente all'analisi multirisoluzione è costituito dalla tecnica dei *lifting scheme*. Questo approccio è stato presentato in [8] e formalizzato in [5] per una realizzazione della DWT alternativa al banco di filtri, più efficiente dal punto di vista computazionale. La teoria dei lifting scheme tuttavia ha trovato un ampio consenso negli ultimi anni anche per i seguenti importanti motivi:

- l'approccio è più intuitivo;
- sono forti i legami con le tecniche di interpolazione più note;
- la condizione di perfetta ricostruzione è implicitamente soddisfatta dalla natura stessa dei *lifting scheme*;
- il significato dei filtraggi è più chiaro ed eventualmente è più semplice l'impiego di trasformate non separabili;
- è possibile estendere il concetto di lifting utilizzando invece del filtraggio lineare (come per la *wavelet*) tecniche non lineari di predizione e di interpolazione.

In questo paragrafo non illustreremo nello specifico il legame tra *wavelet* e *lifting scheme* per cui rimandiamo a [5]. Introduciamo invece la struttura generale del *lifting scheme* in fig. **??** cercando intuitivamente di spiegarne il significato. La generica sequenza x viene dapprima suddivisa in due sottosequenze x_p ed x_d (indici pari e dispari). La sottosequenza di indici pari x_p viene predetta da una versione addolcita mediante il filtraggio $s(.)$ di quella di indici dispari. Alla sottosequenza pari viene sostituito dunque l'errore di predizione così ottenuto. Questo passo è detto appunto di **predizione**. A questo punto la sequenza x_p pari viene modificata con un opportuno filtraggio $t(.)$ dell'errore di predizione in modo da ottenere una sequenza abbastanza rappresentativa sia della sottosequenza dispari x_d che di quella pari x_p. Questo passo è detto di **aggiornamento**. Questa procedura piuttosto semplice ed intuitiva tuttavia è sostituita, nel caso più generale, da un una serie di m passi di *predizione* ed di *aggiornamento* (vedi fig. 2.6) . Infine un'operazione di **normalizzazione** fa sì che le sequenze risultanti, una di trend e una di dettaglio, conservino l'energia delle sequenza originale.

Come già accennato il *lifting scheme*, nella realizzazione di una trasformata wavelet, può sostituire il banco di filtri sia in analisi (fig. 2.6) che in sintesi (fig. 2.7). I vantaggi sono un numero minore di operazioni da effettuare (minor tempo di calcolo) e la possibilità di effettuare calcoli sul posto sovrascrivendo i coefficienti da trasformare con quelli trasformati (minore occupazione di memoria).

2.5.1 Lifting scheme adattativi

I lifting schemes sono stati tuttavia impiegati con successo nello sviluppare schemi di predizione alternativi per l'analisi multirisoluzione. Il filtraggio della fase di *predizione* ha infatti l'obiettivo di addolcire l'andamento della sequenza di riferimento (quella dispari nella nostra trattazione) in modo che la predizione sia più robusta quando si perde la stazionarietà ed un coefficiente della sequenza dispari risulta molto diverso da quello da predire nella sequenza pari. Si può pensare invece di impiegare filtraggi a coefficienti fissi, di impiegare tecniche differenti, di natura adattativa [7], che colleghino in maniera più efficiente il coefficiente da predire a quello di riferimento più adatto alla predizione. Questa operazione può avvenire o soltanto in base ad un'elaborazione dei dati già processati, o in virtù di elaborazioni effettuate anche sui dati ancora da processare. Nel primo caso il banco di sintesi non ha bisogno di nessuna informazione aggiuntiva, così come per il filtraggio non adattativo, nel secondo caso invece vi è bisogno di un'opportuna informazione collaterale (ad esempio come per la compensazione del movimento mediante lifting adattativi nella codifica video [14]). Al fine della codifica mentre la prima tecnica è applicabile nel caso *lossless*, non lo è nel caso *lossy* in quanto le scelte che determinano la particolare predizione sono inquinate dall'errore di quantizzazione. Per questo motivo nel caso *lossy* per le tecniche adattative è necessario l'invio di informazione collaterale, la qual cosa fa perdere gran parte del vantaggio ottenuto. Naturalmente, oltre agli stadi di *predizione* anche gli eventuali stadi di *aggiornamento* vanno opportunamente riprogettati.

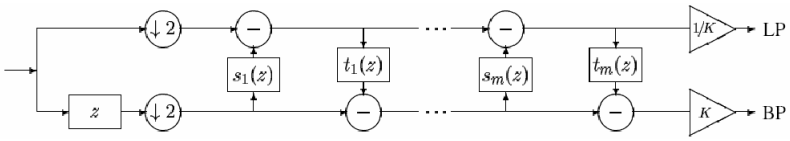


Figura 2.6: Lifting scheme di analisi.

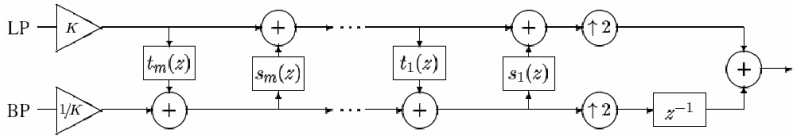


Figura 2.7: Lifting scheme di sintesi.

2.6 Analisi multidimensionale

Nelle applicazioni l'*analisi tempo-frequenza* non sempre è indirizzata a dati monodimensionali o sequenze. Ad esempio è possibile voler analizzare immagini (dati bidimensionali) o sequenze video (dati tridimensionali). Per questo motivo gli strumenti dell'*analisi tempo-frequenza* vanno estesi in maniera opportuna al caso N-dimensionale per cui si parla più in generale di analisi **spazio-frequenza**.
La DFT e la DCT sono **trasformate separabili**, nel senso che la loro versione N-dimensionale si ottiene effettuando separatamente una trasformata monodimensionale lungo ciascuna dimensione. Questa proprietà dipende fondamentalmente dalle funzioni di base (esponenziale complesso, coseno).
La STFT e la DWT possono impiegare diverse funzioni di base. E' possibile nel caso N-dimensionale ricavare facilmente delle basi partendo dal caso monodimensionale. Se effettuiamo semplicemente una STFT o una DWT monodimensionale lungo le N dimensioni otteniamo una STFT o una DWT N-dimensionale. Tuttavia per queste trasformate è possibile impiegare anche funzioni di base non separabili.
Negli ultimi anni l'impiego di *trasformate separabili* è stato messo fortemente in discussione. Sebbene i vantaggi computazionali siano senza dubbio tutt'altro che trascurabili, le *trasformate non separabili* potrebbero adattarsi meglio alla natura multidimensionale dei dati. Poiché l'utilizzo delle *trasformate non separabili* nelle applicazioni è ancora in fase di studio, nell'ambito di questo lavoro di tesi sono state utilizzate le più assestate *trasformate separabili*, anche

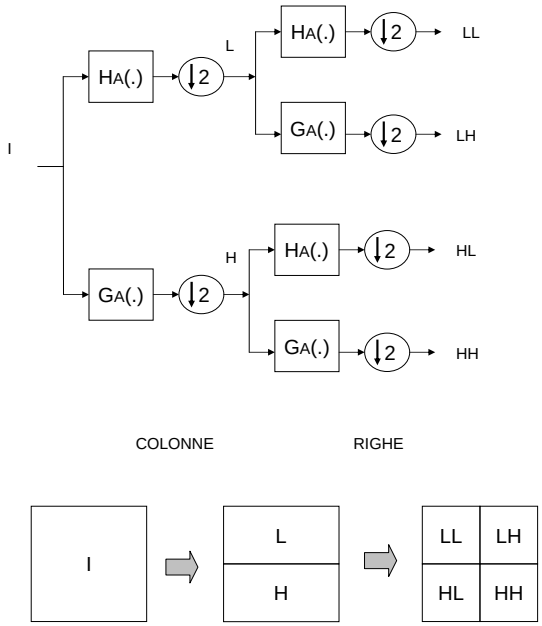


Figura 2.8: Esempio di un livello di decomposizione wavelet bidimensionale ottenuta mediante basi separabili e banchi di filtri sulle colonne e sulle righe di un'immagine.

se i risultati raggiunti sono generalizzabili anche al caso non separabile.

Bibliografia

[1] E. Conte - “Lezioni di teoria dei segnali”, Liguori, 1996.

[2] S. Mallat, “A wavelet tour into signal processing”, Elsevier, 1998.

[3] R. M. Rao, A. S. Bopardikar “Wavelet transform: introduction to theory and applications”, Addison-Wesley, 1998.

[4] M.Vetterli and C.Harley, “Wavelets and Filter Banks: Theory and design”, IEEE Trans. Signal Processing, vol.40, n. 5, pp. 2207-2232, 1992.

[5] I. Daubachies, W. Swedelns, “Factoring wavelet transform into lifting steps”, *Journal on Fourier Analysis Appl.*, vol. 4, n. 3, 1998.

[6] W. Swedelns, “Building your own wavelet at home”, *Wavelets in Computer Graphics*, ACM SIGGRAPH course notes, 1996.

[7] R. L. Claypoole, G. M. Davis, W. Swedelns, G. Baraniuk, “Nonlinear Wavelet Transform For Image coding by Lifting”, *IEEE Trans. on Image Processing*, 2000.

[8] A. Secker and D. Taubman “Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression”, *IEEE Transactions on Image Processing*, vol. 12, n. 12, pp. 1530-1542, dicembre 2003.

[9] J. Solé Rojials, “Optimization and generalization of lifting schemes: application to lossless image compression”, Tesi di Dottorato, Univ. Politecnica della Catalogna, 2006.

[10] K. Sayood “Introduction to data compression”, *Morgan-Kaufmann*, 2000.

Capitolo 3

Codifica di segnali visuali

Parole chiave

Nozioni propedeutiche

Codifica mediante trasformata, DCT, DWT, quantizzazione vettoriale, metodi di codifica lossless, curve tasso-distorsione, metodi di allocazione delle risorse.

Concetti introdotti nel capitolo

Immagini monocromatiche, immagini multicanale, sequenze video, standard di codifica, qualità visiva, trasformate a blocco, trasformate dispersive, organizzazione per blocco, organizzazione per sottobanda, stima e compensazione del movimento, JPEG, JPEG2000, famiglia MPEG, famiglia H26X.

3.1 Segnali visuali multidimensionali

I **segnali visuali**, cioè quelli interpretabili come un'immagine o un'insieme di immagini dal sistema visivo umano, sono, allo stato della tecnologia attuale, quasi esclusivamente trattati in digitale. L'alaborazione di segnali visuali non è dunque altro che una branca dell'elaborazione dei segnali digitali.
La più semplice struttura dati visualizzabile è costituita da un'**immagine monocromatica**, che, nella sua rappresentazione digitale, non è altro che una matrice di elementi discreti (**pixel**), i quali possono assumere valori arbitrari in un dynamic range e con una risoluzione prefissata.
L'*immagine monocromatica*, così definita, ha una struttura bidimensionale. Tuttavia, già nella nostra esperienza visiva quotidiana, possiamo comprendere come esistano segnali visuali a più di due dimensioni. L'occhio umano ad esempio è sensibile a ben tre canali di colore (rosso, verde e blu), per cui già semplicemente per rappresentare quella che per un essere umano è considerata comunemente un'immagine, abbiamo bisogno di ben tre *immagini monocromatiche*, una per ciascun canale RGB. Questo è soltanto il più banale esempio di **immagine multicanale**. Vi sono infatti, nelle realizzazioni tecnologiche, sensori dotati di ben più di tre canali. Ad esempio nelle applicazioni medicali, nella geotecnica, nell'analisi dei materiali e nel telerilevamento sono impiegati spesso sensori che acquisiscono simultaneamente diverse bande elettromagnetiche. Si parla allora di sensori multispettrali o iperspettrali e conseguentemente di *immagini multispettrali* ed *iperspettrali*, che non sono altro che particolari immagini multicanale.
Un'altro tipo di aumento della dimensionalità può essere causato dalla dimensione tempo. E' il caso delle sequenze video in cui si succedono diversi **fotogrammi** o **frame** (immagini monocromatiche o a colori) ad una frequenza di campionamento fissata (**frame rate**). In questo caso abbiamo strutture dati a tre o quattro dimensioni.
Le dimensioni possono ancora aumentare se si considerano immagini di una stessa scena acquisita da diversi punti di vista. Si parla in questo caso di **viste multiple**. La dimensione *vista* si può aggiungere alla dimensione *tempo* e a quella *spettro* per dare origine a dati a ben cinque dimensioni.
Altri tipi di dati, come la *tomografia*, possono catturare la natura tridimensionale degli oggetti. Ogni immagine rappresenta questa volta la sezione di un'oggetto ad una certa ascissa lungo un asse longitudinale. Si parla in questo caso di **immagini spaziali**.
Di tutti questi tipi di rappresentazione visuale, nell'ambito dell'esperienza di dottorato ci si è concentrati soprattutto sulle immagini multispettrali (in par-

ticolare quelle telerilevate) e sulle sequenze video. In seguito dunque approfondiremo la trattazione del caso in cui la dimensionalità aumenta nella direzione spettrale o in quella temporale.

3.1.1 Immagini monocromatiche

La nostra definizione di immagine monocromatica è applicabile a qualsiasi matrice. Infatti nulla vieta di visualizzare una matrice generica come un'immagine. Tuttavia nelle applicazioni reali, le immagini acquisite dai sensori presentano peculiarità e somiglianze statistiche tali che è inevitabile parlare di **immagini naturali**, distinguendo le stesse da una qualsiasi matrice generata dall'uomo.
Non vi è una definizione rigorosa di *immagine naturale* né per essa sono state definite dalla comunità scientifica proprietà specifiche. Lungi da voler fornire un elenco di caratteristiche oggettive, cercheremo comunque di fare delle considerazioni utili per comprendere le strategie di codifica tipicamente impiegate per le immagini naturali.

Per prima cosa si osservi la fig. 3.2, che non è altro che una figura quantizzata a soli 16 toni dell'immagine a 256 livelli di grigio di fig. 3.1. Si noti come l'immagine è ancora abbastanza simile a quella originale. Ora che l'immagine è praticamente suddivisa in 16 regioni omogenee e tra loro distinte, risulta chiaro come una prima approssimazione di un'immagine naturale sia un segnale *omogeneo per regioni*, qualcosa di simile ad un segnale *costante a tratti* nel caso monodimensionale. Un'immagine naturale rivela dunque un comportamento di bassa frequenza su ampie **regioni**, un comportamento discontinuo lungo i **bordi** di queste aree omogenee ed una sovrapposizione generalizzata e spesso poco significativa di frequenze intermedie (**texture**), che variano solitamente da regione a regione.

Ora soffermiamo l'attenzione su due particolarità: la *correlazione tra le fasi di versioni campionate* e la *correlazione tra sottocampionamenti a scale differenti*.

Per prima cosa proviamo a riorganizzare i *pixel* di un'immagine naturale come in fig. 3.3. Non abbiamo fatto altro che raggruppare i *pixel* dell'immagine iniziale in 16 immagini più piccole, considerando di volta in volta solo i *pixel* con coordinate $(4n + r, 4m + c)$, dove per ciascuna immagine r e c sono fissati e compresi tra 1 e 4. Le immagini più piccole sono ottenute dunque da un sottocampionamento 4×4 uniforme (con fase diversa r e c)

Figura 3.1: L'immagine naturale *Lena*.

Figura 3.2: La versione quantizzata dell'immagine originale di fig. 3.1.

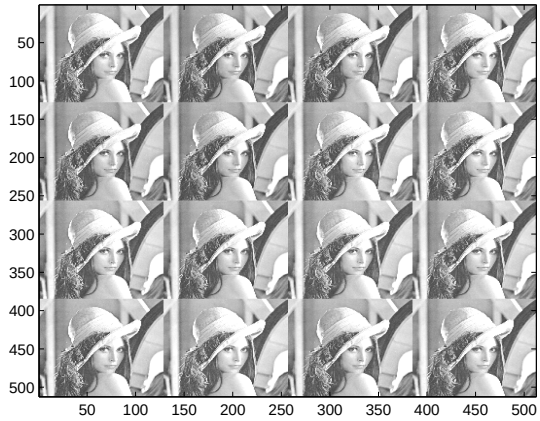


Figura 3.3: Immagine campionata 4×4 a diverse fasi.

dell'immagine originale. Si vede che le varie immagini campionate sono tra loro estremamente simili. Ciò significa che le frequenze significative di un'immagine naturale sono piuttosto basse e che un primo tipo di correlazione che sussiste è proprio quello tra le diverse fasi delle immagini sottocampionate.

Si osservi invece la fig. 3.5. Questa volta abbiamo ottenuto prima 4 campionamenti 2×2 allo stesso modo che nell'esempio precedente (vedi fig. 3.4), poi abbiamo effettuato un ulteriore campionamento 2×2 sulla prima sotto-immagine. Così facendo abbiamo ottenuto sottocampionamenti a differenti scale ($1 : 2$ e $1 : 4$). Dalla somiglianza delle immagini risultanti possiamo renderci conto che non solo vi è una correlazione tra immagini campionate alla stessa scala, ma anche tra immagini afferenti a scale diverse.

Queste due osservazioni suggeriscono due possibili analisi in frequenza: un'analisi in frequenza a risoluzione spazio-frequenza fissa oppure un'analisi in frequenza a risoluzione variabile.

3.1.2 Immagini multicanale

La più comune tra le immagini multicanale è una semplice immagine a colori. Le componenti cromatiche di un'immagine a colori solitamente non sono processate indipendentemente. Su di esse viene invece effettuata una trasformazione spettrale a coefficienti fissi (sul vettore costituito dai *pixel* omologhi di ciascuna componente cromatica) che porta dallo spazio RGB a quello di

Figura 3.4: Immagine campionata 2×2 a diverse fasi.

Figura 3.5: Immagini campionate a diverse fasi e a diverse scale.

luminanza, *tinta*, *saturazione*. Questa trasformazione permette di distinguere la componente di *luminanza* (a cui è più sensibile il sistema visivo umano) da quelle di *crominanza*, consentendo di operare diversamente sulle singole componenti a seconda della loro importanza [3].
Questa strategia è generalizzabile a tutte le immagini multicanale. Le componenti di un'immagine multicanale infatti sono tra loro fortemente correlate. La ridondanza informazionale può essere esplicitata mediante una particolare trasformazione, adattata ai dati (come la KLT), se non si conosce esattamente la natura della correlazione, oppure a coefficienti fissi, se il modello della correlazione spettrale e noto.
Un caso di studio di notevole interesse per questo lavoro di tesi è quello delle *immagini multicanale telerilevate* [23]. In esse ciascun canale rappresenta la risposta del sensore in una particolare banda di frequenza elettromagnetica. Le immagini telerilevate multicanale si distinguono in **multispettrali** (bande di acquisizione larghe nello spettro elettromagnetico, di modesta quantità, tra loro nettamente distanziate) ed **iperspettrali** (bande strette, numerose e a tratti contigue). Sia nelle immagini multispettrali che in quelle iperspettrali vi è una forte correlazione tra i canali, identificati d'ora in avanti con le **bande** di frequenza elettromagnetica ad essi relativi. Mentre però nel caso multispettrale le immagini delle bande sono nettamente diverse le une dalle altre, nel caso iperspettrale la variazione è graduale man mano che si sale nello spettro elettromagnetico, avendo solo qualche discontinuità piuttosto netta in alcuni punti della dimensione spettrale.
Una caratteristica interessante delle immagini multicanale è che ogni oggetto è caratterizzato da un proprio colore. Anche nelle immagini telerilevate possiamo riconoscere particolari colori spettrali che dipendono principalmente, nel caso ottico, dal tipo di terreno presente in una certa regione dell'immagine.

3.1.3 Sequenze video

Le sequenze video sono successioni di immagini campionate temporalmente ad un certo frame-rate, che comunemente è superiore ai 25 Hz. In alcune applicazioni il frame-rate può essere più basso mentre in casi eccezionali (non finalizzati alla visione umana) può anche essere superiore ai 30 Hz.
I *fotogrammi* successivi di una sequenza sono di solito molto simili gli uni agli altri, differenziandosi solo per uno spostamento minimo degli oggetti ripresi o del sensore di acquisizione, o in altri casi per una certa variazione delle condizioni di illuminazione.
Lo spostamento relativo degli oggetti ripresi rispetto al sensore può essere

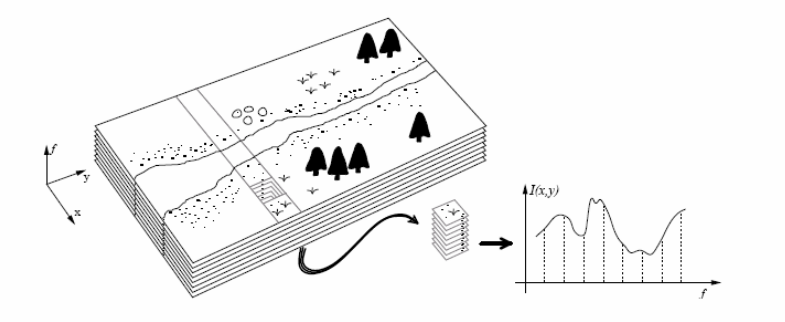


Figura 3.6: Immagine multispettrale.

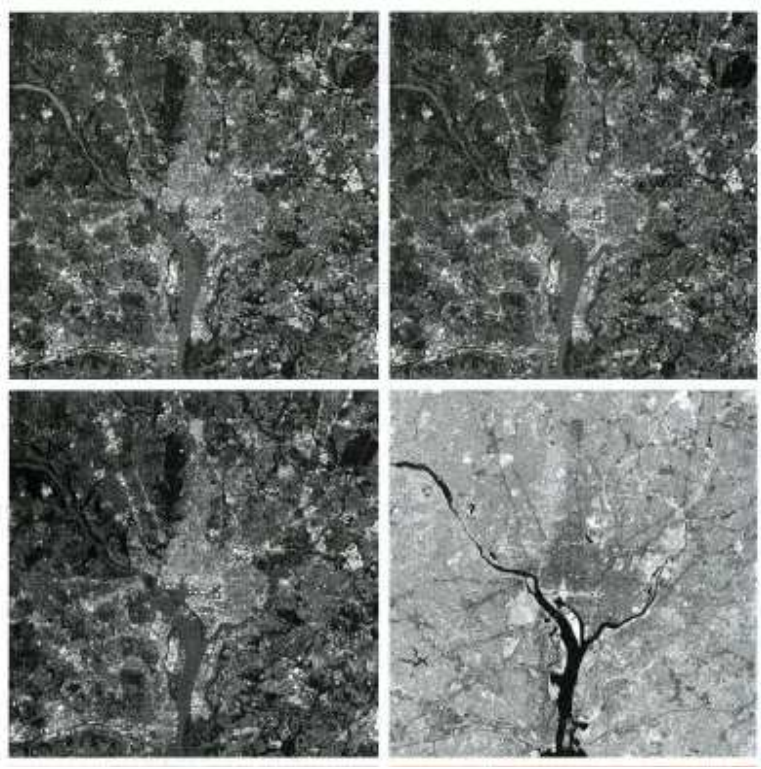


Figura 3.7: Canali (*bande*) di un'immagine multispettrale.

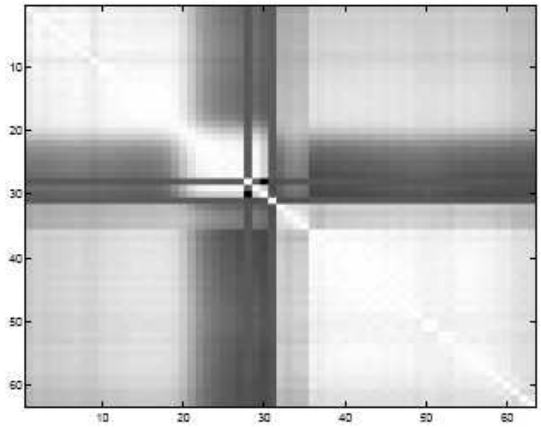


Figura 3.8: Matrice di correlazione tra i canali di un'immagine iperspettrale.

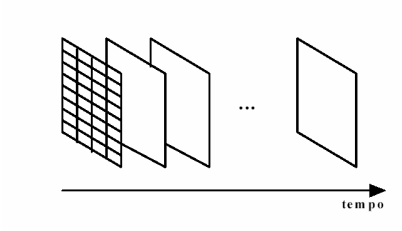


Figura 3.9: Sequenza video.

modellato per evidenziare meglio la somiglianza tra regioni di diversi fotogrammi (successivi o comunque temporalmente vicini). Le tecniche di allineamento locale tra fotogrammi vengono utilizzate per diversi tipi di elaborazione (rilevazione del moto, stima del moto, stima di deformazioni) ed anche per la codifica video.

3.2 Problematiche generali di codifica

Il trattamento di segnali visuali, a prescindere dalla loro dimensionalità, presenta delle problematiche comuni.
Innanzitutto gli schemi di codifica sono tra loro abbastanza simili, cambia invece la particolare strategia che vien utilizzata di volta in volta per modellare i dati e che è fortemente dipendente dalle statistiche dei dati stessi nelle applicazioni reali.
Nella codifica *lossy* un problema molto sentito è quello della valutazione della qualità del segnale ricostruito. La qualità visuale dipende fortemente dalle applicazioni, ma, per ora, soltanto misure semplici ed oggettive come l'errore quadratico medio sono impegnate nelle strategie di codifica.
Lo scopo di un codificatore non è semplicemente quello di compattare o comprimere i dati. Esso deve molto spesso garantire anche alcune *funzionalità*: l'accesso diretto a solo una parte dei dati, la possibilità per i dati di essere codificati da utenti con diverse risorse o diverse esigenze. Vedremo in seguito quali sono le *funzionalità* considerate più importanti.
Infine, quando possibile, per la codifica dei segnali visuali si cerca di convergere verso uno standard. L'obiettivo dello standard è permettere uno scambio dei dati tra i diversi sistemi e le diverse applicazioni quanto più semplice possibile, attraverso sintassi nota e strumenti di codifica universali. Rimane comunque al progettista, fissati gli strumenti e la sintassi, scegliere le strategie di ottimizzazione, la qual cosa può fare ancora la differenza tra le diverse implementazioni, per quanto conformi allo stesso standard di riferimento.

3.2.1 Schema generale di codifica

Uno schema di codifica tipico si compone come segue (fig. 3.10)

- vengono eventualmente stabiliti, in base alle statistiche dei dati, i parametri di un modello di riferimento scelto per esplicitare le dipendenze statistiche dei dati;

- i parametri del modello vengono codificati senza perdita di informazione;

- i dati da codificare sono processati da un blocco di organizzazione che impiega suddetto modello;

- una volta riorganizzati, attraverso una elaborazione reversibile, i dati vengono quantizzati (se la codifica e lossy);
- i dati, eventualmente quantizzati, vengono ora codificati senza perdita di informazione.

Una questione fondamentale, nei sistemi lossy, è quella dell'allocazione delle risorse. Il sistema di allocazione delle risorse è qualcosa che supervisiona tutto il processo di codifica. I dati vengono partizionati in sottoinsiemi elementari, a cui il sistema di allocazione assegna un certo quantitativo di bit per la codifica dal budget complessivo disponibile. Una figura di merito è utilizzata per misurare o stimare la distorsione. I risultati in termini di tasso-distorsione dipendono da vari fattori. Il sistema di allocazione può assegnare le risorse

- attraverso **un'analisi a priori** basata sulle statistiche calcolate o su tutto l'insieme dei dati da codificare, o solo su una porzione di essi, oppure ancora su dati affini già precedentemente codificati;
- in base ad un'**analisi in itinere** di tipo *greedy*, in cui si decide ad ogni passo di codifica a quale insieme assegnare nuove risorse di quelle ancora disponibili;
- a valle di una codifica vera e propria in cui si è raggiunto un tasso molto più alto di quello desiderato, per cui si calcola il troncamento ottimo dei flussi di codifica relativi alle diverse partizioni dell'insieme iniziale attraverso un'**analisi a posteriori**.

Le tecniche *a posteriori* sono quelle indubbiamente più potenti ma anche le più onerose. Una buona analisi *a priori* può invece superare le prestazioni di una *in itinere*.
La bontà del risultato ottenuto non dipende soltanto dal tipo di allocazione ma anche da quanto fittamente è stato partizionato l'insieme iniziale. Più fitta è la suddivisione, migliori sono solitamente i risultati.

La decodifica avviene in maniera del tutto simmetrica. I parametri del modello però questa volta non necessitano più di essere calcolati ma devono essere semplicemente decodificati. Allo stesso modo l'allocazione delle risorse viene effettuata dal codificatore mentre il decodifiocatore ne è del tutto all'oscuro. Ciò comporta che non è necessario al decodificatore sapere come i parametri sono stati calcolati o come le risorse sono state assegnate.
Questo schema generale si può particolarizzare nel caso, diffuso nelle applicazioni, in cui tutti o solo alcuni parametri del modello non sono desunti dalle

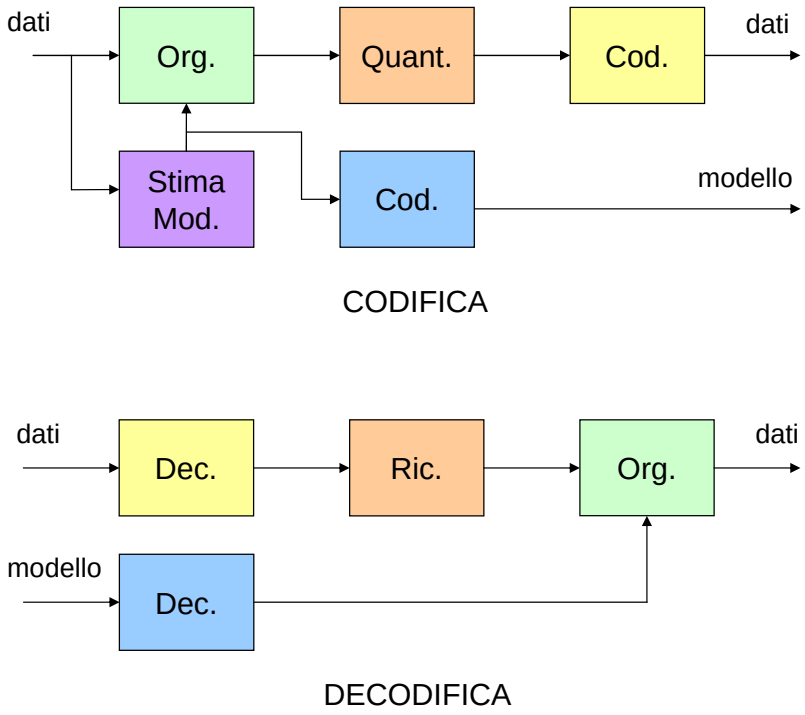


Figura 3.10: Schema tipico di un codificatore e di un decodificatore.

statistiche dei dati ma sono stabiliti a priori. In questo caso il modello può essere meno adattativo ma in compenso non è necessario codificare i parametri.

3.2.2 Figure di merito per la qualità di ricostruzione

Gli algoritmi di codifica hanno bisogno di figure di merito sia per valutare la qualità della codifica a valle della ricostruzione, sia per guidare la codifica stessa verso il risultato migliore.
A differenza che per l'audio, dove gli studiosi sono riusciti a modellare in maniera piuttosto soddisfacente la risposta in frequenza dell'orecchio umano, non esistono modelli analitici o numerici convincenti del sistema visivo. Per quanto alcuni modelli siano stati fino ad ora proposti, gli studi hanno ampiamente dimostrato i modesti miglioramenti rispetto a figure di merito davvero molto semplici e del tutto generiche come ad esempio il *rapporto segnale-rumore*, SNR.
Una misura più ampiamente utilizzata è il **rapporto segnale-rumore di picco**,

o PSNR, in cui alla deviazione standard si sostituisce il dynamic range della profondità di colore DR:

$$\mathrm{PSNR} = \frac{\mathrm{DR}}{\mathrm{MSE}} \tag{3.1}$$

Un'altra possibile misura è l'inverso dell'errore quadradico medio. Solitamente queste grandezze vengono tutte espresse in decibel (dB):

$$F_{dB} = 10 \log_{10} F \tag{3.2}$$

dove con F si è indicata la figura di merito scelta per la qualità.
Quando le distorsioni sono piccole una loro valutazione tramite MSE porta a buoni risultati anche dal punto di vista della percezione. Per tassi di codifica più ridotti, quando si ha a che fare con distorsioni più grandi, si scopre invece che, anche a parità di distorsioni misurate con uno dei metodi visti, si ottengono risultati che dal punto di vista percettivo sono molto differenti. In definitiva il PSNR, e le altre misure basate sull'MSE, si possono impiegare con difficoltà trascurabili soltanto nella seguente casistica:

- qualità visiva (soggettiva) piuttosto buona;
- confronti tra elaborazioni sull'immagine molto simili tra loro (parametri diversi per lo stesso algoritmo, codificatori conformi allo stesso standard etc.);
- confronti tra elaborazioni con risultati qualitativi marcatamente differenti.

In sostanza il PSNR, e in generale tutte le misure oggettive, sono sicuramente utili a scegliere i parametri di un algoritmo o a confrontare le prestazioni tra codec della stessa famiglia, ma sono del tutto inadeguate se si vuol misurare il degrado visivo dell'immagine per l'occhio umano in senso assoluto.
Per questo motivo è stato necessario per gli studiosi del settore e per i responsabili della standardizzazione delle procedure di trattamento e di codifica di immagini, nel caso applicativo assai diffuso in cui l'ultimo passo di elaborazione è semplicemente la visione umana, far ricorso a dei procedimenti di valutazione psicologica e soggettiva della qualità [3]. Il principale riferimento bibliografico in cui sono descritti i più accreditati metodi standard di valutazione soggettiva della qualità visiva, in particolare nel caso delle sequenze video, è l'ITU-R BT.500-11 [4]. In esso sono descritte delle procedure tra di loro abbastanza simili, secondo cui una giuria di esperti, dopo (oppure durante)

la visione, simultanea (oppure sequenziale), di più sequenze video, è chiamata a dare una votazione assoluta (oppure comparativa) di carattere soggettivo [5].

Impiegando questi sistemi di valutazione, si scopre che le figure di merito oggettive e la percezione soggettiva della qualità non portano esattamente agli stessi risultati, anche se è rilevabile una correlazione, così come mostrato in fig. 3.11.

Per impieghi diversi da quello della visione umana (telerilevamento, diagnostica per immagini, visione artificiale per la robotica, etc.) altri tipi di misure sono state introdotte, sostanzialmente allo scopo di valutare la qualità dei dati non a priori ma nei confronti dell'elaborazione finale. Ciò a conferma che la versatilità di impiego di un'immagine rende praticamente impossibile una valutazione che prescinda dalle sue specifiche finalità d'utilizzo.

A favore dell'MSE c'è da dire che esso, fino ad ora, è l'unica figura di merito realmente impiegabile ed impiegata nella definizione degli obiettivi dei vari algoritmi di codifica. Seppure altri criteri soggettivi ed oggettivi sono stati indicati per misurare la qualità di ricostruzione a valle della codifica, sia nelle applicazioni che nella letteratura scientifica, le tecniche di compressione non fanno altro che impiegare una strategia, diversa da caso a caso, di minimizzazione dell'MSE.

3.2.3 Le funzionalità

Come accennato, un sistema di codifica spesso deve soddisfare altri requisiti, oltre che le buone prestazioni in termini di compattazione o di curve tasso-distorsione. Questi requisiti vengono denominai **funzionalità** [23]. Funzionalità particolarmente importanti sono le seguenti.

Accesso diretto. Si può ritenere utile accedere a parte dei dati senza dover decodificare l'intero flusso di codifica.
Si pensi ad esempio a quando, nella visione di un film in DVD, si vuole accedere ad un particolare capitolo o ad una particolare scena, senza dover vedere il film fin dall'inizio.
A seconda dell'applicazione si può richiedere:

- *accesso diretto spaziale*;
- *accesso diretto temporale*;
- *accesso diretto spettrale*;
- *accesso diretto alla vista*.

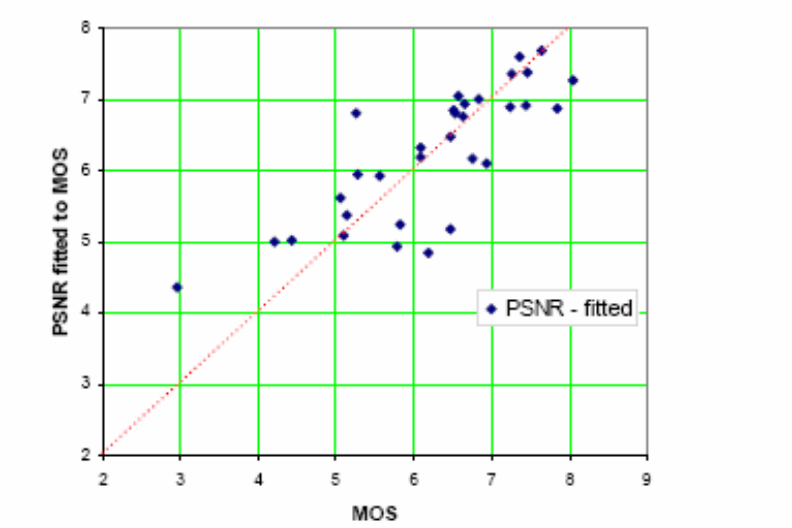


Figura 3.11: Correlazione tra PSNR ed una misura soggettiva di qualità in test effettuati su alcune sequenze video [5].

Scalabilità. Gli stessi dati possono esser richiesti da utenti con diverse risorse (diversi dispositivi di decodifica, diversa larghezza di banda su canale, diversi sistemi di visualizzazione) o comunque con esigenze eterogenee. A questo punto si può pensare o di codificare il contenuto in maniera diversa a seconda della specifica esigenza (con ridondanza di banda su canale o di memoria), oppure di effettuare una singola codifica scalabile, nel senso che l'utente, manifestata la propria esigenza e messo in condizione di poter decodificare solo quella parte del flusso di codifica a cui è realmente interessato, senza dover accedere alle parti che non vuole o non può utilizzare.
Si pensi ad esempio ad un'applicazione di TV broadcast indirizzata sia alla televisione convenzionale, sia alla televisione ad alta definizione (HDTV), sia ad i video-cellulari. La risoluzione in *pixel* dei monitor di questi dispositivi è completamente differente. Se si volessero trasmettere tre video, uno per ciascun tipo di dispositivo, sarebbe necessaria una banda su canale ben più ampia di quella necessaria alla trasmissione del video più complesso (quello per HDTV), opportunamente suddivisa in tre parti in maniera tale che ciascun dispositivo possa decodificare solo la quantità di dati da esso realmente gestibile.

Le forme di scalabilità possono essere differenti, a seconda del contenuto visuale da codificare. Si può pensare ad esempio a:

- *scalabilità spaziale*: diversa risoluzione in *pixel*;
- *scalabilità temporale*: diverso frame-rate;
- *scalabilità spettrale*: diversa quantità di canali spettrali a disposizione;
- *scalabilità spettrale*: diversa quantità di viste;
- *scalabilità in qualità*: diversa qualità di ricostruzione.

3.2.4 Gli standard

Gli obiettivi della standardizzazione di un sistema di codifica sono diversi. Da un lato l'interesse di tutti è la convergenza verso una stessa tecnologia che permetta ai produttori di hardware e di software di utilizzare pochi strumenti, robusti, efficienti e ampiamente testati. Dall'altro negli standard confluisce il lavoro di menti brillanti di diverse provenienze, che riesce a garantire soluzioni avanzate e di successo.
Vediamo in breve cos'è uno standard di codifica. Uno standard definisce gli strumenti di codifica da utilizzare ed la sintassi del flusso di codifica [23]. In pratica lo standard definisce univocamente solo il decodificatore di un sistema di codifica. Il codificatore gode infatti di due principali gradi di libertà:

- come assegnare i parametri al modello di codifica (sia nel caso *lossless* che in quello *lossy*);
- come allocare le risorse tra le diverse componenti informazionali del segnale da comprimere (naturalmente solo nel caso *lossy*).

Gli standard, oltre a definire la sintassi della codifica, indicano anche come implementare e descrivere sintatticamente le funzionalità più importanti. Inoltre, come nel caso del video, definiscono spesso anche come codificare altri media associati ai segnali visuali, ad esempio l'audio.

3.3 La codifica delle immagini monocromatiche

Le *immagini monocromatiche* sono gli elementi fondamentali di qualsiasi sorgente visuale. Per questo motivo le strategie di codifica delle *immagini monocromatiche* sono alla base di tutte le tecniche di codifica di segnali visuali più complessi.
Lo schema più diffuso per la codifica lossy di immagini monocromatiche è quello basato su trasformata. Le trasformate più impiegate non sono adattate alle statistiche dell'immagine (come la KLT) ma sono a coefficienti fissi (come DCT e DWT). Sicuramente l'impiego di filtraggi prestabiliti ed isotropi (non direzionali) non permette di modellare nella maniera più adeguata la natura non stazionaria (o meglio *stazionaria per regioni*) delle immagini. D'altra parte il modello di organizzazione dei dati è in questo modo estremamente semplice e non devono essere codificati parametri aggiuntivi.

Le *trasformate* si distinguono per diverse caratteristiche.
Una prima caratteristica è la *dispersività*. Una trasformata è **non dispersiva** se esiste una porzione di dati, detta *blocco*, tale che la trasformazione T dall'immagine X all'immagine Y è esprimibile nel seguente modo:

$$Y = T(X) = T(\cup_i B_i) = \cup_i T(B_i) \tag{3.3}$$

dove $B_i \cap B_j = \emptyset$ per ogni coppia (i, j). Questo tipo di trasformata è solitamente implementata separando l'immagine in blocchi elementari, per cui è detta **trasformata a blocco**. Quando viceversa $B_i \cap B_j \neq \emptyset$ la trasformata è detta **dispersiva**.
Una seconda caratteristica è il tipo di *analisi spazio-frequenza*. Essa infatti può essere **monorisoluzione** (stessa risoluzione per tutte le coordinate spaziali e per tutte le frequenze) oppure **multirisoluzione** (risoluzione diversa a seconda delle coordinate spaziali e delle frequenze considerate). L'analisi multirisoluzione a sua volta può essere *diadica* (come ad esempio mostrato per la DWT diadica nel capitolo precedente) o *irregolare* (**packet**) [1].
Un'ultima distinzione si può fare tra trasformate *separabili* (scomponibili in trasformate monodimensionali lungo righe e colonne) o *non separabili*. Le prime sono quelle di impiego più diffuso nelle applicazioni, mentre le seconde sono per ora diffuse in maniera trascurabile.

La trasformata a blocco più impiegata è la DCT. Essa è solitamente utilizzata in analisi monorisoluzione. Su essa si basa lo standard per codifica di immagini JPEG.
La trasformata dispersiva più impiegata è invece la DWT ed è utilizzata essen-

zialmente per analisi multirisoluzione. La DWT è il principale strumento del più recente standard di codifica per immagini JPEG2000.

Una volta effettuata l'analisi, sono due le scelte più diffuse di organizzazione dei dati per la codifica lossless.
Una prima soluzione è quella di organizzare i dati in base alla localizzazione spaziale, secondo uno schema di **organizzazione a blocco**. Si organizzano i coefficienti trasformati in blocchi, ognuno dei quali rappresenta l'analisi di una particolare area spaziale dell'immagine originaria. Questo tipo di organizzazione punta a considerare non la correlazione spaziale tra i coefficienti trasformati nelle varie sottobande frequenziali, quanto piuttosto la forma dello spettro frequenziale alle coordinate spaziali considerate. Organizzare i coefficienti in questo modo è particolarmente immediato per le trasformate non dispersive, in cui praticamente ad ogni blocco B_i dell'immagine originale corrisponde una propria analisi in frequenza $T(B_i)$. Nulla vieta però di scegliere la stessa organizzazione anche per trasformate dispersive. In questo caso al blocco B_i corrisponderà un'analisi $B'_i = T(B_i \cup \Delta_i)$, dove B'_i ha le stesse dimensioni di B_i e Δ_i è un *vicinato* di B_i costituito da tutti i *pixel* che concorrono a calcolare B'_i.
Una soluzione alternativa è invece l'**organizzazione per sottobande**. I coefficienti vengono questa volta raggruppati i sottoimmagini in maniera tale che ognuna sia costituita da elementi di una particolare sottobanda frequenziale. Questo tipo di organizzazione mira ad esplicitare la correlazione spaziale per i *pixel* di una stessa sottobanda.

3.3.1 Soluzioni basate su trasformate a blocco

Le trasformate a blocco presentano il vantaggio di mantenere una forte localizzazione spaziale, con una conseguente semplicità di organizzazione dei dati e di calcolo. La localizzazione spaziale ha tuttavia come contropartita la mancata capacità di cogliere l'andamento spaziale del segnale nella sua continuità. Nel caso della compressione di immagini, a bassi bit-rate, la ricostruzione può soffrire di fastidiosi effetti di bordo lungo i margini dei blocchi quadrati in cui è stata suddivisa l'immagine. Questo effetto, noto come **blocchettamento**, è il principale artefatto visivo della suddetta categoria di trasformate ed è spesso mitigato con opportuni filtraggi in fase di post-elaborazione. E' opportuno evidenziare che l'effetto di blocchettamento non è legato soltanto alle trasformate monorisoluzione ma anche a quelle multirisoluzione. E' infatti una conseguenza della non dispersività.

Figura 3.12: Analisi DCT di un'immagine (blocchi 4×4).

In fig. 3.12 sono mostrati i coefficienti a valle di un'analisi DCT con blocchi 4×4 organizzati per sottobande.

3.3.2 Lo standard JPEG

Lo standard JPEG impiega come strumento di analisi la DCT bidimensionale effettuata su blocchi 8×8. I coefficienti trasformati sono disposti secondo un'organizzazione a blocco. La fase di quantizzazione prevede, eventualmente a valle di un'opportuna allocazione delle risorse non specificata dallo standard, di assegnare a ciascuna delle 64 frequenze spaziali (i, j) un certo intervallo di quantizzazione I_{ij}. Una quantizzazione uniforme *midtread* viene dunque effettuata su ciascun coefficiente trasformato x, assegnandovi l'etichetta $e(x) = [\frac{x}{I_{ij}} + 0.5]$, dove con $[.]$ denotiamo l'operatore "parte intera". A questo punto viene eseguita una codifica *lossless* sulle etichette mediante una strategia combinata di codifica *run-length* e codifica di *Huffman*. La tecnica di codifica non sfrutta la correlazione spaziale all'interno delle sottobande ma il tipico comportamento frequenziale all'interno di ciascun blocco: energia concentrata soprattutto alle basse frequenze, con pochi valori diversi da 0 alle alte frequenze a valle della quantizzazione.
In JPEG gli unici parametri da codificare sono i 64 coefficienti I_{ij} della matrice di quantizzazione.

3.3.3 Soluzioni basate su wavelet

La trasformata wavelet si è rilevata un potentissimo strumento di analisi anche per la codifica di immagini. I principali vantaggi sono:

- possibilità di analisi multirisoluzione;
- conseguente implementabilità di codificatori scalabili in risoluzione;
- filtraggio dispersivo;
- implementazioni veloci;
- versioni intere per la codifica *lossless*.

L'analisi multirisoluzione si è rivelata sperimentalmente particolarmente efficace nell'analisi di immagini naturali. Molto importante è la funzionalità di scalabilità in risoluzione. A differenza che per i filtraggi non dispersivi è meno efficace invece l'accesso diretto.
Le basi wavelet più impiegate nella codifica di immagini sono separabili (fig. 3.13) e biortogonali [1][6]. La biortogonalità è necessaria se si vogliono filtraggi simmetrici lungo le righe e lungo le colonne. Poiché non c'è ragione di supporre un modello di dispersione asimmetrico che pesi diversamente *pixel* precedenti e successivi, sicuramente la proprietà di simmetria è desiderabile.
La coppia di filtri di *Daubachies 9-7* [1] è la più impiegata nella codifica *lossy*, presentando proprietà simili alla *spline bicubica* [8] (il filtro migliore di interpolazione nel campo dell'elaborazione di immagini). Nell'ambito della codifica *lossless* invece si preferiscono i filtri biortogonali 5-3 [1], i quali sono derivati da un interpolatore *spline bilineare* [8] e hanno la possibilità di essere implementati con coefficienti interi.
Nella decomposizione diadica, quella più impiegata, ad ogni passo la sottobanda di più bassa frequenza viene decomposta con un solo livello di trasformata wavelet. Per tre livelli di decomposizione la sequenza dei risultati delle elaborazioni è quella della fig. 3.14. Una tipica analisi multirisoluzione ottenuta mediante wavelet diadica è rappresentata in fig. 3.15. I coefficienti sono raggruppati per sottobanda.

A valle della trasformata wavelet i coefficienti possono essere codificati in differenti modi. Un approccio di successo è quello degli *zerotree coder* [4] [5]. Questi codificatori organizzano i coefficienti trasformati per blocchi. Ciascun blocco è codificato per bitplane. Un particolare modello, lo *zerotree*, è utilizzato per codificare in maniera efficiente la forma approssimativa dello spettro

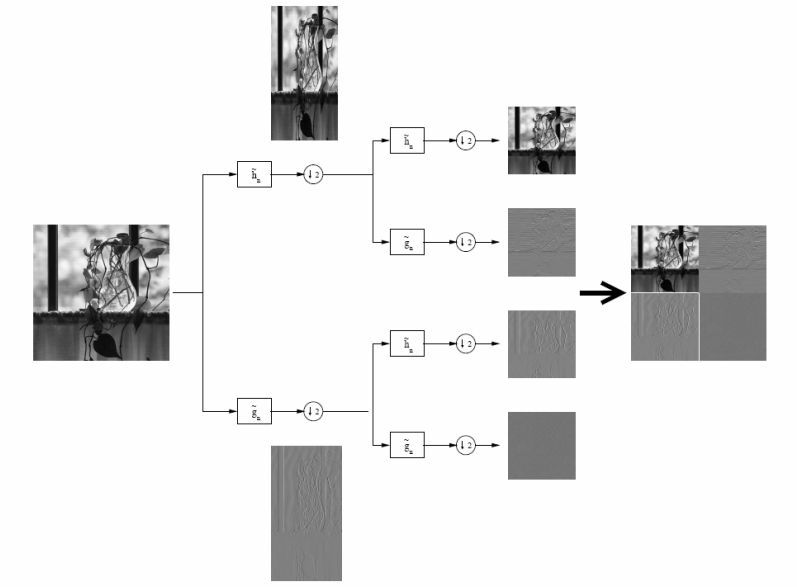


Figura 3.13: Filtraggio wavelet separabile implementato con banchi di filtri.

di ciascun blocco. Il principio è che, una volta rappresentati i coefficienti in virgola fissa, la posizione del primo bit 1 del modulo, detta informazione di **significatività** e che determina grosso modo l'ordine di grandezza dei coefficienti, è un'informazione altamente correlata. Il modello originario dello *zerotree coder* è specifico per l'analisi multorisoluzione diadica, per quanto sia stato successivamente adattato anche ad altri tipi di analisi multirisoluzione [11].
Le prestazioni dei nuovi codificatori wavelet superano di gran lunga quelli di JPEG in PSNR e in qualità soggettiva.

3.3.4 Lo standard JPEG2000

Il proposito di adottare una tecnica più flessibile e più adattabile a diversi tipi di analisi multirisoluzione e la decisione di voler utilizzare una strategia di allocazione delle risorse più efficace di una semplice codifica progressiva per bitplane, ha portato alla definizione dell'algoritmo EBCOT, *Embedded Block Coding with Optimized Truncation*, per la codifica di coefficienti wavelet [8].
EBCOT è alla base del moderno standard basato su wavelet JPEG2000 [14].
La più importante novità introdotta da JPEG2000 è la filosofia per cui ad

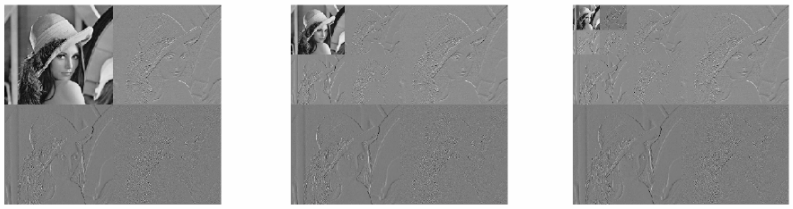


Figura 3.14: Tre livelli di decomposizione successivi.

Figura 3.15: Analisi wavelet diadica di un'immagine (2 livelli di decomposizione).

uno stesso flusso codificato possono corrispondere diverse decodifiche, una prima vera implementazione della scalabilità nel mondo degli standard. Già gli *zerotree coder* avevano dimostrato come tramite la wavelet fosse possibile ottenere scalabilità in qualità (codifica per bitplane) e in risoluzione (analisi multirisoluzione). JPEG2000 riprende questi concetti implementandoli in maniera più eficace. Ulteriori funzionalità proposte dallo standard sono l'accesso diretto, la possibilità di ruotare l'immagine direttamente operando sul flusso di codifica, la codifica di oggetti di forma arbitraria [10].

Il funzionamento di JPEG2000 è semplice. L'immagine viene trasformata mediante DWT. Sono possibili diverse decomposizioni, tra cui anche quella diadica. Dopo la trasformata i coefficienti sono quantizzati e rappresentati in segno e modulo ad altissima risoluzione (codifica *near lossless*). A questo punto agisce uno stadio *lossless* che provvede a codificare la *significatività* mediante opportune strategie predittive, proprio come per gli *zerotree coder*. A differenza che per questi ultimi tuttavia la predizione non è effettuata sulla forma dello spettro ma viene piuttosto sfruttata la correlazione spaziale tra *pixel* vicini all'interno di una stessa sottobanda. La codifica *lossless* avviene separatamente per ciascun *blocco di codifica*, ricavato ritagliando una finestra di *pixel* da una sottobanda specifica. Man mano che la codifica viene effettuata, vengono registrati il tasso corrente e la distorsione. Lo stadio che realmente fa la differenza è quello finale, in cui il sofisticato sistema di allocazione delle risorse di EBCOT tronca in maniera ottima le curve tasso distorsione ottenute procedentemente. In questo modo la codifica *lossy* desiderata non è altro che il troncamento ottimale della codifica *near-lossless* iniziale.

In fig. 3.16 vengono confrontate le prestazioni di diverse implementazioni dello standard JPEG2000 con lo standard JPEG. Innanzitutto si noti il forte gap prestazionale in PSNR (almeno 2 dB ad alto bit rate, e molto di più a tassi inferiori). Come seconda cosa si osservi come diverse implementazioni abbiano diverse prestazioni. Infatti sono lasciati come gradi di libertà per il progettista: il tipo di analisi multirisoluzione, il numero di livelli di decomposizione, i filtri wavelet, la dimensione dei blocchi di codifica.

JPEG2000 prevede anche una versione *lossless*, la cui descrizione però esula dallo scopo di queste note.

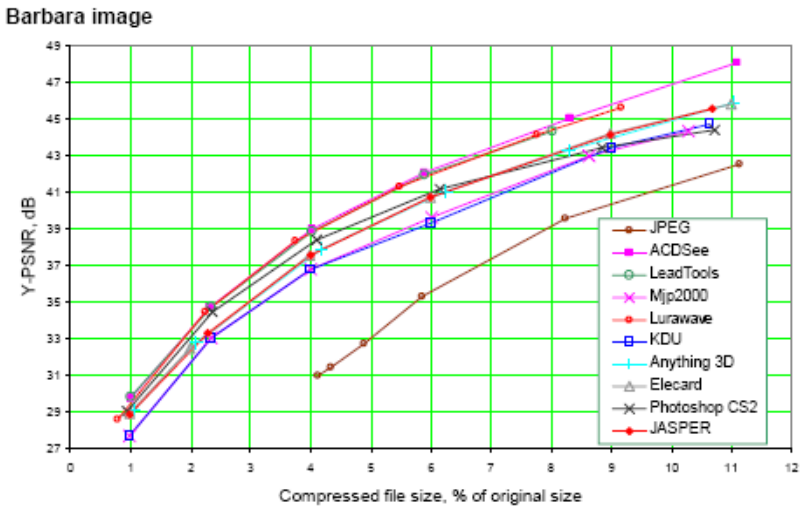


Figura 3.16: Confronto tra varie implementazioni di JPEG2000 e lo standard JPEG.

3.4 La codifica di immagini multicanale

Le immagini multicanale presentano un'elevata correlazione spettrale che vale la pena di esplodere. Una strategia diffusa è quella di effettuare una trasformata decorrelante tra i canali prima di codificare ogni singola componente con le tecniche classiche di codifica di immagini monocromatiche (*lossless* o *lossy*, a seconda).

Per le immagini naturali, la decorrelazione del colore è effettuata da parte degli standard in maniera piuttosto semplice. Vengono definite delle trasformazioni invertibili a coefficienti fissi tra lo spazio RGB ed uno spazio di luminanza e di crominanza. A volte, come in JPEG e negli standard di codifica video, le componenti di cominanza sono sottocampionate. Un tipo di campionamento assai diffuso è quello $2:1$ per righe e per colonne.

Per altri tipi di immagini multicanale, solitamente con un numero di canali maggiore di tre, gli standard non suggeriscono una strategia univoca di decorrelazione spettrale. JPEG2000, ad esempio, prevede anche la codifica di immagini multicanale, ammettendo eventualmente il preprocessing mediante una trasformata lineare tra le bande, tuttavia spetta al progettista definire esattamente i coefficienti della trasformata. E' ammissibile, per il caso lossy, anche scartare i canali meno significativi a valle della trasformazione. Questa strategia può essere utile nel caso i canali siano davvero tanti (ad esempio nel caso delle immagini iperspettrali [17]).

Nella dimensione trasversa ciascuna immagine, nelle soluzioni più recenti, è analizzata mediante trasformata wavelet. Le trasformata spettrale più efficace è invece la KLT [18]. Una DWT spettrale diadica tuttavia spesso è preferita [19], in quanto meno onerosa computazionalmente. La DWT è particolarmente efficace quando il segnale è continuo a tratti nella direzione spettrale, come ad esempio nel caso delle immagini iperspettrali.

Se le componenti sono molte l'impiego della quantizzazione vettoriale può portare a buoni risultati [20]. Lo schema di codifica è il seguente. Dapprima un quantizzatore vettoriale è esercitata sui vettori costituiti da *pixel* omologhi lungo la direzione spettrale. L'immagine quantizzata viene dunque sottratta a quella originale. I residui a questo punto possono essere codificati con perdita di informazione. A seconda della cardinalità del dizionario VQ i residui possono essere più o meno correlati. Se la correlazione è piuttosto alta essi godono ancora di proprietà simili a quelle delle immagini naturali, per cui possono essere efficientemente codificati con le tecniche classiche di codifica di immagini monocromatiche. Se il dizionario VQ è ampio e la correlazione tra i residui è bassa, si possono invece impiegare con risultati analoghi tecniche

di codifica per segnali monodimensionali. E' importante notare che gli indici VQ suddividono l'immagine in regioni abbastanza connesse, le quali rappresentano una sorta di mappa dell'immagine originale. La mappa è anche essa un immagine monocromatica, a pochi livelli, la quale può essere agevolmente codificata con tecniche *lossless*.

Negli ultimi anni sono state proposte tecniche più sofisticate che usano la mappa ottenuta dalla VQ (o da altre strategie di segmentazione) per classificare l'immagine in regioni omogenee. A ciascuna regione è associata una particolare trasformazione che tiene conto del suo comportamento spettrale specifico [21]. Eventualmente la successiva trasformata bidimensionale sulle immagini trasformate è analizzata con tecniche per oggetti di forma arbitraria [7]. Di questa famiglia di tecniche innovative ci occuperemo nel dettaglio nel capitolo 6, fornendo un contributo innovativo nel capitolo 7.

3.4.1 Allocazione delle risorse per la codifica multicanale

La bontà dell'allocazione delle risorse dipende, oltre che dalla tecnica di allocazione, anche dalla granularità delle unità elemntari di allocazione. Nel caso di immagini multicanale l'unità elementare può essere un blocco bidimensionale della singola componente spettrale, un blocco tridimensionale (unione di più blocchi monocromatici) oppure un'intera immagine della pila. Una strategia può essere anche quella di assegnare le risorse secondo un criterio gerarchico: prima ad unità di allocazione intermedie, poi, all'interno della singola unità di allocazione intermedia, alle varie unità elementari. Ad esempio si può prima assegnare le risorse tra le varie immagini trasformate e poi ripartire capillarmente il budget di bit di ciascuna componente spettrale ai vari blocchi spaziali.
Il risultato migliore dipende dalla giusta combinazione di strategia generale di allocazione (*a priori*, *a posteriori*, *in itinere*) e di suddivisione in unità di allocazione. In generale gli approcci gerarchici sono un buon compromesso tra complessità e prestazioni.

3.5 La codifica video

La compressione video è normalmente *lossy*. Essa si basa sui principi della ridondanza spaziale e di quella temporale.

Come esplodere la ridondanza spaziale è stato discusso già nella sezione sulla codifica di immagini monocromatiche. Alcune tecniche di compressione video, il cui scopo è una codifica near lossless ed una bassa latenza temporale, si occupano solo di eliminare la ridondanza spaziale codificando il video fotogramma per fotogramma. Lo standard JPEG e JPEG2000 ad esempio hanno una loro versione per la codifica video, Motion JPEG e JPEG2000, che viene impiegata proprio nelle applicazioni a bassa latenza ed alta qualità, come ad esempio la registrazione ad alta fedeltà.

Tuttavia, nella maggior parte delle applicazioni, la disponibilità limitata di risorse non consente di trascurare l'importante fattore della ridondanza temporale. Una tipica tecnica di codifica video inizia la codifica comprimendo il primo fotogramma mediante tecniche di codifica per immagini. Le frame successive vengono poi opportunamente codificate identificando i cambiamenti rispetto ad i fottogrammi precedenti. Solo questi cambiamenti vengono realmente codificati, con un enorme vantaggi sul tasso di codifica. Se però si rileva che un certo fotogramma è davvero molto distante da quelli precedenti, di certo non conviene codificarlo come differenza rispetto ad una frame di riferimento. Dunque anche esso va convenientemente codificato come una semplice immagine. Nella letteratura tecnica un fotogramma codificato indipendentemente è detto **intra-frame**, mentre un fotogramma codificato usando i suoi predecessori è detto **inter-frame**. Nelle soluzioni più recenti un fotogramma può essere codificato anche a partire da frame sia precedenti che successive. In questo caso si parla di **predizione bidirezionale**. Con riferimento agli standard di codifica video, d'ora in avanti indicheremo con la lettera **I** una frame di tipo intra, con la lettera **P** una di tipo inter che ha come riferimento frame precedenti, con la lettera **B** un fotogramma predetto bidirezionalmente.
I sistemi che impiegano due strategie di codifica differenti per le *intra-frame* e le *inter-frame*, sono detti **codificatori ibridi** [23].

A differenza che per la codifica di immagini, nel caso della codifica video è fondamentale il parametro tempo. Infatti quasi sempre la sequenza video deve poter essere codificata o riprodotta ad un certo frame-rate e sotto il vincolo di una certa latenza massima. Si mira quindi all'ottimizzazione di tre parametri: tasso, distorsione e tempo di calcolo.

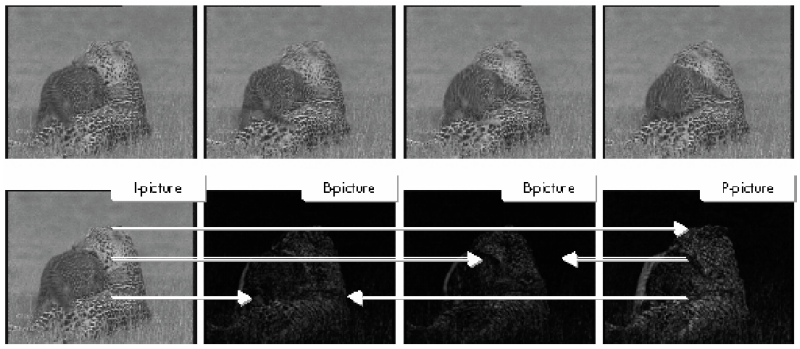


Figura 3.17: Fotogrammi di tipo I, P e B.

3.5.1 Stima e compensazione del movimento

Il modo più semplice di predire una frame da un'altra sarebbe semplicemente quello di riporre la frame di riferimento. Questa strategia, per quanto semplice, è però inefficace in quanto è quasi sempre presente un certo moto relativo tra il sensore di acquisizione e parte della scena. Per questo motivo vale la pena di introdurre opportune strategie di stima e compensazione del movimento. La **stima del movimento** è un'elaborazione in grado di cercare in un immagine di riferimento la posizione originaria di un oggetto presente nella rappresentazione attuale. La **compensazione del movimento** deforma l'immagine di riferimento in maniera che da essa si possa ricavare la predizione ottimale (a posteriori) dell'immagine corrente da codificare. Di una *inter-frame* dunque se ne codifica la differenza rispetto alla versione predetta mediante compensazione del movimento. Chiaramente dovranno essere codificati anche tutti quei parametri necessari a specificare la deformazione dei fotogrammi di riferimento.

3.5.2 Campi di moto

Esistono diversi modelli per descrivere, in una sequenza video, la deformazione di un fotogramma allo scorrere del tempo. Ad esempio un modello possibile è quello delle *maglie triangolari*, **triangular mesh**, in cui l'immagine è partizionata in triangoli ed il movimento è visto come una deformazione della maglia da essi costituita (vedi fig.).

Il modello più diffuso, a causa della sua estrema semplicità di calcolo e di

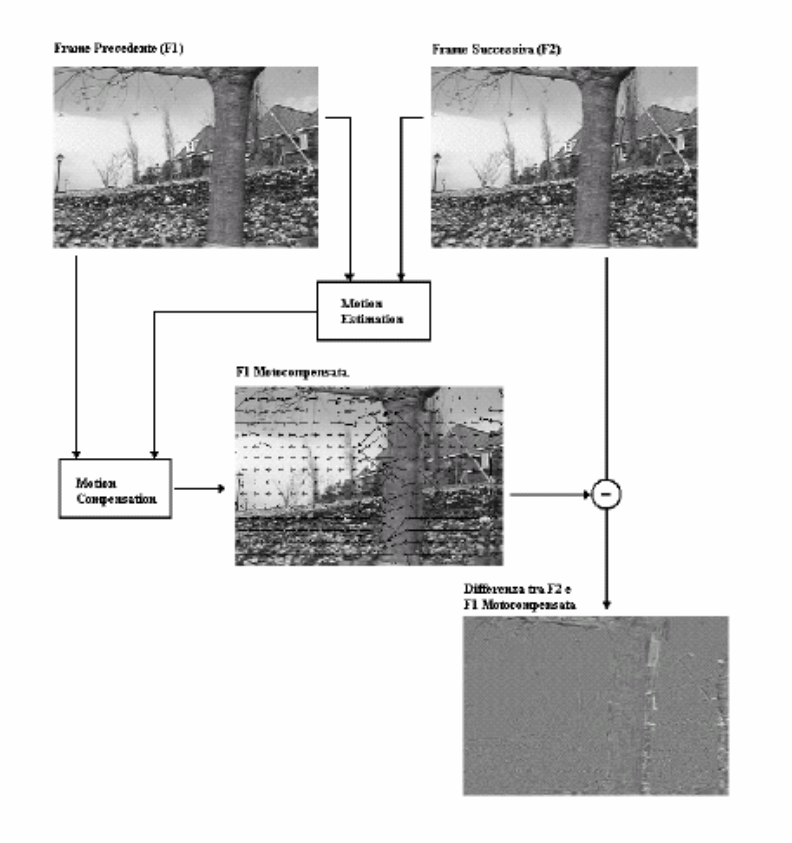


Figura 3.18: Stima e compensazione del movimento.

codifica, e quello di moto traslazionale. Il fotogramma corrente è suddiviso in blocchi. Ogni blocco è associato ad un blocco analogo nel fotogramma di riferimento traslato opportunamente di un certo vettore, detto **vettore di movimento**. L'insieme di tutti i vettori di movimento è detto **campo di moto**.
Sebbene il moto traslazionale e per blocchi sia una mera semplificazione delle reali deformazioni tra fotogrammi, comunque si riesce ad abbattere di molto l'energia dei residui, intesi come differenza tra il fotogramma da codificare e la predizione. Ricavare i vettori di movimento è piuttosto semplice e inoltre codificare il *campo di moto* può essere molto meno oneroso che codificare i parametri di un modello più complesso.
Caratteristiche fondamentali di un campo di moto sono la dimensione dei blocchi ed i possibili valori ammessi per le componenti dei vettori di movimento. Di solito essi possono estendersi all'interno di una finestra di ricerca rettangolare. Le componenti di un vettore possono essere intere o frazionarie. Nell'ultimo caso, durante la fase di ricerca, il confronto tra blocchi è effettuato con tecniche di interpolazione. Tutti questi parametri solitamente sono noti sia al codificatore che al decodificatore.
Il codificatore invece è pienamente libero di adottare la strategia di ricerca più opportuna. Sicuramente la strategia di ricerca scelta influenza le prestazioni, come anche la sua complessità pesa sul frame-rate e sui tempi di latenza.

3.5.3 Anello di retroazione

Un problema fondamentale in fase di decodifica è il seguente. Le frame di riferimento che vengono utilizzate dal decodificatore per ricostruire la predizione non sono quelle originali, ma sono ricostruite a valle di un processo di quantizzazione. Questo genera sicuramente delle inefficienze. Per questo motivo può convenire effettuare la predizione non dai fotogrammi originali ma da quelli quantizzati. Per questo motivo un decodificatore è inserito in retroazione nel codificatore stesso in modo da poter effettuare la compensazione del movimento sui fotogrammi precedentemente codificati. Si parla in questo caso di *ciclo chiuso* e di *anello di retroazione* al codificatore.

3.5.4 Gli standard per la codifica video

Lo schema descritto nel paragrafo precedente è proprio quello della famiglia degli standard di codifica multimediale MPEG, nonchè quello della famiglia degli standard di codifica video ITU H.26X. Questi standard, la cui versione più recente è H.264/AVC [24] [25], impiegano tutti uno schema di *codifica*

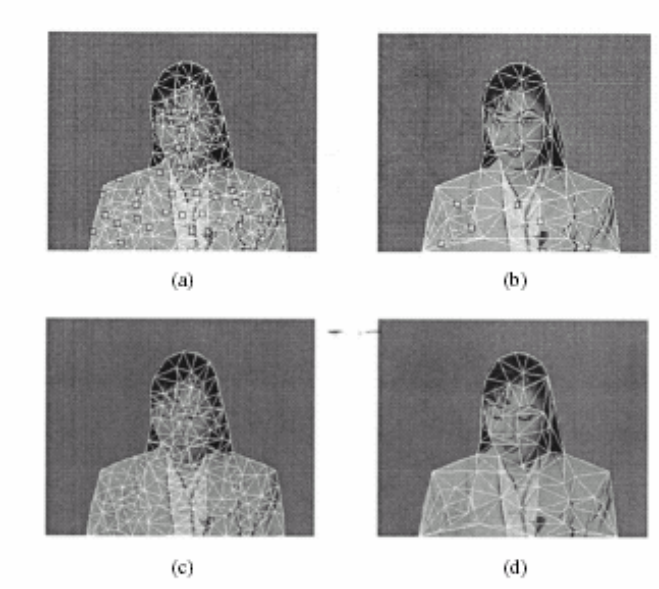


Figura 3.19: Stima del movimento mediante *triangular mesh.*

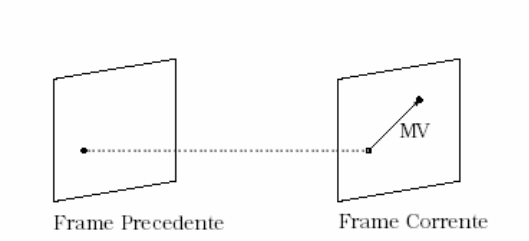


Figura 3.20: Determinazione del vettore di moto che descrive lo spostamento di un oggetto.

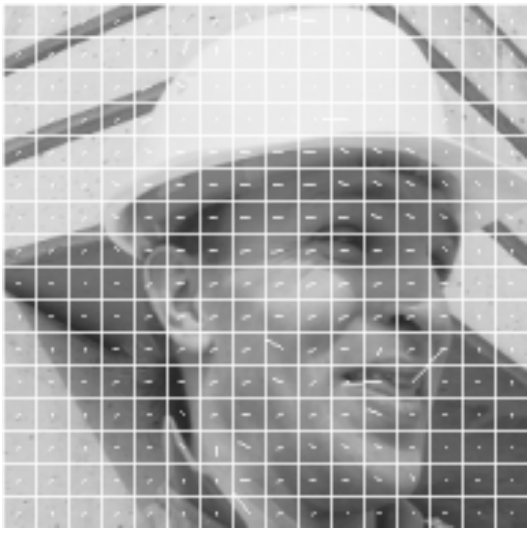


Figura 3.21: Stima del movimento basata su suddivisione in blocchi e modello di moto traslazionale.

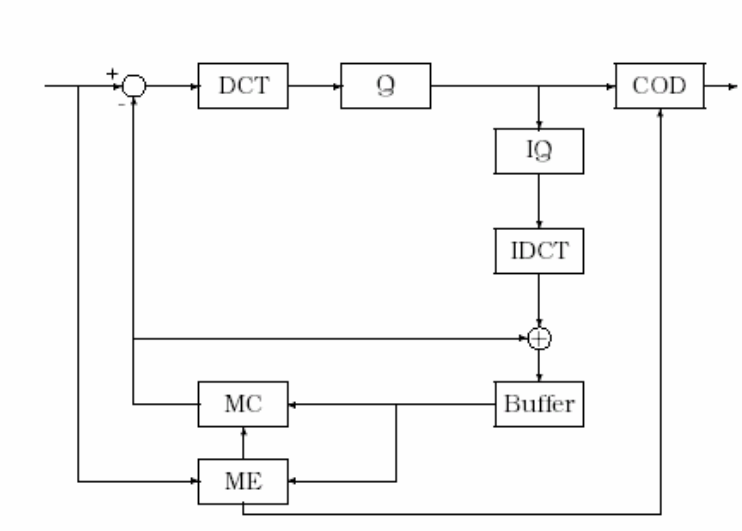


Figura 3.22: Codificatore ibrido a ciclo chiuso.

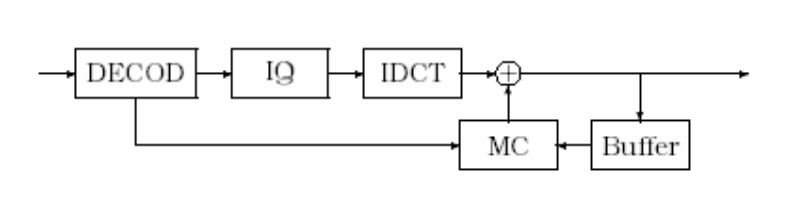


Figura 3.23: Decodificatore ibrido.

ibrida con *anello di retroazione*. Sia la codifica *intra-frame* che quella *inter-frame* effettuano trasformate a blocco. I coefficienti vengono quantizzati in maniera analoga a come avviene in JPEG. Complessi algoritmi di codifica entropica si occupano poi di compattare gli indici di quantizzazione ed i vettori di movimento.

Le novità dell'ultimo standard, H.264, rispetto a questo schema generale sono:

- un modello gerarchico per il campo di moto (vedi fig.);
- la possibilità di predirre un'*inter-frame* da un numero arbitrario di fotogrammi di riferimenti;
- maggior precisione dei vettori di movimento;
- una maggiore attenzione alla codifica entropica.

Le prestazioni tasso-distorsione, grazie a questi diversi fattori, sono molto migliorate rispetto alla parte 2 dello standard MPEG4, così come mostrato in fig. 3.26.

Inoltre lo standard di codifica H.264 si integra nel precedente MPEG4 [25], il quale oltre ad offrire strumenti di codifica più sofisticati rispetto alle versioni precedenti, aveva introdotto numerose funzionalità quali:

- scalabilità in risoluzione;
- scalabilità in qualità;
- accesso diretto temporale migliorato;
- editing diretto sul flusso di codifica;
- codifica combinata di contenuti naturali e sintetici;

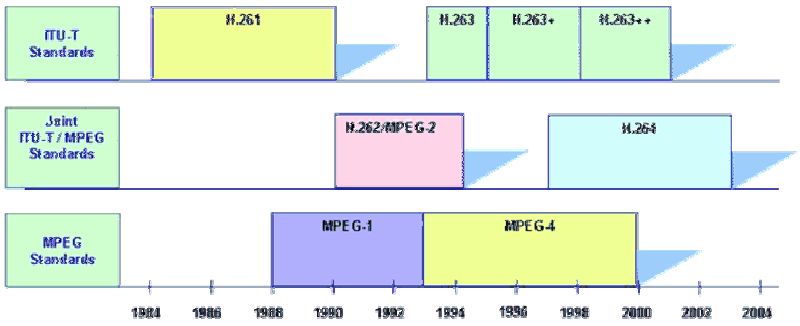


Figura 3.24: Evoluzione degli standard di codifica video nel tempo.

- codifica di flussi video concorrenti;
- robustezza rispetto agli errori.

Non è lo scopo di queste note illustrare tutte le funzionalità e gli strumenti previsti dai moderni standard di codifica, che tra l'altro come nel caso di MPEG sono standard di codifica multimediale (quindi oltre che di video, di audio e di altri contenuti ausiliari). Resta importante capire che le prestazioni tasso-distorsione sono solo uno dei numerosi aspetti da tenere in conto per una codifica ottimale e funzionale dei dati.

3.5.5 Codificatori wavelet con filtraggio temporale

Implementare la scalabilità spaziale e temporale non è molto semplice per gli schemi di codifica ibrida. Inoltre la codifica di un flusso video scalabile non è altrettanto efficiente di quella di un flusso video non scalabile. Un'alternativa alla codifica ibrida, che permette una scalabilità completa a basso costo di codifica, è la codifica basata su trasformata wavelet tridimensionale. In questo schema la DWT viene effettuata sia lungo la dimesione temporale sia bidimensionalmente per ciascuna frame. Il filtraggio temporale viene esegito su tutti i *pixel* in maniera tale che, a valle della successiva DWT-2D sui fotogrammi trasformati, la decomposizione finale non è diadica ma *packet*. Sperimentalmente si dimostra che tale decomposizione è la più adatta a decorrelare il segnale video.
In realtà a questo semplice schema, che già di per se è in grado di fornire le stesse prestazioni tasso-distorsione di un codificatore MPEG2, può essere eventualmente migliorato inserendo anche la stima e la compensazione del movimento [27] [28] (fig. 3.27).

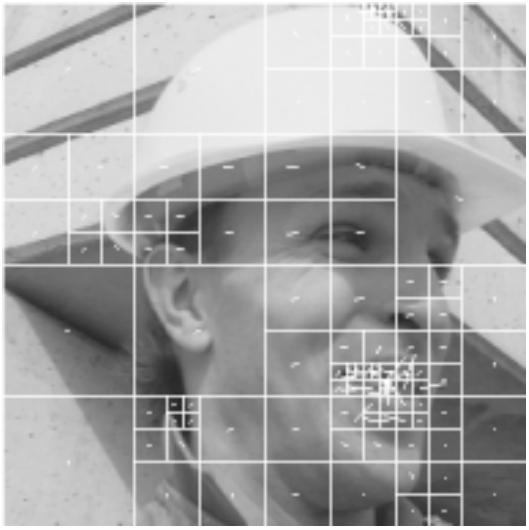


Figura 3.25: Modello gerarchico per il campo di moto.

Uno schema di particolare semplicità ed efficacia è quello proposto in [14] che propone nella dimensione temporale un lifting scheme non lineare, in cui al filtraggio di predizione e di aggiornamento delle frame contigue si sostituisce il filtraggio delle frame moto-compensate.
Detti $P(.)$ (*predict*) ed $U(.)$ (*upgrade*) i filtri di predizione e di aggiornamento, $\{x_k\}$ le frame della sequenza originale, $\{l_k\}$ le frame di trend ed $\{h_k\}$ le frame di anomalia ad un livello di decomposizione, il generico passo di lifting senza compensazione del movimento è il seguente:

$$h_k = x_{2k+1} - P(\{x_{2n}\}_n) \tag{3.4}$$

$$l_k = x_{2k} + U(\{h_n\}_n) \tag{3.5}$$

Il passo di predizione $P(.)$ e quello di aggiornamento $U(.)$, sono generalmente filtraggi su diversi coefficienti della sequenza di frame pari $\{x_{2n}\}$ e della sequenza di fotogrammi di anomalia $\{h_n\}$. Per semplicità abbiamo esposto il caso in cui il lifting scheme comporti un solo passo di predizione e di aggiornamento ed un solo livello di decomposizione, ma la procedura è facilmente estendibile a schemi con diversi passi di *predict* ed *upgrade* e a diversi livelli di decomposizione. Attraverso il lifting scheme si può implementare una trasformata wavelet temporale senza compensazione del movimento.
Supponiamo invece di aver calcolato il campo di moto tra la frame x_n ed una

foreman

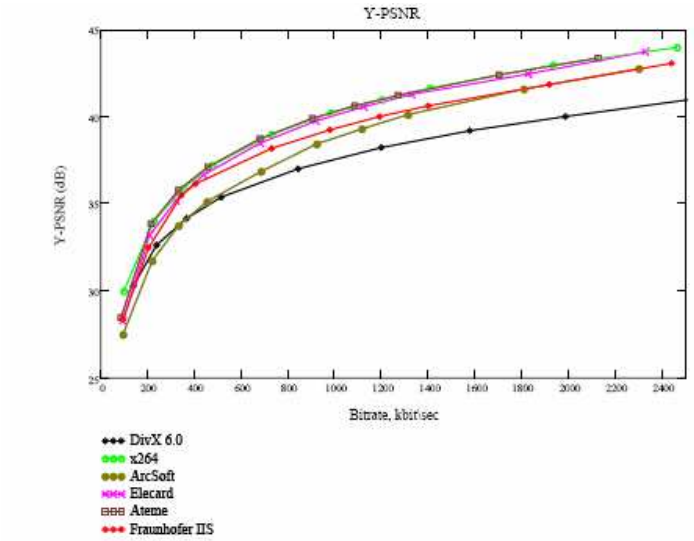


Figura 3.26: Prestazioni di vari codificatori della famiglia H.264 rispetto ad uno dei migliori codificatori MPEG4 parte 2: il codificatore DivX.

frame x_m, dove $m > n$. Il campo di moto che allinea x_n ad x_m è detto **campo diretto** (*forward*) quello che viceversa allinea x_m ad x_n è detto **campo inverso** (*backward*). I passi di predict e di upgrade questa volta possono essere effettuati sulle versioni moto-compensate delle frame elaborate.

$$h_k = x_{2k+1} - P(\{MC(x_{2n})\}_n) \quad (3.6)$$
$$l_k = x_{2k} + U(\{MC^{-1}(h_n)\}_n) \quad (3.7)$$

L'operazione di compensazione del movimento è stata indicata con $MC(.)$. Per la stima viene utilizzato il campo diretto o inverso a seconda che la frame preceda o succeda temporalmente quella da codificare. L'operazione $MC^{-1}(.)$ indica invece una compensazione del movimento di segno opposto rispetto a quella effettuata in precedenza. Campi inversi vengono utilizzati lì dove prima erano impiegati campi diretti e viceversa. Lo schema è esposto in fig. 3.28 ed approfondito in [31].
Per essere più chiari riportiamo qui le semplici equazioni del lifting che implementa la wavelet *biortogonale* 5-3. Senza compensazione del movimento il lifting scheme è il seguente:

$$h_k = x_{2k+1} - \frac{1}{2}(x_{2k} + x_{2k+2}) \quad (3.8)$$
$$l_k = x_{2k} + \frac{1}{4}(h_k + h_{k+1}) \quad (3.9)$$

Con la compensazione invece lo schema si può scrivere così:

$$h_k = x_{2k+1} - \frac{1}{2}(F_{2k\to 2k+1}(x_{2k}) + B_{2k+2\to 2k+1}(x_{2k+2})) \quad (3.10)$$
$$l_k = x_{2k} + \frac{1}{4}(B_{2k\to 2k+1}(h_k) + F_{2k+2\to 2k+1}(h_{k+1})) \quad (3.11)$$

Per chiarezza sono state omesse le costanti di normalizzazione dell'energia. Solitamente con buona approssimazione si possono ottenere i campi inversi a partire da quelli diretti, per cui è necessario codificare solo questi ultimi. Si noti che lo schema descritto non è a ciclo chiuso. L'utilizzo di JPEG2000 in schemi di codifica wavelet con lifting schemes compensati in movimento è approfondito in [15].
La codifica wavelet con compensazione del movimento presenta buoni risultati quando il campo di moto è accurato e rappresenta bene il movimento all'interno della scena. Infatti è importante che a valle della wavelet temporale i fotogrammi trasformati preservino la loro natura di immagini naturali,

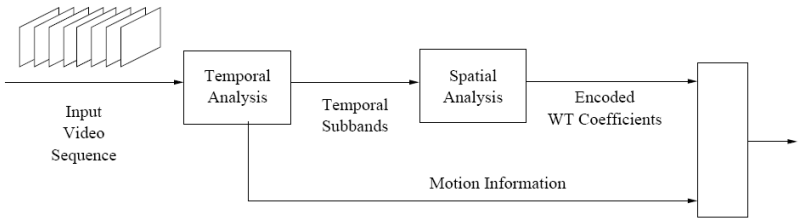


Figura 3.27: Codificatore basato su wavelet tridimensionale.

sulla quale la DWT diadica è lo strumento più adatto. Qualora ciò non accada non sempre i risultati sono soddisfacenti. Inoltre la scalabilità spaziale in uno schema così semplice risulta compromessa dalla compensazione del movimento e per preservarla c'è bisogno di passare a schemi più articolati [31].

3.5.6 Allocazione delle risorse per la codifica video

Nel caso della codifica video l'obbiettivo dell'allocazione delle risorse può essere differente. Nel caso di un video che può essere codificato *off-line* la strategia migliore è quella di avere una **qualità costante** per tutte le frame. Per un video codificato in presa diretta invece l'obiettivo da raggiungere è sfruttare al massimo la capacità di canale, effettuando appunto una codifica a **bit rate costante**. Anche quando la codifica avviene *off-line* spesso risulta eccessivo predisporre unità elementari di allocazione delle risorse troppo piccole. Solitamente l'unità elementare di allocazione è proprio l'intera frame. Nelle implementazioni reali non ci dobbiamo aspettare dunque, come invece avviene nella codifica di immagini fisse, risultati particolarmente ottimizzati rispetto al contenuto spaziale del singolo fotogramma. I fenomeni al centro dell'attenzione del codificatore sono quelli temporali, per cui la ridondanza spaziale è spesso considerata di secondaria importanza.

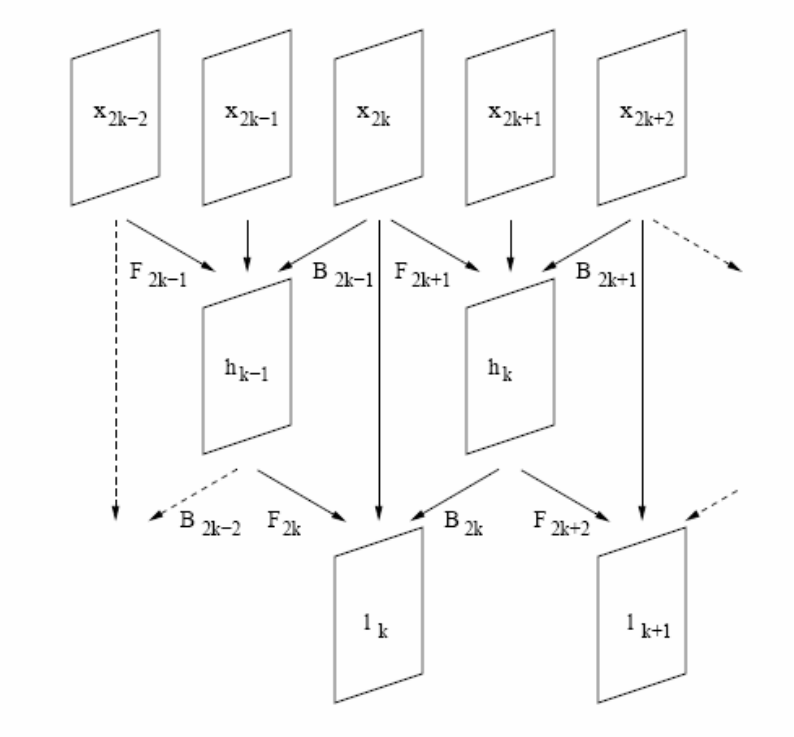


Figura 3.28: Lifting scheme con compensazione del movimento.

Bibliografia

[1] D. Salomon “Data compression - the complete reference”, *Springer*, IV Ed., 2007.

[2] J. A. Richards, X. Jia , “Remote sensing digital image analysis, an introduction”, IV Ed. , Springer 2005.

[3] S. Winkler, “Digital Video Quality: Vision Models and Metrics”, Wiley, 2005.

[4] ITU-R BT.500-11, “Methodology for the subjective assessment of quality of television pictures”.

[5] O. P., Dmitry Vatolin - “MSU Subjective Comparison Of Modern Video Codecs”, *MSU Graphics and Media Lab*.

[6] R. M. Rao, A. S. Bopardikar “Wavelet transform: introduction to theory and applications”, Addison-Wesley, 1998.

[7] S. Mallat, “A wavelet tour into signal processing”, Elsevier, 1998.

[8] W. Swedelns, “Building your own wavelet at home”, *Wavelets in Computer Graphics*, ACM SIGGRAPH course notes, 1996.

[9] J.M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients”, *IEEE Trans. On Signal Processing*, vol. 41, n.12 pp. 3445-3463, dicembre 1993.

[10] A. Said, W. A. Pearlman, “New fast and efficient image codec based on set partioning in hierarchical tree”, *IEEE Trans. On Circuits and Systems on Video Technology*, vol. 6, pp. 243-250, giugno 1996.

[11] N. Sprljan, S. Grcjc, M. Grcjc, “Modified SPIHT for packet wavelet image coding”, *Real Time Imaging*, Elsevier, 2005.

[12] S. Li, W. Li, “Shape adaptive dicrete wavelet transforms for arbitrarely shaped visual object coding”. IEEE Trans. on Circuits and System for Video Technology, pp. 725-743, agosto 2000.

[13] D. S. Taubman, “High performance scalable image compression with EBCOT”, *IEEE Trans. On Image Processing*, vol. 9, n. 7, pp. 1158 - 1170, luglio 2000.

[14] D. S. Taubman, M. W. Marcellin, “JPEG2000, image compression fundamentals, standards and pratice”, Norwell, MA, Kluwer Academic, 2002.

[15] J.A.Saghri, A.G.Tescher, J.T.Reagan, “Practical transform coding of multispectral imagery”, *IEEE Signal Processing Magazine*, pp.32-43, Jan. 1995.

[16] J. T. Rucker, J. E. Fowler, N. H. Younan, “JPEG2000 coding strategies for hyperspectral data”, in International Geoscience and Remote Sensing Symposium, vol. 1, pp. 128-131, Seul, Corea del Sud, luglio 2005.

[17] Q. Du, J. E. Fowler, “Hyperspectral image compression using JPEG2000 and principal component analysis”, IEEE Geoscience and Remote Sensing Letters, vol. 4, pp. 201-2005, aprile 2007.

[18] P.L.Dragotti, G.Poggi, A.R.P.Ragozini, “Compression of multispectral images by three-dimensional SPIHT algorithm”, *IEEE Transactions on Geoscience and Remote Sensing*, pp.416-428, Jan. 2000.

[19] X.Tang, W.A.Pearlman, J.W.Modestino, “Hyperspectral Image Compression Using Three-Dimensional Wavelet Coding”, capitolo in *Hyperspectral Data Compression*, Kluwer Academic Publishers, 2005.

[20] S.E.Qian, “Hyperspectral data compression using a fast vector quantization algorithm”, *IEEE Transactions on Geoscience and Remote Sensing*, pp.1791-1798, Aug. 2004.

[21] G.Gelli, G.Poggi, “Compression of multispectral images by spectral classification and transform coding”, *IEEE Transactions on Image Processing*, pp.476-489, Apr. 1999.

[22] M. Cagnazzo, R. Gaetano, S. Parrilli, and L. Verdoliva. “Region based compression of multispectral images by classified klt”, *European Signal Processing Conference* (EUSIPCO), Firenze, settembre 2006.

[23] I. E. G. Richardson, “Video Codec Desig”, John Wiley and Sons, 2002.

[24] T. Wiegand, G. J. Sullivan, G. Biontegaard, A. Luthra, “Overview of the H.264/AVC video coding standard”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, luglio 2003.

[25] I. E. G. Richardson - “H.264 and MPEG-4 video compression”, John Wiley and Sons, 2003.

[26] D. Kulikov, A. Parshin, D. Vatolin - “MPEG4 Video Codecs Comparison”, *MSU Graphics and Media Lab*.

[27] J. R. Ohm, “Three-dimensional subband coding with motion compensation”, *IEEE Trans. on Image Processing*, vol. 3, n. 5, pp. 559 - 571, settembre 1994.

[28] S. J. Choi, J. W. Woods, “Motion compensated 3D subband coding of video”, *IEEE Trans. on Image Processing*, vol. 8, n. 2, pp. 155 - 167, febbraio 1999.

[29] A. Secker and D. Taubman “Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression”, *IEEE Transactions on Image Processing*, vol. 12, n. 12, pp. 1530-1542, dicembre 2003.

[30] M. Cagnazzo, “Wavelet transform and three-dimensional data compression”, tesi di dottorato, Univ. Federico II di Napoli ed Univ. di Nizza-Sophia Antipolis (Francia), marzo 2005.

[31] D. Taubman, E.Change, A. Zakhor “Directionality and scalability in subband image and video compression”, *Image Technology: Advances in Image Processing, Multimedia and Machine Vision* , Springer, 1996.

Parte II

I quattro capitoli che compongono questa parte costituiscono una relazione scientifica completa sulle attività di ricerca effettuate durante il corso di dottorato. La ricerca svolta si integra nei filoni di attività dell'unità di ricerca sull'elaborazione di immagini dell'*Università degli Studi "Federico II" di Napoli* coordinata dal prof. Giovanni Poggi. Particolare attenzione è rivolta alle applicazione del settore aerospaziale, in cui opera il *Centro Italiano Ricerche Aerospaziali*, ente che ha parzialmente finanziato il dottorato. L'unità di ricerca di Napoli da diversi anni si occupa di codifica di segnali visuali e tra gli argomenti di maggiore interesse vi sono la *codifica mediante analisi multirisoluzione* e la *codifica mediante classificazione*. L'attività di studio del dottorato è stata incentrata principalmente su questi due paradigmi.
Per quel che riguarda le tecniche di **codifica multirisoluzione**, l'attenzione del mondo scientifico e tecnologico, negli ultimi anni, è rimasta fissa sulla *trasformata wavelet* e, di recente, si è spostata su trasformate più complesse dalla *wavelet* direttamente derivate. Per queste trasformate sono stati progettati diversi algoritmi di codifica. Sicuramente i più celebri, anche se non ancora adottati dagli standard, sono gli *zerotree coder*. Pur essendo estremamente famosi, al punto di diventare il principale *benchmark* per la codifica multirisoluzione, al giorno d'oggi degli *zerotree coder*, non esiste ancora una formalizzazione completa. Nell'ambito di questo lavoro di tesi si propone, per la prima volta, un paradigma generale in grado di descrivere le più famose istanze storiche di questa famiglia di codificatori, lasciando anche spazio al progetto di nuove soluzioni. Nel quarto capitolo, infatti, si illustra il modello generale di *zerotree coder* e nel quinto capitolo si effettua un'accurata analisi sperimentale per trarre alcune linee guida per il progetto dei codificatori.
La **codifica mediante classificazione** è un argomento estremamente interessante dal punto di vista scientifico. La classificazione dei pixel di un'immagine è interpretabile come una segmentazione; in questo modo, il contenuto di un'immagine multicanale può essere organizzato in una mappa bidimensionale

di regioni e in una matrice tridimensionale di texture. E' come se si separasse il "significato" dei dati dall'informazione che ne completa la rappresentazione, per avere, in questo modo, una forma di elaborazione di più alto livello incapsulata all'interno del flusso di codifica, funzionalità estremamente avanzata che può trovare non trascurabili riscontri applicativi nel prossimo futuro. Un'altra importante possibilità è quella di poter ricorrere a *trasformate classificate*, cioè adattate alle statistiche della particolare regione, per cui immancabilmente più efficienti. La *trasformata classificata* è uno strumento di codifica nuovo, non ancora ben formalizzato, ma con interessanti potenzialità teoriche ed applicative. Perciò, nel presente lavoro, è stato studiato il più semplice degli schemi proposto dal gruppo di ricerca, con il fine di ingegnerizzarlo e di dimostrare sia la sua fattibilità sia la sua bassa complessità. Il sesto capitolo, invece, è stato dedicato alla presentazione degli ultimi sviluppi della ricerca sulla codifica mediante classificazione; mentre, nel settimo capitolo, è stato approfondito l'adeguamento dello schema originale alla codifica *on-board* di immagini multispettrali telerilevate.

Capitolo 4

Il paradigma degli zerotree coder

Parole chiave

Nozioni propedeutiche

Codifica lossless, rappresentazione in segno e modulo, quantizzazione scalare, codifica mediante trasformata, trasformata wavelet, quantizzatore a reticolo, quantizzatore uniforme, zona morta, classificazione, clustering, sistema a stati finiti, diagrammi di transizione degli stati, codifica aritmetica adattativa.

Concetti introdotti nel capitolo

Partizionamento gerarchico, bit di significatività, bit di rappresentazione, bit di segno, bit di raffinamento, codifica della significatività, codifica per bit-plane, codifica embedded, zerotree, ordine di zerotree, EZW, SPIHT, SPECK, PROGRES, implementazione mediante liste, implementazione mediante liste multiple.

Concetti innovativi illustrati nel capitolo

HPM (hierarchical partitioning model), codificatore gerarchico, zeroset, zeroset coder, zerotree coder, azioni elementari di codifica per uno zerotree coder, diagrammi di evoluzione degli alberi, tabelle di evoluzione degli alberi.

4.1 Introduzione

Nell'ambito della *codifica mediante trasformata*, esistono diverse tecniche per la quantizzazione e codifica dei coefficienti. Alcune tecniche di grande successo applicativo separano l'*informazione di significatività* dei coefficienti dal resto, individuando in essa la principale ridondanza informazionale.
Gli *zeroset coder* in particolare mirano ad una codifica efficiente dell'*informazione di significatività* quando essa è sparsa tra gli elementi da codificare, per cui sono particolarmente adatti alla *trasformata wavelet*. Il principio ispiratore è l'organizzazione dell'*informazione di significatività* in insiemi ricorsivamente scomponibili secondo uno schema gerarchico. La gerarchia è, come vedremo, rappresentabile attraverso un *sistema di alberi di partizionamento*, struttura che rende estremamente semplice il progetto e l'implementazione di questo tipo di codificatori. Gli *zeroset coder* costituiscono una famiglia piuttosto ampia. Tra di essi gli *zerotree coder* si caratterizzano per la definizione della gerarchia a partire da una particolare struttura, anche essa ad albero, detta *albero descrittore della trasformata*. L'*albero descrittore* serve a sintetizzare direttamente il *sistema di alberi di partizionamento*, in maniera iterativa e ricorsiva. Altre strutture, diverse dall'*albero descrittore*, permettono di generare in maniera iterativa e/o ricorsiva la gerarchia di alcuni altri *zeroset coder*.

Le più apprezzate proprietà degli *zeroset coder* sono la semplicità di calcolo e di progetto, la possibilità di implementare diversi tipi di scalabilità, le prestazioni vicine a quelle di metodi più complessi (almeno dal punto di vista progettuale). Le applicazioni riguardano soprattutto la codifica con o senza perdita di *informazione* di immagini e di sequenze video, solitamente a valle di una *trasformata wavelet*. Il loro principale limite è l'ipotesi piuttosto forte, anche se genericamente verificata, sull'andamento energetico dei coefficienti trasformati (bassa robustezza, solo immagini naturali). L'interesse nei loro riguardi è vivo in quanto da un lato essi costituiscono un importante riferimento di complessità e prestazioni, dall'altro i principi su cui essi si fondano risultano assai efficaci nonostante la loro natura piuttosto semplice ed intuitiva.

L'approfondimento dello studio di questa famiglia di tecniche e l'elaborazione di una teoria più rigorosa puo' portare a nuove soluzioni eventualmente migliori sotto il profilo della robustezza, delle prestazioni, della complessità, delle funzionalità. Sicuramente la sottofamiglia degli *zerotree coder* rappresenta un caso particolare di estremo interesse in quanto

ampiamente impiegata nelle applicazioni e studiata in modo approfondito in letteratura. Per questi motivi, ad essa verrà rivolta una particolare attenzione nei seguenti paragrafi, con la speranza però che gli studi rivolti agli *zerotree coder* aiutino a capire meglio la più ampia famiglia degli *zeroset coder*.

4.1.1 Obiettivi del capitolo

Lo scopo del lavoro presentato in questo capitolo è l'unificazione della vasta famiglia degli *zeroset coder* ed in particolare degli *zerotree coder*. Si mira cioè alla proposizione di un unico schema di riferimento dotato di proprietà generali la cui particolarizzazione di volta in volta, da un lato permetta di ricavare importanti algoritmi proposti in letteratura, per un più agevole e limpido confronto, dall'altro possa semplificare ed ispirare il progetto di nuove soluzioni, avendo individuato con maggiore chiarezza l'influenza dei singoli aspetti sulle funzionalità e sulle prestazioni.

4.1.2 Organizzazione degli argomenti

Gli argomenti oggetto del capitolo saranno trattati secondo la seguente organizzazione per paragrafi:

4.2, storia breve degli zeroset coder e degli zerotree coder: una panoramica storica sullo sviluppo degli *zeroset* coder e degli zerotreee coder;

4.3, codificatori gerarchici: si definisce formalmente l'HPM (*hierarchical partitioning model*), il modello di partizionamento gerarchico di insiemi utilizzato dai *codificatori gerarchici*;

4.4, zeroset coder: si descrivono il funzionamento generale degli *zeroset coder* e le implementazioni mediante liste e liste multiple;

4.5, zerotree coder: con maggior dettaglio si introducono gli *zerotree coder* basati sull'impiego dell'*albero descrittore della trasformata*;

4.6, strumenti di porgetto per zerotree coder: si introduce il modello a stati finiti per gli *zerotree coder* e la possibilità di descrivere le transizioni attraverso l'iterazione e la ricorsione del modello dei sottostati;

4.7, *codifica aritmetica* adattativa si introduce un modello generale di *codifica aritmetica* adattativa per *zerotree coder*;

4.8, configurazioni storiche: con gli strumenti acquisiti si rivisitano gli algoritmi EZW, SPIHT, PROGRES e SPECK;

4.9, conclusioni: si fa il punto sullo sforzo di generalizzazione svolto, sui suoi limiti e sulle sue potenzialità.

4.2 Cenni storici

Prima di introdurre il modello da noi proposto è opportuna una panoramica storica sugli *zeroset coder*, di cui gli *zerotree coder* costituiscono la sotto-classe più autorevole. Concetti, che in seguito saranno trattati in maniera più rigorosa, saranno invece nel corrente paragrafo solo brevemente introdotti. Non scoraggi il lettore il fatto che delle tecniche citate saranno messi in luce aspetti non sempre considerati fin'ora fondamentali o caratterizzanti, in quanto la piena comprensione delle argomentazioni riportate potrà avvenire agevolmente solo dopo la presentazione sistematica del modello.

4.2.1 Codifica della significatività attraverso partizionamento gerarchico ed albero descrittore della trasformata: EZW

La storia degli algoritmi degli *zeroset coder* inizia probabilmente con EZW di Shapiro [4]. EZW (the *Embedded Zerotree Wavelet* coder) si propone come un algoritmo di quantizzazione e codifica di coefficienti trasformati mediante wavelet. I coefficienti trasformati sono rappresentati idealmente in forma binaria, a virgola fissa, in segno e modulo. Le informazioni da codificare vengono suddivise in due categorie. Un primo tipo di *informazione* è la posizione del primo bit 1 del modulo di ciascun coefficiente. Un secondo tipo di *informazione* è invece costituito dal bit segno e dai bit necessari a rappresentare il modulo del coefficiente stesso con precisione via via crescente, cioè quelli successivi al primo 1. L'*informazione* del primo tipo, che d'ora in avanti definiremo di **significatività**, è vista come fortemente interdipendente tra coefficienti distinti. L'*informazione* del secondo tipo, che definiamo di **rappresentazione**, è assunta invece indipendente. L'interdipendenza dell'*informazione di significatività* per i coefficienti trasformati chiaramente è una proprietà dei dati, per cui è corretto affermare che in generale gli *zeroset coder* servono a codificare quelle classi di dati la cui *informazione di significatività* presenta un'interdipendenza statistica non trascurabile. EZW nello specifico si propone di rappresentare in maniera efficiente l'*informazione di significatività* di immagini naturali decomposte mediante *trasformata wavelet* diadica, che, come ben noto, presentano una forte localizzazione spaziale dei coefficienti significativi nelle sottobande di alta frequenza. Ipotesi statistiche sulla distribuzione in ciascun *bitplane* dell'*informazione di significatività*, basate sulla natura dei dati e sulla struttura della trasformata, portano alla

definizione dell'algoritmo vero e proprio.

Le considerazioni fin qui riportate costituiscono informalmente i principi fondamentali su cui si basa una classe di algoritmi di codifica ben più vasta di quella basata su modello gerarchico, che definiremo algoritmi basati su *codifica della significatività*. Di questa classe di algoritmi fa parte lo stesso EBCOT di Taubman [8], che costituisce di fatto la tecnica di quantizzazione e codifica dello standard per immagini fisse JPEG2000. Per tutti gli algoritmi basati su *codifica della significatività*, la codifica puo' avvenire progressivamente per *bitplane* in modo da ottenere la scalabilità in qualità (*codifica embedded*). Procedendo per *bitplane* è possibile ottenere una codifica con perdita di *informazione* codificando soltanto i primi *bitplane* dell'intero insieme di coefficienti, in modo da ottenere il bit rate desiderato.

L'apporto più significativo di EZW sta nell'originale metodo di codifica della significatività per i diversi *bitplane*. L'*informazione di significatività* del *bitplane* n-simo è organizzata inizialmente secondo una partizione di insiemi imposta dal *bitplane* precedente $n-1$. A ciascuno degli insiemi della partizione è associata un'aspettativa della *significatività* di tutti i suoi elementi in quel *bitplane*. Qualora la predizione risulti non corretta, ogni insieme puo' essere partizionato a sua volta in sottoinsiemi la cui *significatività* stavolta è predetta da un modello ben preciso. Tutti gli insiemi quindi vengono partizionati ricorsivamente finchè ad ognuno non sia possibile associare la *significatività* corretta. A ciascuna operazione di partizionamento corrisponde un'opportuna sequenza di bit che costituisce la codifica della significatività per quel particolare *bitplane*. Le particolarità esperesse fino ad ora informalmente caratterizzano il comportamento tipico degli *zeroset coder*. In realtà l'idea di *partizionamento gerarchico* per quanto già presente in latenza in EZW viene messo in luce soltanto con un lavoro successivo di Said e Pearlman, i quali presentano un algoritmo più efficiente di EZW e che sarà destinato a diventare il punto di riferimento principale per gli studi del settore: SPIHT [5].

SPIHT, come EZW, effettua il partizionamento gerarchico in maniera iterativa e ricorsiva grazie all'impiego di una particolare struttura, che qui indicheremo come *albero descrittore della trasformata*. L'*albero descrittore* tuttavia non è in grado di descrivere tutti i possibili *zeroset coder*. Gli *zeroset coder* che impiegano un *albero descrittore* per descrivere il sistema partizionamento sono chiamati in letteratura *zerotree coder*. In questo capitolo introdurremo invece

per la prima volta una struttura diversa, che chiamiamo, *sistema di alberi di partizionamento*, ben distinta dall'*albero descrittore* ed in grado di sintetizzare il comportamento di tutti gli *zeroset coder*.

4.2.2 Esplicitazione del principio del partizionamento gerarchico: SPIHT

La sigla EZW significa letteralmente *Embedded Zerotree Wavelet*. Il termine *embedded* si riferisce alla natura scalabile in qualità del flusso di codifica e quindi è legato alla codifica progressiva per *bitplane*. Il termine *zerotree* esprime invece il fatto che la gerarchia degli insiemi è mappata dall'*albero descrittore* (*tree*) e che tutti gli insiemi di cardinalità superiore ad 1 previsiti dal modello sono costituiti da coefficienti in quel *bitplane* non significativi, il cui valore del bit di significatività è cioè pari a 0 (*zero*). La seconda caratteristica è propria di tutti gli *zeroset coder*.

SPIHT introduce alcune novità importanti rispetto ad EZW. La prima è che considera un numero maggiore di partizioni possibili per la descrizione dei *bitplane* di *significatività* e di conseguenza adotta anche una legge di partizionamento più complessa. La seconda è che introduce il formalismo delle liste, particolarmente chiaro per la progettazione e la comprensione dell'algoritmo, anche se non il più efficiente computazionalmente e sicuramente non l'unico possibile [5]. Le prestazioni sono significativamente migliori rispetto ad EZW e SPIHT si candida a far parte di un futuro standard per la codifica di immagini fisse basato su wavelet, sebbene alla fine perda la sfida nel confronto con EBCOT.
Il merito più grande di Said e Pearlman è probabilmente quello di aver esposto in maniera più chiara ed esplicita il principio della suddivisione gerarchica. SPIHT significa infatti *Set Partitioning In Hierarchical Trees*.

4.2.3 Il concetto di zeroset coder e SPECK

Nel frattempo in [4] viene introdotto SPECK (*Set Partitioned Embedded bloCK coder*). Esso è uno *zeroset coder* ma a differenza di EZW e SPIHT non sfrutta l'*albero descrittore*, per cui non è uno *zerotree coder*. Si presenta inoltre, come elemento fortemente innovativo, la possibilità molto interessante di avere diverse leggi di partizionamento a seconda dell'insieme considerato (vedi parag. 4.8.3).

4.2.4 Nuovi zerotree coder

Ben presto diventa ben chiaro che SPIHT, introdotto per immagini naturali fisse, puo' essere esteso anche alla codifica di immagini multispettrali e di sequenze video, adeguando la struttura ad albero al caso tridimensionale [6].
Dragotti, Poggi e Ragozini in [7] intuiscono la possibilità di estendere la tecnica ad altri tipi di trasformata e soprattutto delineano un criterio per la progettazione dell'albero, il quale viene strutturato secondo la trasformata stessa.
Danyali e Mertins in [10] e poi in [11] introducono liste multiple per gestire l'importante funzionalità di scalabilità in risoluzione. Nel recente [14] invece si adatta SPIHT alla *codifica lossless* evitando la computazionalmente costosa codifica per *bitplane*.

4.2.5 Funzionalità per gli zeroset coder

Nonostante l'introduzione di SPECK, gli *zerotree coder* restano gli *zeroset coder* più studiati. I buoni risultati di SPIHT hanno portato ad una riflessione più attenta sui principi ispiratori. Oltre all'importante contributo di Danyali e Mertins per quel che riguarda l'implementazione di funzionalità attraverso liste multiple, segnaliamo il lavoro di He, Dong, Zheng e Gao [9] e quello di Cho e Pearlman [12] per le riflessioni sul progetto degli *alberi descrittori*.
Nel primo, sulla scia di [7], è presentato un più accurato studio sulle specifiche di progetto degli *alberi descrittori*. Nei lavori di Cho e Pearlman invece si introduce il concetto di *ordine dello zerotree* che riprenderemo ampiamente in seguito.

4.2.6 Capire gli zeroset coder partendo dagli zerotree coder

L'esperienze di successo fino ad ora presentate non sono valse tuttavia a far entrare gli *zeroset coder* negli standard. Forse la teoria non è sufficientemente assestata per portare al massimo l'efficienza e soprattutto per garantire una certa robustezza rispetto alle statistiche, non sempre prevedibili, dei dati. In qualche modo, per superare questi limiti, gli *zeroset coder* dovrebbero poter diventare in una certa misura adattativi. Non si puo' però comprendere come opportunamente riprogettarli se non è ben chiaro un modello unico che permetta di analizzarne e capirne le proprietà fondamentali. In questo capitolo verranno analizzate con attenzione le proprietà degli *zeroset coder*, soffermandoci però con maggiore attenzione sugli *zerotree coder*, i quali sono stati fino ad ora approfonditamente studiati in letteratura.

4.3 Codificatori gerarchici

Introduciamo i *codificatori gerarchici*. Essi presiedono alla codifica della *significatività* negli *zeroset coder*, come sarà chiarito in seguito.

Esporremo nel corrente paragrafo:

- le caratteristiche generali del modello che proponiamo per descrivere il partizionamento gerarchico di insiemi: **HPM** (*Hierarchical Partitioning Model*);
- i principi di funzionamento dei *codificatori gerarchici*;

Gli *zeroset coder* ed gli *zerotree coder* saranno invece specificamente trattati nei paragrafi successivi.

4.3.1 Il modello di partizionamento gerarchico

Enunciamo le seguenti definizioni ed i seguenti teoremi per ottenere una struttura concettualmente robusta ma progettualmente flessibile (l' HPM), alla quale ricondurre le tecniche di *zeroset coding*. L'HPM è un modello impiegabile per vari scopi. In questo sottoparagrafo lo introdurremo in modo informale come modello per la descrizione (informazionalmente efficiente) di una *classificazione* M-ria. Ricordiamo che una **classificazione** è una associazione molti ad uno tra gli elementi di un certo insieme X ed un insieme discreto di etichette C, dette **classi**. Nel paragrafo successivo, invece, verranno esposte le proprietà dell'HPM come modello puramente insiemistico, trascurando l'aspetto applicativo.

Sia dunque X un insieme di N elementi da classificare in M classi. Se volessi rappresentare la mia *classificazione* semplicemente elencando l'appartenenza di ciascun elemento di X alla classe m-sima, avrei bisogno di $\log_M N$ bit. Posso ridurre il numero di bit necessari, quindi la complessità della descrizione, associando invece un codice a ciascuna delle possibili M^N *classificazioni* di X in base alla probabilità di realizzazione.
Una singola *classificazione* di X è rappresentabile come l'unione di un certo numero di *cluster*. I **cluster** di una certa *classe* m sono tutti e soli gli insiemi che costituiscono una partizione di m secondo un certo criterio.
Se i possibili *cluster* sono tutt'altro che equiprobabili, l'HPM puo' essere un'utile modello statistico delle possibili *classificazioni* di X. Sfruttando l'HPM, la generica *classificazione* è rappresentabile mediante un elenco di coppie

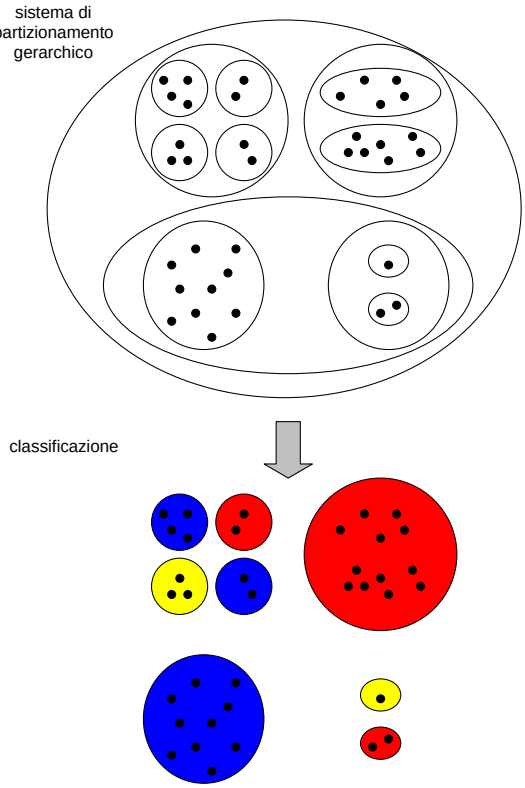


Figura 4.1: *Classificazione* ottenuta con un *modello di partizionamento gerarchico*.

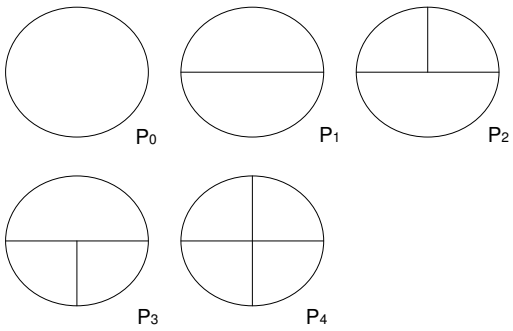


Figura 4.2: Un esempio di sistema di **partizionamento gerarchico**: la famiglia S^*.

cluster-classe, lì dove l'unione di tutti i cluster è chiaramente una partizione di X. Tutte le partizioni ammessibili costituiscono $S(X)$, cioè un *sistema di partizionamento* di X. $S(X)$ si dice un *sistema gerarchico* se rispetta alcune proprietà formali descritte nel paragrafo successivo, che consentono di mappare tutte le partizioni ammissibili in un *sistema di alberi di partizionamento* $T(X)$. In $T(X)$, ad ogni nodo corrisponde un *cluster* ammesso dal sistema. Tale struttura gode della seguente proprietà fondamentale: "ciascuna partizione è ottenibile come l'insieme delle foglie di un opportuno troncamento del sistema di alberi". Alberi più grandi descrivono dunque partizioni (e dunque *classificazioni*) più complesse. La citata proprietà è molto importante per la rappresentazione efficiente di una *classificazione*, in quanto è possibile descrivere la stessa tramite la successione di operazioni di suddivisione che servono a generarla a partire dai *cluster primitivi* $\{X_n\}$ associati alle radici degli alberi del sistema $T(X)$. Se le partizioni più semplici sono più probabili di quelle complesse allora l'HPM introduce una codifica efficiente. Questo è informalmente il principio di funzionamento dei *codificatori gerarchici*.

4.3.2 Aspetti formali dell'HPM

Lo scopo di questa sezione è la definizione formale delle proprietà insiemistiche di un *sistema di partizionamento gerarchico*. E' inoltre giustificata la rappresentazione mediante un *sistema di alberi di partizionamento*.

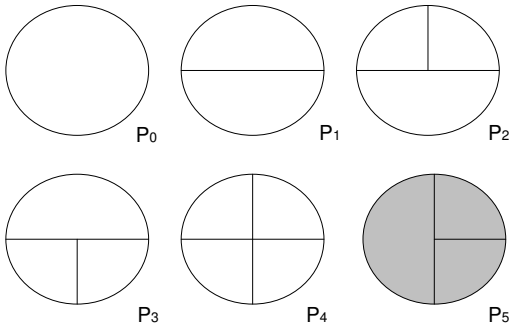


Figura 4.3: Una famiglia di partizioni che non costituisce un *sistema di partizionamento gerarchico* a causa della **mancata coerenza**: le partizioni P_3 e P_5 presentano insiemi con intersezioni non legati da relazioni di inclusione.

Definizione 2. *Sia X un insieme con un numero finito di elementi. $S(X) = \{P_n(X)\}$ sia una famiglia (necessariamente di cardinalità finita) di partizioni di X. $S(X)$ è un **sistema di partizionamento gerarchico** se valgono le due seguenti proprietà:*

coerenza: *per una coppia qualsiasi di partizioni $P_k(X)$ e $P_h(X)$ di $S(X)$, non è possibile trovare due insiemi distinti I ed I' rispettivamente appartenenti a $P_k(X)$ e a $P_h(X)$, tali che $I \cap I' \neq \emptyset$;*

completezza: *ad eccezione di un unica partizione $P_{min}(X)$, detta **partizione primitiva**, per ciascuna partizione $P_k(X)$ di $S(X)$ esistono almeno una partizione $P_h(X)$ ed un insieme $I \in P_h(X)$, tale che $P_k(X)$ è ottenibile da $P_h(X)$ mediante un partizionamento di I.*

$P_{min}(X)$ è chiaramente la partizione di $S(X)$ con minore cardinalità. La partizione $P_k(X)$ si dice invece **generata** da $P_h(X)$ attraverso I. Se la partizione $P_k(X)$ è *generata* (attraverso I) solo da $P_h(X)$ e da nessun altra partizione si dice **generata direttamente** (attraverso I) da $P_h(X)$.
Se $I \subset X$ è un sottoinsieme appartenente a $P(X)$, indichiamo ora con $P(I)$ la partizione del solo I costituita degli insiemi di $P(X)$ a cui appartengono tutti e soli i sottoinsiemi di I.

Teorema 4. *Presa la generica partizione $P_k(X)$ di un* sistema di partizionamento gerarchico $S(X) = \{P_n(X)\}$, *per ciascun insieme $I \in P_h(X)$,*

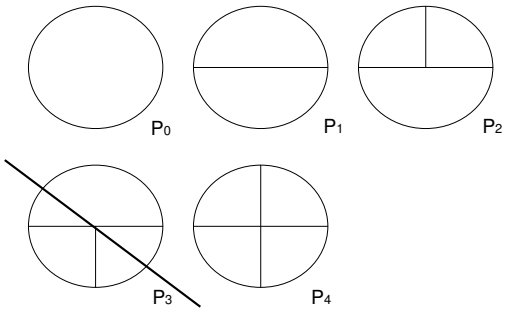


Figura 4.4: Una famiglia di partizioni che non costituisce un *sistema di partizionamento gerarchico* a causa della **mancata completezza**: non è possibile generare P_4 a partire dalla suddivisione di una qualsiasi tra le altre partizioni della famiglia P_0, P_1 e P_2. Se anche P_3 facesse parte della famiglia essa sarebbe *completa* e *coerente*.

$S(I) = \{P_n(I)\}$ *è ancora un* sistema di partizionamento gerarchico, *detto* ***sistema estratto***.

Le proprietà di coerenza e di completezza devono valere chiaramente per tutti i sistemi estratti da $S(X)$, in quanto se così non fosse tali proprietà, per definizione, non varrebbero nemmeno per lo stesso X. Non dimostriamo per esteso il teorema ma illustriamo qui in breve il senso di una possibile dimostrazione. Si prenda in considerazione invece del sistema $S(I)$, il sistema $S'(X) \subset S(X)$ dove l'unica partizione ammessa per $X - I$ è arbitrariamente fissata a $P_0(X - I)$. Se la completezza o la coerenza di $S'(X)$ dipendessero dal particolare $P_0(X-I)$, significherebbe che qualche insieme di I si possa ottenere come partizione di un insieme J, ammesso da $S(X)$, con elementi sia in I che in $X-I$, la qual cosa contraddice le definizioni di coerenza e completezza per X. Ciò comporta la coerenza e la completezza per $S'(X)$ qualsiasi sia $P_0(X - I)$. Poiché la coerenza e la completezza di I sono indipendenti dal particolare complemento $X - I$, in quanto, come visto dipendono solo dai sottoinsiemi di I, sostituendo ad $X - I$ l'insieme $\emptyset$ le suddette proprietà devono ancora valere. Da qui si dimostra l'asserto.

Definizione 3. *Il concetto di generazione si puo' estendere alla* ***famiglia degli insiemi del sistema*** *e cioè quelli appartenenti alle partizioni di $S(X)$, che d'ora indicheremo con $F(X) = F(S, X)$. Si dice infatti che gli insie-*

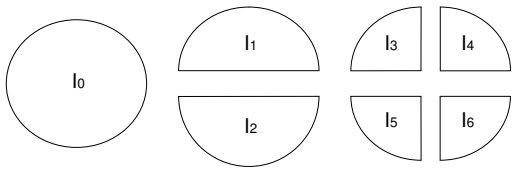


Figura 4.5: La famiglia F^* di tutti gli insiemi presenti nel sistema di partizionamento gerarchico S^* della figura 4.2.

mi $\{I_1, ..., I_M\}$ *sono* ***generati*** *da* I *se* $P_k(I) = \{I_1, ..., I_M\}$ *è* generata *da* $P_h(I) = \{I\}$ *attraverso* I. *Allo stesso modo gli insiemi* $\{I_1, ..., I_M\}$ *sono* ***generati direttamente*** *da* I *se nessuno di loro puo' essere generato da un altro sottoinsieme di* I.

Teorema 5. *Per ciascun insieme* I *di* $F(X)$ *esiste un unico insieme* I' *di* $F(X)$ *da cui* I *è* generato direttamente.

Dimostrazione. Se non fosse così, esisterebbe un altro insieme I'' da cui I è *generato direttamente*. I'' non deve includere I' perchè I' genera I direttamente, né naturalmente ne deve essere incluso perchè anche esso genera direttamente I. I farebbe parte dunque dell'intersezione tra I' ed I'', intersezione diversa da I', ciò in contraddizione con la proprietà di coerenza. □

Teorema 6. *Ciascun insieme* I *di* $F(X) = F(S, X)$ *che ne genera altri, ne deve generare almeno due direttamente.*

Dimostrazione. Un insieme I non ne puo' generare uno solo per definizione. Quindi se genera I' deve generare assieme ad esso altri insiemi. $I - I'$ deve essere infatti ricoperto da almeno una partizione $P_h(I - I')$ ricavata da $P_h(X)$ e di cui I' fa parte. $P_h(I - I')$ deve contenere almeno un insieme, per cui il numero minimo di insiemi generati da I è due. Se I non generasse insiemi direttamente, per ciascun insieme J generato da X esisterebbe un altro insieme J' generato da X che genera J. Ciò è naturalmente valido anche per J' e così via. In questo modo la cardinalità di $S(X)$ sarebbe infinita, contro la definizione di *sistema di partizionamento gerarchico.* □

Teorema 7. *Sia* $P_{max} = \{x_n\}$ *la partizione a massima cardinalità di* $S(X)$ *o* ***partizione fondamentale****. Tutti gli altri insiemi di* $F(S, X)$ *generano almeno un insieme di* P_{max}.

Dimostrazione. Se non fosse così, esisterebbe un insieme I che non genera insiemi di $P_{max} = \{x_1, ..., x_N\}$. Preso il generico J_k, I per coerenza o lo contiene o ne è contenuto. Se però I contenesse Y_k dovrebbe contenere anche tutti gli insiemi di P_{max} inclusi in $I - x_k$. Sicuramente ve ne sono in P_{max} a costituire una copertura di $I - x_k$, in quanto P_{max} è una partizione dell'insieme X, e sicuramente essi costituiscono anche una partizione di $I - x_k$ per *coerenza*. Dunque se I contiene J_k lo genera anche.
D'altra parte se I fosse incluso in x_k esisterebbe almeno un insieme x_1 incluso in $x_k - I$ e generato da J_k. $J_k - I$ deve essere infatti ricoperta dalla partizione $P_h(x_k - I)$ per completare la partizione $P_h(X)$ di cui I fa parte. A questo punto se considerassimo la partizione $P_h(X) = (P_{max} - x_k) \cup P_h(x_k - I)$, essa avrebbe sicuramente una cardinalità maggiore di quella di P_{max}, contraddicendo l'ipotesi di partenza per cui P_{max} è la partizione a massima cardinalità del sistema. □

Gli insiemi $\{x_n\}$ di P_{max} vengono detti **insiemi fondamentali** di $S(X)$. Se gli insiemi fondamentali sono tutti i singoletti di X allora $S(X)$ è detto **partizionamento esaustivo**.

Teorema 8. *Ciascun sistema di partizionamento gerarchico $S(X)$ è rappresentabile mediante un* sistema di alberi $T(X) = T(S, X)$.

Dove il **sistema di alberi di partizionamento** T è definito come segue.

Definizione 4. *Sia $S(X) = \{P_k(X)\}$ un* sistema di partizionamento gerarchico *di X. Sia invece $P_{min} = \{X_n\}$ la* partizione primitiva *di X. Ciascuno degli insiemi X_n di P_{min} è detto **insieme primitivo**. $T(X) = \{T(X_n)\}$, **sistema di alberi** di $S(X)$, è costruito nel seguente modo:*

- *a ciascun* insieme primitivo *X_n è associato un albero $T(X_n) = T(S, X_n)$;*
- *i nodi dell'albero $T(X_n)$ sono tutti e soli gli insiemi $F(X_n)$ delle partizioni appartenenti a $S(X_n)$;*
- *il generico nodo di $T(X_n)$ associato all'insieme I genera a sua volta i nodi associati a tutti gli insiemi* generati direttamente *da I.*

Dimostrazione. Per dimostrare il teorema 8 si proceda alla seguente costruzione. Si prenda P_{max}, *partizione fondamentale* di $S(X)$. Si colleghi

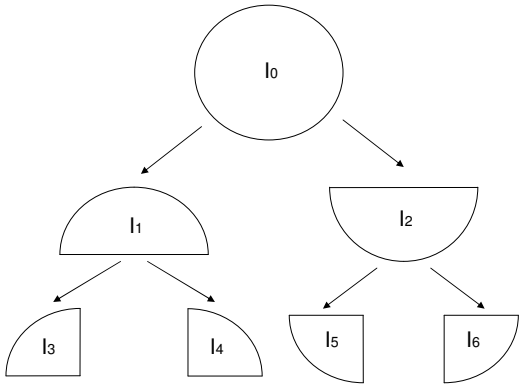


Figura 4.6: Il *sistema di alberi* T^* associato al sistema di partizionamento S^*. Poiché la *partizione primitiva* di S^* è costituita dall'unico insieme I_0, il sistema conta un unico albero.

ciascun *insieme fondamentale* all'insieme da cui esso è direttamente generato in un rapporto nodo figlio nodo padre, dove il nodo figlio è l'insieme generato ed il nodo padre l'insieme generatore. Così via per tutti gli insiemi che così vengono chiamati in causa. Il teorema è dimostrato se nessun insieme viene escluso durante la costruzione.
Per assurdo un insieme verrebbe escluso dalla costruzione soltanto se non ci fosse nessun insieme fondamentale *generato* da esso, la qual cosa è in contraddizione con il teorema 7, o in alternativa se non ci fosse nessun insieme che ne genera direttamente un altro, ipotesi mai verificata in virtù del teorema 6. □

Teorema 9. *I nodi di un* sistema di alberi*, le cui radici sono una partizione di* X*, rappresentano gli insiemi di un* sistema di partizionamento gerarchico*.*

Dimostrazione. Tutti gli insiemi associati ai nodi sono o l'uno incluso nell'altro o disgiunti (*coerenza*). Ciascun insieme si puo' generare da un altro (*completezza*). Il numero di insiemi e' finito, quindi anche il possibile numero di partizioni (finitezza). □

Definizione 5. *Un* ***troncamento*** $T'(X)$ *di un sistema di alberi* $T(X)$ *è ottenuto "potando" tutti i rami di* T *generati da un insieme di nodi* $P = \{I_1, ..., I_N\}$ *associati ad insiemi disgiunti.*

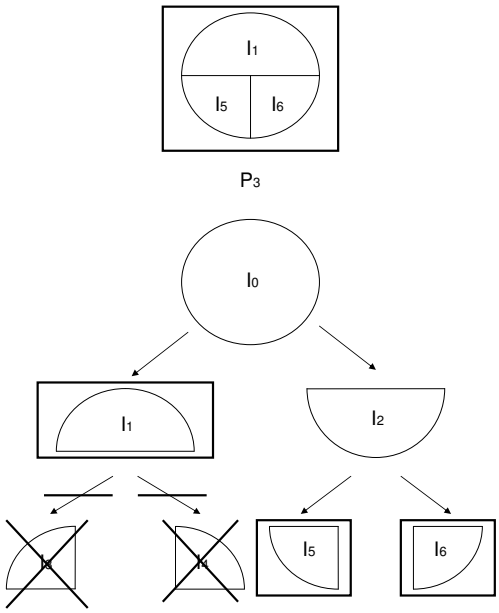


Figura 4.7: La partizione P_3 di S^*, viene ottenuta *troncando* l'albero associato. Il *troncamento* consiste nell'eliminazione di tutta la discendenza di un determinato nodo, che in questo caso è quello associato all'insieme I_1.

Teorema 10. *Sia* $T'(X)$ sistema di alberi di partizionamento *costituito da un* troncamento *degli alberi di* $T(X) = T(S, X)$ *attraverso gli insiemi* $\{I_1, ..., I_N\}$. $T'(X)$ *è ancora un* sistema di partizionamento gerarchico. *Inoltre tutte le foglie degli alberi appartenenti a* $T'(X)$ *costituiscono una delle partizioni* $P_k(X)$ *di* $S(X)$. *Tutte le partizioni* $P_k(X)$ *di* $S(X)$ *sono ottenibili troncando* $T(X)$.

Dimostrazione. $T'(X)$ presenta ancora tutte le proprietà che definiscono $T(X)$ per cui i suoi nodi costituiscono ancora un *partizionamento gerarchico* di X. I nodi foglia sono associati ad insiemi disgiunti. Resta da dimostrare, affinchè essi costituiscano una partizione, che non esiste nessun insieme di P_{max} (*partizione fondamentale*) non incluso in uno degli I_k. In effetti il generico insieme fondamentale x_h o è uno degli I_k o appartiene ad uno dei rami recisi, per cui essendo generato da I_k è strettamente incluso in esso.

Se esistesse una partizione $P_k(X)$ di $S(X)$ non ottenibile dal *troncamento* di $T(X)$ significherebbe o che gli insiemi di quella partizione non sono nodi di $T(X)$ o che almeno una coppia di essi è tale che uno generi l'altro. Ciò è palesemente falso per costruzione. □

Ne deriva il seguente corollario.

Teorema 11. *Un* sistema di partizionamento gerarchico *è univocamente definito o da un* sistema di alberi di partizionamento, *in base alla* regola di troncamento *esposta nella definizione 5.*

4.3.3 Qualche considerazione storica sugli alberi

Un *sistema di partizionamento gerarchico* $S(X)$ è dunque una famiglia di partizioni di un insieme X definito univocamente o dalle proprietà esposte nella definizione 2 o dal *sistema di alberi di partizionamento* costruito nella definizione 4.
Meno formalmente si potrebbe definire un *sistema di partizionamento gerarchico* come un insieme di partizioni di un insieme X generate da una suddivisione progressiva di una partizione iniziale P_{min}. La procedura di suddivisione è comodamente rappresentata da uno schema ad albero, i cui nodi sono gli insiemi ammessi nelle diverse partizioni possibili. Le diverse partizioni ammesse sono quelle costituite dalle foglie di ciascun *troncamento* dell'albero espresso nella definizione 5.
Si noti come il generico albero del sistema $T(X)$ appena introdotto sia ben diverso dall'albero definito in [4] e in [5]. Ciascun nodo di T è infatti un insieme, mentre in [4] e in [5] i nodi sono gli stessi coefficienti della trasformata, o meglio le loro coordinate.
La nuova struttura presenta il vantaggio di essere valida non solo per gli *zerotree coder* (come [4] e [5]) ma per i più generali *zeroset coder*, alla cui categoria appartiene ad esempio [4]. Per gli *zerotree coder* comunque introdurremo anche la struttura ad albero di [4] e [5], denominandola *albero descrittore della trasformata* ed indicandola con il simbolo Y. Vedremo poi come la struttura Y serva a ricavare in maniera iterativa e ricorsiva il *sistema di alberi di partizionamento* T.

4.3.4 Principio di funzionamento dei codificatori gerarchici

I **codificatori gerarchici** codificano una certa *classificazione* dei dati tramite l'HPM. Abbiamo già visto come il *partizionamento gerarchico* possa essere

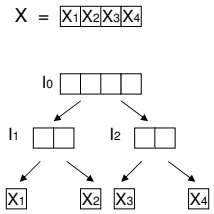


Figura 4.8: Esempio di sistema di alberi di partizionamento associata ad un **codificatore gerarchico di simboli binari**. Considerata una sequenza binaria X si possono individuare le sottosequenze omogenee più probabili di 1 e di 0. Tra queste si possono scegliere le sottosequenze più adatte a costituire un *sistema di partizionamento gerarchico* di X, in modo che le realizzazioni più probabili di X siano descrivibili come una breve successione di sottosequenze omogenee.

impiegato per la codifica di dati M-ari. Consideriamo ora il caso binario ($M = 2$).

Si supponga di avere una sequenza X di variabili aleatorie binarie e si supponga di conoscere la proabilità di ciascuna realizzazione. Supponiamo inoltre che si verifichi che sottosequenze lunghe di 0 e di 1 siano molto più probabili che sottosequenze brevi.
Se ciò si verifica sarà possibile descrivere le realizzazioni più probabili di X come una successione breve di sottosequenze omogenee di 0 e di 1. Le realizzazioni meno probabili di X saranno invece successioni di sottosequenze lunghe.
Se posso individuare tra le possibili sottosequenze quelle con più alta probabilità di essere omogenee, posso tentare di costruire un *sistema di partizionamento gerarchico esaustivo* di $S(X)$, i cui insiemi sono proprio le sottosequenze ad alta probabilità di omogeneità. Il sistema deve essere esaustivo se nessuna realizzazione di X ha probabilità nulla di verificarsi.

Una codifica efficiente consisterà dunque:

- nella codifica della partizione P_k corrispondente alla corretta successione di sottosequenze omogenee;

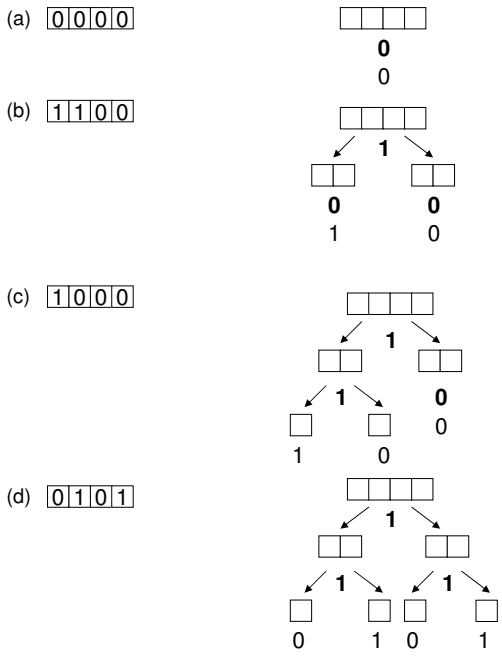


Figura 4.9: Principio di funzionamento dei **codificatori gerarchici**. La codifica della realizzazione delle sequenza X avviene codificando il *troncamento* T_k associato alla partizione P_k ed il valore degli insiemi della partizione. T_k è codificato in figura associando 1 alle *generazioni dirette* e 0 ai troncamenti (cifre in neretto). Le cifre non in neretto indicano invece il valore degli elementi di ciascun insieme, che puo' essere 0 o 1. In figura esempi di codifica per le realizzazioni 0000, 1100, 1000 e 0101.

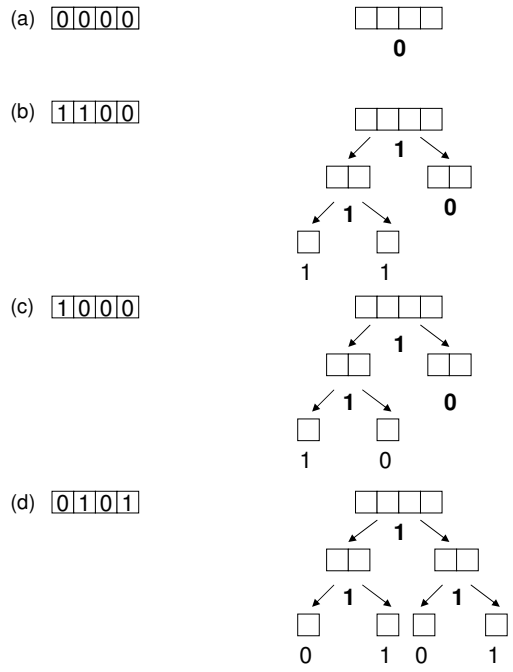


Figura 4.10: Principio di funzionamento degli **zeroset coder**. Se le sequenze lunghe sono soprattutto di bit 0, le sequenze di 1 di lunghezza non unitaria non vengono prese in considerazione. Gli insiemi a cardinalità maggiore di 1 sono dunque tutti di bit 0 ed il valore 0 degli elementi dell'insieme puo' essere omesso. Per gli insiemi a cardinalità unitaria invece va indicato esplicitamente il valore 0 oppure 1. In figura esempi di codifica per le realizzazioni 0000, 1100, 1000 e 0101.

- nel corretto etichettamento di ciascuna sottosequenza come sottosequenza di 1 o di 0.

In particolare la partizione P_k puo' essere codificata sfruttando il sistema di alberi T associato ad $S(X)$, mediante una descrizione efficiente del *troncamento* T_k associato a P_k. Ad esempio indicando ordinatamente (a partire dalla radice) con 1 ciascuna *generazione diretta* che non sia stata esclusa dal *troncamento*, e con 0 l' eventuale potamento del ramo, si puo' dimostrare come qualsiasi *troncamento* possa essere codificato.
Tanto più una realizzazione della sequenza X è omogenea, tanto più corta è la sequenza di bit necessaria a codificare il *troncamento* T_k. Ciò risulta particolarmente vantaggioso dal momento che abbiamo ipotizzato che le sequenze più omogenee siano anche quelle più probabili.

Facciamo ora l'ulteriore ipotesi che sequenze lunghe di zeri siano molto più probabili che sequenze lunghe di 1. Puo' essere conveniente omettere l'etichetta 0 del tipo di sequenza, non prendendo in considerazione sequenze di lunghezza superiore all'unità per i bit 1. Le sequenze omogenee di 1 dovranno essere codificate elemento per elemento, ma le sequenze omogenee di 0 non necessiteranno di un bit di etichetta che ne descriva il contenuto. Sotto questa ulteriore ipotesi vale anche che quante più sottosequenze a valore 0 vi sono nella realizzazione di X tanto più essa risulta semplice da codificare. Questo è il principio di funzionamento dei codificatori gerarchici utilizzati dagli *zeroset coder* per la codifica della significatività.

4.4 Zeroset coder

Il modello di *zeroset coder* che proporremo rispetta le seguenti specifiche:

- generalità rispetto alle tecniche di partizionamento;
- generalità rispetto alla dimensionalità dei dati;
- generalità rispetto al tipo di trasformata;
- implementabilità di funzionalità quali: scalabilità in qualità, scalabilità in risoluzione, accesso diretto a regioni di interesse.

Per semplicità di trattazione il tipo di rappresentazione e approssimazione dei coefficienti trasformati, sebbene generalizzabile a quello ottenibile da un quantizzatore a reticolo, sarà fissato a quello ottenibile da un quantizzatore uniforme a *zona morta*. Ogni coefficiente dunque si deve intendere idealmente rappresentato con virgola fissa, in segno e modulo. Il bit di **segno** sarà 0 se il coefficiente è positivo, 1 altrimenti. La sequenza dei bit del modulo sarà spezzata in due sottosequenze. Tutti gli 0 iniziali ed il primo 1 costituiscono i bit di **significatività**, i bit rimanenti quelli di **raffinamento**. Tutti i bit del modulo in posizione n-sima costituiscono il **bitplane** n dell'insieme di coefficienti considerato.
I coefficienti così rappresentati possono essere approssimati impiegando solo i primi N bit di modulo (quantizzazione uniforme). Il bit di segno diventa necessario solo allorquando i primi N bit di modulo non sono tutti nulli, cioè solo quando il coefficiente è *significativo* al *bitplane* N-simo.
Lo scopo degli algoritmi basati su *partizionamento gerarchico*, come già prima accennato, è la codifica efficiente dei bit di *significatività*, i bit di *segno* sono eventualmente compattati mediante algoritmi di codifica entropica, mentre i bit di *raffinamento* solitamente sono codificati esplicitamente.

Gli **zeroset coder** sono appunto codificatori binari che adottano una codifica gerarchica della significatività, il cui *sistema di partizionamento* prevede insiemi a valore 1 solo di cardinalità unitaria (singoletti) ed insiemi a valore 0 (**zeroset**) di cardinalità non necessariamente unitaria.
Poiché, oltre alla significatività, bisogna codificare anche il segno ed i bit di raffinamento, lo *zeroset coder* alternerà alla codifica gerarchica della significatività la codifica (esplicita) delle altre informazioni. Possiamo dunque vedere lo *zeroset coder* come composto da un *codificatore gerarchico* e da un codificatore in chiaro, tra loro opportunamente combinati.

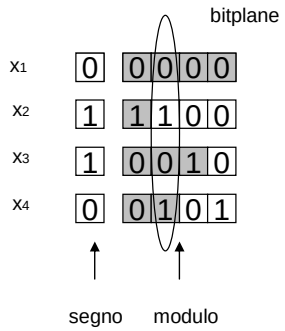


Figura 4.11: Rappresentazione in segno e modulo. Ombrati i **bit di significatività**.

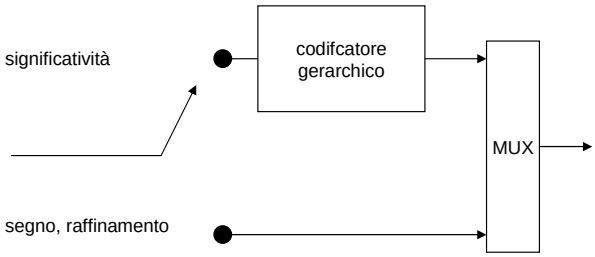


Figura 4.12: Schema di principio di uno **zeroset coder**.

4.4.1 Implementazione con liste

Gli *zeroset coder* descrivono la sequenza dei bit di significatività come un'unione di diverse sottosequenze. Queste in particolare sono:

- sequenze di più di un elemento di bit 0, dette **sequenze non significative** (**IS** = *Insignificant Sequence*);
- sequenze di un unico elemento con valore 0, detti **elementi non significativi** (**IE** = *Insignificant Element*);
- sequenze di un unico elemento con valore 1, detti **elementi significativi** (**SE** = *Significant Element*).

Si usa inoltre anche il termine **sequenza significativa**. Una sequenza si dice *significativa* quando almeno un suo elemento è *significativo*.

Così come suggerito da Said e Pearlman in [5], per implementare uno *zeroset coder* è possibile introdurre tre liste:

- **LIS**: *List of IS*, la lista che contiene tutti gli IS ad un certo *bitplane* n;
- **LIE**: *List of IE*, la lista che contiene tutti gli elementi ancora non significativi ad un certo *bitplane* n;
- **LSE**: *List of SE*, la lista che contiene tutti gli elementi già significativi nei *bitplane* precedenti.

La LIS e la LIE servono a classificare il *bitplane* corrente in elementi significativi e non. La *classificazione* è descritta mediante l'enumerazione degli *zeroset*.
Scelto il sistema di partizionamento il codificatore deve essere in grado di impiegare sempre la descrizione più sintetica, e dunque la partizione a minor cardinalità possibile del sistema di partizionamento.
La LSE elenca invece gli elementi significativi la cui rappresentazione, passando da un *bitplane* al successivo, deve essere raffinata.

L'algoritmo 1 rappresenta una versione abbastanza generale di *zeroset coder* ed è implementato in modo da codificare anche i bit di *rappresentazione* oltre a quelli di *significatività*. L'algoritmo 1 genera un flusso *embedded* di bit scalabile in qualità, codificando un *bitplane* per volta. La codifica dei *bit di segno* viene effettuata ogni qual volta si ottiene un SE durante l'esplorazione

della LIE o della LIS. I bit di rappresentazione del modulo per il *bitplane* corrente sono invece codificati per ciascun elemento della LSE. Quest'ultima operazione è detta di **raffinamento**.

Algoritmo 1 Zeroset coder: implementazione con liste.

procedure ZEROSETCODER(X,b_{quant})
  Inizializza(LIS, LIE);
  $B := \log_2 \max_X\{|x|\}$;
  $T := 2^B$;
  $bitplane := 0$;
  while $bitplane \leq b_{quant}$ **do**
    EsploraLIS(LIE, LSE);
    EsploraLIE(LIE, LIS, LSE);
    EsploraLSE(LSE);
    $bitplane := bitplane + 1$;
    $T := T/2$;
  end while
end procedure
! X coefficienti trasformati, b_{quant} massimo numero di bit di risoluzione del quantizzatore uniforme *a* zona morta.

Algoritmo 2 Scansione della LIE.

procedure ESPLORALIE(LIE,LSE)
  for each $x \in LIE$ **do**
    CodificaSignif(x);
    if Signif(x) $= 1$ **then**
      Accoda(LSE, x);
      CodificaSegno(I);
    end if
  end for
end procedure

I bit di significatività del *bitplane* n si ottengono esplorando gli IE e gli IS al *bitplane* $(n-1)$, dunque scorrendo e modificando la LIE e la LIS. Gli elementi della LIE vengono o confermati come IE oppure riclassificati come SE e trasferiti nella LSE. Gli elementi della LIS invece, qualora non siano più

Algoritmo 3 Scansione della LIS.

```
procedure ESPLORALIS(LIE,LIS,LSE)
    for each I ∈ LIS do
        t := Test(I)
        CodificaTest(t);
        if Signif(I) = 1 then
            {I_1, ..., I_N} := GeneraPartizione(I, t);
            for each k ∈ {1, ..., N} do
                if Dim(I_k) = 1 and Signif(I_k) = 0 then
                    CodificaSignif(I_k);
                    Accoda(LIE, I_k);
                else if Dim(I_k) = 1 and Signif(I_k) = 1 then
                    CodificaSignif(I_k);
                    CodificaSegno(I_k);
                    Accoda(LSE, I_k);
                else !caso in cui Dim(I_k) > 1
                    Accoda(LIS, I_k);
                end if
            end for
            Elimina(LIS, I);
        end if
    end for
end procedure
```

Algoritmo 4 Scansione della LSE.

```
procedure ESPLORALSE(LSE)
    for each x ∈ LSE do
        output mod_2[⌊|x|/T⌋];
    end for
end procedure
```

sottosequenze omogenee di 0, vengono suddivisi in sottosequenze più corte *generate* dalla sequenza originaria, così come previsto dal *sistema di alberi di partizionamento*. Quale generazione effettuare tra quelle ammesse, viene deciso in base ad un test sulla significatività degli elementi dell'IS.
Se si rileva, tra le sottosequenze, un SE, dopo averne emesso il segn esso viene inserito nella LSE, in quanto al *bitplane* $(n+1)$ la significatività del coefficiente non dovrà essere più testata mentre lo stesso dovrà essere raffinato. Le generazioni continuano finchè non si isolano tutti i SE. Le sottosequenze di tipo IE vengono inserite nella LIE, mentre quelle di lunghezza maggiore di uno vengono accodate nella LIS in attesa di venir definitivamente classificate come di tipo IS per il *bitplane* corrente oppure di essere ulteriormente decomposte. Gli algoritmi 2,3,4 esprimono nel dettaglio tutte queste operazioni.

Nello pseudocodice riportato nell'algoritmo 3 la funzione **Signif(.)** è un test binario che restituisce 1 se l'insieme testato è *significativo*, la funzione **Test(.)** è un test generico sulla significatività dei singoli elementi dell'insieme, la procedura **GeneraPartizione()** ottiene invece una partizione dell'insieme tra quelle ammesse dal sistema di partizionamento in funzione del risultato del test. Si evidenzia che in letteratura, l'unico test finora proposto per la funzione Test(.) è il test Signif(.), mentre l'unico tipo di generazione prevista dalla funzione GeneraPartizione(.) è quella diretta. A nostro giudizio queste scelte sono troppo restrittive. Vedremo in seguito come scelte più generali comportino a volte una maggiore efficienza degli algoritmi di codifica.

4.4.2 Implementazione con array di liste.

Abbiamo già visto come, attraverso la gestione di alcune liste, si possa realizzare uno generico *zeroset coder*. Uno *zeroset coder* così come implementato dall'algoritmo 1 dispone i bit codificati ordinatamente per *bitplane*. Tuttavia potrebbero esserci differenti altri criteri di ordinamento (slice, livello di risoluzione spaziale, canale, etc...) che si potrebbero voler implementare.
A questo scopo è possibile associare allo spazio dei coefficienti trasformati X una mappatura in classi di equivalenza. Supposta ciascuna classe di equivalenza individuata da N indici $\{c_1, ..., c_N\}$ appartenenti al prodotto cartesiano di insiemi $C_1 \times ... \times C_N$ l'algoritmo più generale è il 5.

Le classi di equivalenza impongono un ordinamento parziale via via più stringente quanto maggiore è il numero delle stesse. Quando le classi di equivalenza coincidono proprio con ciascun *zeroset* ammesso dal *sistema di par-*

Algoritmo 5 Zeroset coder: implementazione con array di liste.

procedure ZEROSETCODER(X,b_{quant})

! inizializzazione

while $bitplane \leq b_{quant}$ **do**

for each $c_1 \in C_1$ **do**
...
for each $c_N \in C_N$ **do**
EsploraLIE($(c_1, ..., c_N), LIE[.], LSE[.]$);
end for
...
end for

for each $c_1 \in C_1$ **do**
...
for each $c_N \in C_N$ **do**
EsploraLIS($(c_1, ..., c_N), LIE[.], LIS[.], LSE[.]$);
end for
...
end for

for each $c_1 \in C_1$ **do**
...
for each $c_N \in C_N$ **do**
EsploraLSE($(c_1, ..., c_N), LSE[.]$);
end for
...
end for

$bitplane := bitplane + 1$;

end while

end procedure

tizionamento, l'ordinamento è totale.
E' inoltre possibile innestare il ciclo **while** in un sottoinsieme dei cicli **for** per ottenere un ordinamento diverso da quello per *bitplane*.
L'uso di liste multiple per particolari applicazioni è riportato in [10], in [11] e in [13].

4.4.3 Ottenimento del sistema di partizionamento con tecniche di iterazione e di ricorsione

Se i coefficienti da codificare sono molti, la descrizione completa del sistema di partizionamento apparentemente potrebbe diventare eccessivamente complessa, con danni irreparabili per l'efficienza di codifica. Per risolvere il problema della complessità di descrizione vengono introdotti due principi:

iterazione : un *sistema di partizionamento elementare* viene iterato sui dati un certo numero di volte, riducendo la complessità di descrizione in maniera lineare;

ricorsione : il *sistema di partizionamento elementare* viene descritto a sua volta in maniera ricorsiva, semplificando la complessità in maniera esponenziale.

Nell'esempio in fig. 4.13 vediamo come ad una certa sequenza binaria possa essere associato un sistema di partizionamento scomponibile in forma iterativa e ricorsiva.

Con riferimento alle definzioni già date, possiamo ridefinire la tecnica iterativa e quella ricorsiva nella seguente maniera:

tecnica iterativa : la *partizione primitiva* P_{min} è costituita da insiemi equipotenti $\{X_n\}$ su ognuno dei quali è imposto lo stesso **sistema di partizionamento elementare** $P_E(X_n)$;

tecnica ricorsiva : il *sistema di partizionamento elementare* è ottenuto mediante una **regola ricorsiva elementare** di *generazione diretta* G_E.

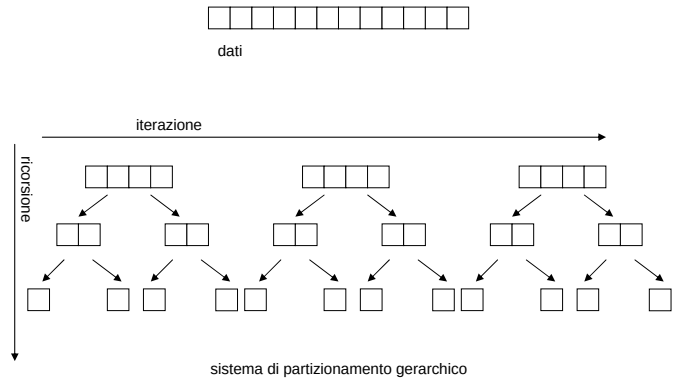


Figura 4.13: Esempio di sistema di partizionamento ottenuto con la **tecnica iterativa** e con la **tecnica ricorsiva**.

4.5 Zerotree coder

Una famiglia molto importante di *zeroset coder* è costituita dal più ristretto gruppo degli **zerotree coder**. Essi si ispirano ad una particolare organizzazione ad albero dei coefficienti trasformati che ne mette in luce alcune fondamentali proprietà. Questa particolare organizzazione, detta *albero descrittore*, permette di descrivere il sistema di partizionamento tramite la tecnica iterativa e la tecnica ricorsiva.

4.5.1 Albero descrittore della trasformata

Le proprietà della significatività dei coefficienti trasformati vanno debitamente prese in considerazione al fine di progettare nella maniera più adatta il codificatore ed in particolare il *sistema di partizionamento gerarchico* su cui esso si basa. Solitamente i coefficienti trasformati hanno le seguenti importanti proprietà:

- *rappresentazione spazio-frequenza*: i coefficienti hanno una certa indeterminazione spaziale e frequenziale, sono cioè distinti dall'appartenenza ad una certa sottobanda frequenziale e ad una certa posizione spaziale all'interno della sottobanda stessa;
- *dynamic range caratteristico della frequenza*: il modulo dei coefficienti ha dynamic range medio assai simile per sottobande dello stesso ordine frequenziale;

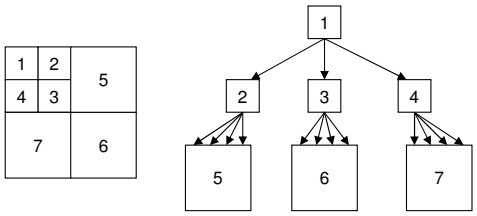


Figura 4.14: Un esempio di **albero descrittore** per una *trasformata wavelet* diadica bidimensionale.

- *dynamic range decrescente con la frequenza*: il dynamic range medio dei coefficienti decresce con la frequenza;
- *localizzazione spaziale dei coefficienti ad alta energia*: i coefficienti ad alta energia presentano una certa corrispondenza spaziale passando da una frequenza più bassa ad una più alta.

Queste caratteristiche possono essere tenute in conto attraverso una struttura ad albero descrittiva delle relazioni spazio-frequenziali tra i coefficienti, ben distinta da quella presentata per rappresentare il sistema di partizionamento, che fino ad ora abbiamo indicato con la lettera T. Denominiamo questa nuova struttura *albero descrittore della trasformata* e la indichiamo con la lettera Y.

Un *albero descrittore* in realtà è definito da proprietà piuttosto banali.

Definizione 6. *Un **albero descrittore** $Y(X)$ di un insieme di coefficienti trasformati X è una struttura ad albero i cui nodi sono tutti e soli i coefficienti di X, senza ripetizioni.*

Definizione 7. *Un **sistema di descrizione** $Q(X)$ di un insieme di coefficienti trasformati X, è un insieme di alberi descrittori $\{Y_1(X_1), ..., Y_k(X_k)\}$, con $\{X_1, ..., X_k\}$ sottoinsiemi di X che ne costituiscono una partizione.*

Come vedremo l'*albero descrittore* permette di definire la *regola di generazione elementare*, secondo i canoni della *tecnica ricorsiva*, mentre il *sistema di descrizione* permette il completamento del *sistema di partizionamento* mediante l'impiego della *tecnica iterativa*.

In realtà le proprietà introdotte sono ben lungi dal definire un *albero* o un *sistema di descrizione* con caratteristiche adeguate per la codifica.
Un *albero descrittore* è costruito invece secondo le seguenti regole pratiche:

- la radice dell'albero è uno dei coefficienti della sottobanda di più bassa frequenza;
- ciascun nodo dell'albero è un coefficiente trasformato;
- ciascun nodo dell'albero genera coefficienti di ordine frequenziale superiore;
- ciascun nodo dell'albero genera coefficienti con localizzazione spaziale a bassa indeterminazione.

L'albero così costruito gode dunque delle seguenti fondamentali proprietà:

- *energia decrescente*: l'energia dei coefficienti decresce lungo i rami;
- *energia localizzata*: qualora vi siano coefficienti ad alta energia essi si propagano lungo un ramo a partire da un certo livello dell'albero in poi.

Queste regole di buona progettazione saranno chiare allorquando il *sistema descrittore della trasformata* sarà impiegato per generare il *sistema di partizionamento gerarchico* del codificatore, così come vedremo in seguito.

4.5.2 Zerotree

Scelto un particolare *albero descrittore* Y è possibile sinteticamente definire alcuni particolari *zeroset* che risultano di largo impiego per gli *zerotree coder*. Essi sono detti *zerotree* e sono stati classificati da Cho e Pearlman in [12].

Definizione 8. *Sia j un generico indice della sequenza X da codificare ed $Y_0(j)$ il sottoalbero dell'albero descrittore Y che ha radice in j. L'insieme $O_k(j)$ è costituito da tutti i discendenti di j presenti al livello k-simo del sottoalbero $Y_0(j)$. $O_k(i)$ verrà d'ora in avanti denominato semplicemente* ***livello k-simo della discendenza*** *di i.*

Ad esempio: $O_0(i)$ è i stesso, $O_1(i)$ è l'insieme dei discendenti diretti di i, $O_2(i)$ è l'insieme dei discendenti diretti di tutti gli elementi di $O_1(i)$.

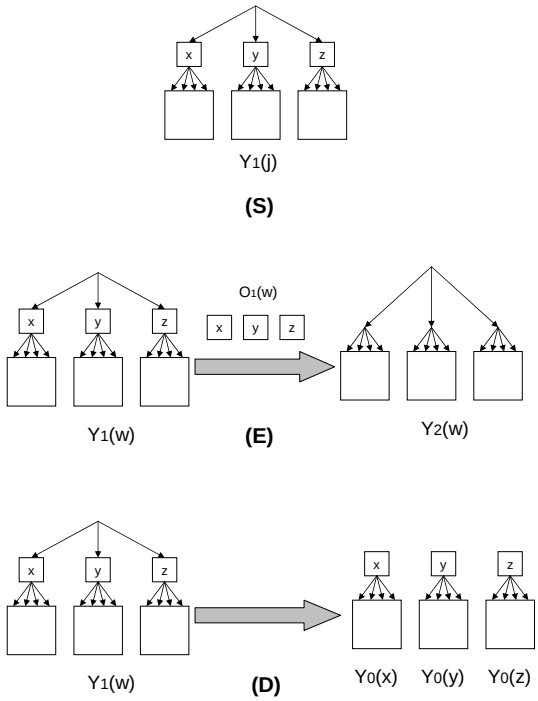


Figura 4.15: Zerotree di **ordine** 0**,** 1 **e** 2.

Definizione 9. *Sia* $Y_k(j)$ *quella parte della discendenza* $Y_0(j)$ *ottenuta da* $Y_0(j)$ *escludendovi tutti i livelli minori di* k*.* $Y_k(i)$ *è detto* ***albero di ordine*** k *relativo all'elemento* j*.*

In un *zeroset coder*, per definizione, gli insiemi di cardinalità superiori ad uno ammessi sono insiemi di coefficienti non significativi. Per questo motivo d'ora in avanti indicheremo gli alberi di ordine qualsiasi con il termine **zerotree**.

4.5.3 Dall'albero descrittore al sistema di partizionamento

Definizione 10. *Sia* X *un insieme di coefficienti trasformati. Sia* $Q(X)$ *un* sistema di descrizione della trasformata. *Sia* $S(X)$ *un* sistema di partizionamento gerarchico. *Sia infine* C *uno* zeroset coder *associato ad* $S(X)$*.*
C *è uno* ***zerotree coder*** *se e soltanto se* S *contiene solo insiemi che sono zerotree di ordine qualsiasi rispetto ad* $Q(X)$*. Se l'ordine massimo degli zerotree è pari a* K*,* C *si dice uno zerotree coder di ordine* K *rispetto ad* Y*.*

Uno *zerotree coder* è dunque vincolato ad un certo *sistema di descrizione della trasformata*, tuttavia si badi bene che non è univocamente determinato da esso. Per uno stesso *sistema di descrizione* possono esistere infatti diversi *sistemi di partizionamento*.

Il sistema di partizionamento puo' essere molto complesso ma i sistemi proposti in letteratura si basano su alcune semplificazioni:

- *iterazione*: ciascun insieme primitivo X_i è costituito da elementi mappati da uno stesso *albero di descrizione* Y^* ed è soggetto allo stesso *sistema di partizionamento* S^*, lı' dove l'*albero di descrizione* ed il *sistema di partizionamento* sono definiti a meno delle coordinate spazio-frequenziali del coefficiente radice dell'*albero descrittore*;

- *ricorsione*: la *regola di partizionamento*, attraverso la quale si stabilisce la *generazione diretta* di un insieme da un altro, è di tipo ricorsivo.

La prima affermazione significa che esiste una relazione di equivalenza tra le coordinate spazio-frequenziali dei coefficienti dei diversi *insiemi primitivi* tale che l'*albero di descrizione* ed il *sistema di partizionamento* siano sempre gli stessi sulle classi di equivalenza. La seconda esprime l'esistenza di una relazione tra le coordinate spazio-frequenziali dei coefficienti che consente di generare direttamente un insieme da un altro.

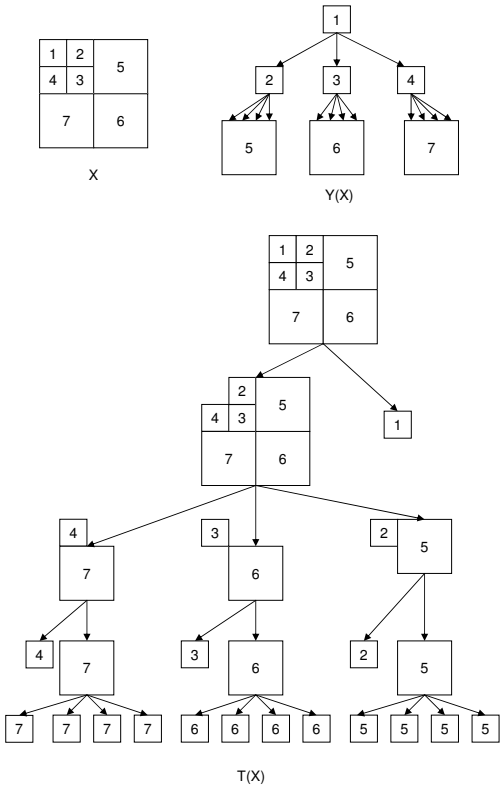


Figura 4.16: Come da un **sistema di descrizione** si puo' passare ad uno di partizionamento. Nel caso illustrato lo zerotree codec sarà di ordine 1 in quanto tutti gli zerotree sono di ordine 0 oppure 1.

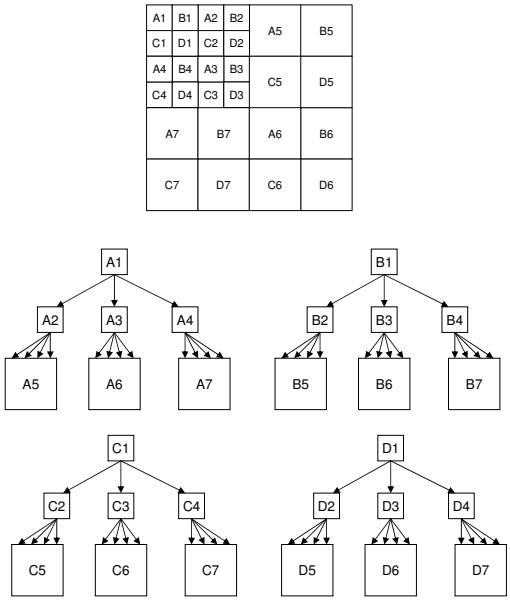


Figura 4.17: Gli zerotree coder impiegano un sistema di descrizione della trasformata basato sulla ripetizione dello stesso albero di descrizione. Nell'esempio l'**organizzazione della *trasformata wavelet* diadica bidimensionale** effettuato dall'algoritmo **EZW**.

4.6 Strumenti di progetto per zerotree coder

I concetti di *iterazione* e di *ricorsione* semplificano notevolmente il progetto di uno *zerotree coder*. L'*iterazione* consente di ridurre il problema della codifica dei coefficienti X semplicemente a quello di codifica della porzione di coefficienti riconducibili ad un singolo *albero di descrizione* Y^*, cioè quelli afferenti al generico insieme X_i della partizione primitiva.
La *ricorsione* permette invece di generare il *sistema di partizionamento* S^* del generico insieme primitivo X_i con un algoritmo particolarmente semplice.

4.6.1 Azioni elementari per la generazione diretta

Lo scopo degli *zerotree coder* è la descrizione dello spazio dei coefficienti trasformati mediante una partizione dello stesso in insiemi di zeri a cardinalità multipla (IS) o singola (IE) e coefficienti significativi (SE). Gli IS sono *zerotree* nell'*albero di descrizione della trasformata*. Se si suppone di effettuare una codifica per *bitplane*, ad ogni passo di raffinamento della risoluzione con cui i coefficienti vengono rappresentati, la partizione si modifica: via via si scende di livello lungo l'albero associato al sistema di partizionamento.
Le possibili *generazioni dirette*, a partire da un certo IS, sono ottenibili dalla composizione di alcune **azioni elementari**:

(S): **stop**, l'IS non genera nessun insieme e l'ordine dello zerotree rimane inalterato;

(E): **estrazione**, lo zerotree di ordine k genera uno zerotree di ordine $k+1$, mediante l'estrazione del livello k-simo dell'*albero descrittore*;

(D): **decomposizione**, lo zerotree di orine k viene decomposto in un certo numero di zerotree di ordine $k-1$.

L'*estrazione* comporta un test di significatività dei coefficienti estratti ed il loro inserimento nella LIE oppure nella LSE. A ciascuna di queste *azioni* (o ad una sequenza di esse) viene associata naturalmente una sequenza di bit che la codifichi.
La sequenza scelta, anch'essa caratterizza il particolare codificatore.

4.6.2 Diagrammi di transizione degli stati

Possiamo pensare uno *zerotree coder* come un sistema a stati finiti. Lo stato corrente è la partizione P_k del sistema di partizionamento S utilizzata per rappresentare l'*informazione* di significatività, essendo associato inoltre a ciascun

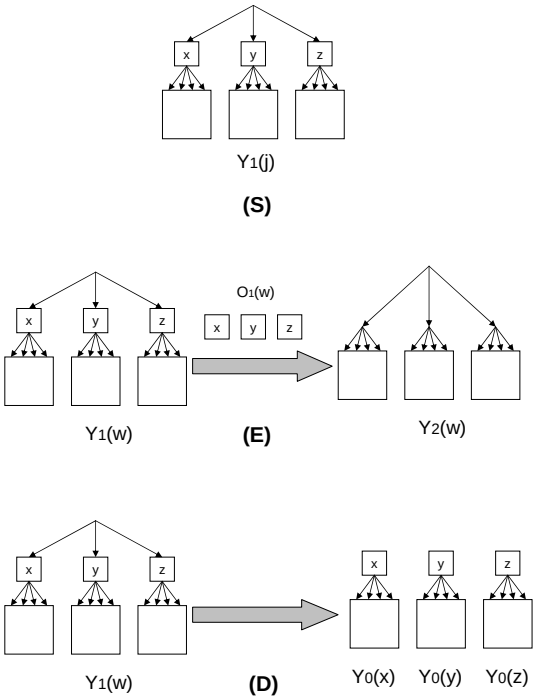


Figura 4.18: Tipi di **azioni elementari**.

insieme di cardinalità unitaria un identicativo che lo contraddistingua come IE oppure SE. L'uscita del sistema è proprio la codifica dello stato.
Una rappresentazione di tipo ridondante dello stato è costituita dall'elenco esaustivo F di tutti gli insiemi ottenibili attraverso il sistema di partizionamento, a ciascuno dei quali è associato un flag binario che censisce o meno l'appartenenza alla partizione P_k (l'insieme puo' essere dunque **attivo** o **inattivo**). I *singoletti* devono essere specificati come IE o SE. Poiché gli insiemi che non sono singoletti sono tutti *zerotree*, essi sono identificati unicamente dalla radice e dall'*ordine*. In definitiva lo stato del codificatore è rappresentabile come l'elenco esaustivo dei coefficienti e degli *zerotree* da essi generati. Ai coefficienti corrisponderà uno stato di tipo **singoletto** il quale potrà evolvere tra le condizioni di inattività (="off"), di "IE" e di "SE". Agli *zerotree* corrisponderà invece uno stato di tipo **albero**, il quale evolverà tra la condizione di "off" e gli *ordini* ammessi di zerotree "Y_n".
Questa rappresentazione è particolarmente utile in quanto permette di descrivere le transizioni dello stato complessivo del sistema, nonostante la sua proibitiva dimensionalità, a partire dalle transizioni dei due tipi di sottostati componenti. Nel diagramma in fig. 4.19 i sottostati sono rappresentati con dei rettangoli. All'interno dei rettangoli è rappresentata l'evoluzione del sottostato tramite un grafo. I nodi del grafo sono i valori che il sottostato puo' assumere. Gli archi orientati tra i nodi descrivono le transizioni: su ogni arco è riportato a sinistra il risultato del test che causa la transizione e a destra la codifica in bit della transizione stessa. Altri archi orientati invece connettono una transizione di uno stato a quella di un'altra quando la prima attiva la seconda.
Specificato il tipo di sottostato è possibile definirne il comportamento in due circostanze:

- durante le transizioni interne e quando esso attiva transizioni in stati differenti (**diagramma di evoluzione**);
- quando una sua transizione è attivata da stati differenti (**diagramma di attivazione**).

Fissato il tipo di stato (*albero* o *singoletto*) e assunto il medesimo comportamento per tutti gli stati dello stesso tipo è semplice ottenere, in maniera iterativa e ricorsiva, a partire descrizione del comportamento dei singoli tipi di sottostato, il comportamento di tutto lo stato del sistema.

In fig. 4.19 il **diagramma di transizione** degli stati è quello dell'algoritmo EZW+, il quale è molto simile ad EZW di Shapiro e costituisce forse il più

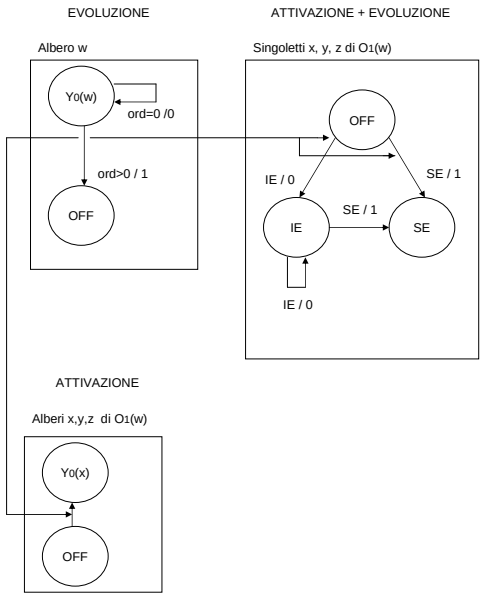


Figura 4.19: Diagramma di transizione degli stati, così come in **EZW+**.

semplice esempio di *zerotree coder*. Come EZW, EZW+ usa solo zerotree di ordine 0. Finché è possibile, si cerca di rappresentare ciascun insieme della partizione come uno *zerotree*. Quando al corrente livello di qualità della rappresentazione l'insieme in questione diventa significativo, non resta che *estrarre* la radice w e *decomporre* il rimanente *zerotree* di ordine 1 in un certo numero di *zerotree* di ordine 0. Le radici dei novelli *zerotree* sono tutti e soli gli elementi di $O_1(w)$. La coppia di *azioni elementari* appena illustrate è codificata con il bit 1, mentre la condizione per cui lo zerotree originario rimane tale è codificata con il bit 0.

4.6.3 Diagrammi di evoluzione degli alberi

Uno zerotree codec puo' essere definito univocamente da:

- la *partizione primitiva* del sistema di partizionamento (**inizializzazione**);
- l'*albero descrittore* della trasformata;

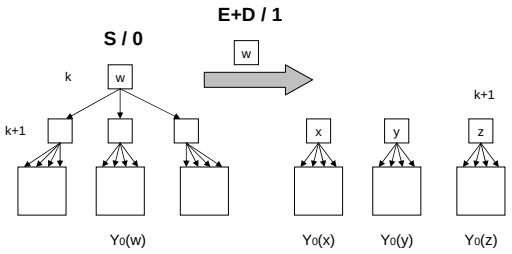


Figura 4.20: Diagramma di evoluzione degli alberi, così come in **EZW+**. Il diagramma è incompleto in quanto non sono riportate le condizioni di test.

- il *diagramma di transizione dei sottostati albero e singoletto*;
- l'*ordine di scansione dei coefficienti* (dato ad esempio dalle liste).

Il *diagramma di transizione* dei sottostati è uno strumento molto preciso ed universalmente comprensibile (rispetta ad esempio le specifica di un diagramma degli stati così come pensato in linguaggi di progettazione ampiamente diffusi, come lo standard UML). Tuttavia ci sembra opportuno introdurre una rappresentazione più intuitiva per l'evoluzione del sottostato *albero*, in quanto la stessa costituisce la parte più delicata del codificatore dal punto di vista algoritmico.
Il diagramma in questione, che denominiamo **diagramma di evoluzione dell'albero**, è molto simile ad un grafo di evoluzione dello stato, è riportato in figura 4.20 per l'algoritmo EZW+. Le differenze sono le seguenti:

- spesso sugli archi orientati non sono riportate le *condizioni di test* (ciò eventualmente rende il diagramma *incompleto*);
- sugli archi orientati sono riportate le *azioni elementari*;
- sugli archi orientati sono riportate le attivazioni dei *singoletti*;
- lo stato di inattività dell'albero è rappresentato con la descrizione dell'attivazioni degli *alberi* da esso attivati.

Si potrà notare come l'azione di *estrazione* attiva stati *singoletto*, mentre quella di *decomposizione* disattiva lo stato *albero* corrente ed attiva altri stati *albero*.

4.6.4 Tabelle di evoluzione degli alberi

Le **tabelle di evoluzione** completano eventualmente i *diagrammi di evoluzione degli alberi* descrivendo esplicitamente le condizioni di test. Il test di significatività effettuato sugli zerotree costituisce l'ingresso della tabella, l'uscita della tabella è costituita dalla codifica in bit della transizione, la sequenza di *azioni elementari* determina univocamente la transizione del sottostato e le attivazioni di altri sottostati. Le *tabelle di evoluzione* specificano senza ambiguità la transizione dello stato se corredate di un diagramma di transizione dei sottostati singoletto.

Ordine(k-1)	**Ordine(k)**	**Transizione**	**Codifica**
$= 0$	$= 0$	P	0
$= 0$	> 0	E+D	1

Tabella 4.1: Tabella di evoluzione di **EZW+**.

4.7 Codifica aritmetica

Uno stadio di *codifica aritmetica* può eventualmente potenziare le funzionalità del *codificatore gerarchico* di uno *zerotree coder*. Di solito tanto maggiore è l'apporto alle prestazioni dello stadio di *codifica aritmetica* quanto meno potente è la rappresentazione derivante dal sistema di partizionamento adottato.
Nelle analisi sperimentali presentate nel seguente capitolo, lo stadio di *codifica aritmetica* sarà costituito da un **banco di codificatori aritmetici adattativi**. Ciascun codificatore del banco sarà preposto alla codifica dei dati distinti per bitplane e sottobanda.
Anche sui i bit di segno sarà effettuata la *codifica aritmetica*, separatamente per bitplane e sottobanda di appartenenza.
Non sempre in letteratura è stato adottato proprio questo tipo di *codifica aritmetica*. In ogni caso i risultati dei più importanti articoli di riferimento non sono molto differenti da quelli presentati in questo lavoro, per cui le considerazioni fatte potranno essere considerate dal lettore sufficientemente generali.

4.8 Configurazioni storiche

Le configurazioni storiche che presentiamo in questo paragrafo sono quasi tutti *zerotree coder* proposti in celebri lavori scientifici del settore. Soltanto l'ultimo algoritmo è uno *zeroset coder* che non appartiente alla famiglia degli *zerotree coder*. Verrà descritto per ultimo per meglio illustrare come, nel caso generale degli *zeroset coder*, altre strutture possano prendere il posto degli *alberi descrittori* per il progetto *iterativo* e/o *ricorsivo* dei *sistemi di partizionamento*. Le versioni dei codificatori che presenteremo saranno quelle per la codifica di immagini a toni di grigio. Tuttavia essi sono stati utilizzati anche per la codifica di altri tipi di dati.

Si evidenzia che nel presente lavoro sono state effettuate le seguente generalizzazioni, che hanno permesso di racchiudere questi algoritmi nelle categorie di *zeroset coder* ed eventualmente di *zerotree coder*:

- *codificatore gerarchico* per la *significatività*;
- implementabilità mediante liste multiple;
- *albero descrittore* (solo per gli *zerotree coder*);
- *zerotree* generalizzato mediante concetto di ordine dello zerotree (solo per gli *zerotree coder*);

A queste va aggiunta la condizione di test generalizzata introdotta nell'algoritmo 3, la quale costituisce un elemento di novità rispetto agli algoritmi presentati nel corrente paragrafo.

4.8.1 EZW

L'algoritmo **EZW** (*Embedded Zerotree Wavelet*) di Shapiro è molto simile alla versione EZW+ già presentata. L'inizializzazione, nel caso di decomposizione wavelet di un immagine, è ottenuta ponendo come radice degli *zerotree* tutti i coefficienti della banda LL. L'albero descrittore è quello di fig. 4.14. EZW differisce da EZW+ per la codifica delle transizioni degli stati. La *tabella di evoluzione degli alberi* è la 5.10. Anche l'*evoluzione dei singoletti* è codificata differentemente. I SE sono codificati con la sequenza di bit 01, mentre gli IE con il bit 1.
Poiché ogni volta che avviene una *riduzione* necessariamente deve essere codificato un IE, oppure un SE (seguito dal suo segno) Shapiro individua in [4] quattro casistiche di codifica:

- **zerotree**, codifica 00;
- **isolated zero**, codifica 01;
- **significant positive**, codifica $1+$;
- **significant negative**, codifica $1-$.

Questo tipo di codifica è meno efficiente di quella di EZW+ in quanto l'algoritmo di Shapiro non prescinde da uno stadio successivo di codifica entropica.
Per questo motivo, per un più trasparente confronto tra la potenza descrittiva di sistemi di partizionamento diversi, utilizzeremo, in seguito, la più razionale codifica di EZW+.
E' da notare inoltre che l'algoritmo originale di Shapiro non fa uso delle liste ed usa un ordinamento totale per quel che riguarda l'esplorazione dei coefficienti.

Ordine(k-1)	Ordine(k)	Transizione	Codifica
$= 0$	$= 0$	S	00
$= 0$	> 0	E+D	$-$

Tabella 4.2: Tabella di evoluzione di **EZW**.

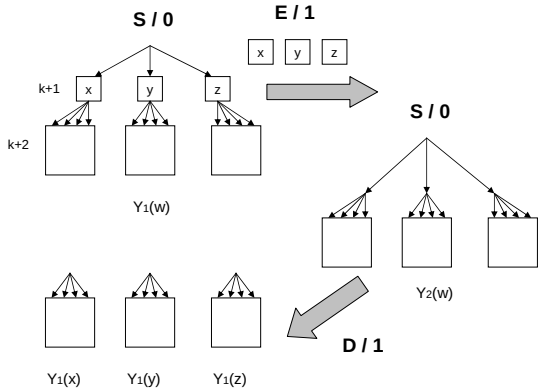


Figura 4.21: Partizionamento con alberi di ordine 1 e 2, così come in **SPIHT**.

4.8.2 SPIHT.

SPIHT (*Set Partitioning In Hierarchical Trees*) di Said e Pearlman [5], a differenza di EZW, non impiega zerotree di ordine 0, ma zerotree di ordine 1 e 2. La tabella di evoluzione di SPIHT è la 4.3 mentre il diagramma di evoluzione degli alberi è riportato in figura 4.21.

Ordine(k-1)	Ordine(k)	Transizione	Codifica
$= 1$	$= 1$	S	0
$= 1$	> 1	E	1
$= 2$	$= 2$	S	0
$= 2$	> 2	D	1

Tabella 4.3: Tabella di evoluzione di **SPIHT**.

L'*albero descrittore* dell'articolo originale è diverso da quello della fig. 4.14. La differenza tra i due alberi verrà descritta nel capitolo successivo.

4.8.3 PROGRES

In [14] SPIHT viene arrangiato per la *codifica lossless* in modo da evitare l'onerosa codifica per bitplane. L'algoritmo risultante è stato denominato **PROGRES** (Progressive Resolution Coding)e qui viene riproposto in maniera alternativa e con qualche piccola variante rispetto al lavoro originale. Come in [14] l'algoritmo viene suddiviso in due passi. Nel primo si codifica unicamente la *significatività*. Nel secondo invece si codificano il *segno* e tutti i bit di *raffinamento*. L'ordinamento è totale sia tra gli *zeroset*, nel primo passo, che tra i coefficienti nel secondo. In questo modo viene minimizzato il numero di volte in cui lo stesso coefficiente viene rivisitato.

L'implementazione con liste multiple in questo caso non è certo la più efficiente. Tuttavia proviamo ad immaginare proprio questo tipo di implementazione in modo da capire in che ordine devono essere visitati gli *zeroset.* Supponiamo che sia associata una lista a ciascuno degli *zeroset* (essa sarà una LIE o una LIS a seconda che lo *zeroset* sia un IE oppure un IS). Poiché l'ordinamento è totale siamo nel caso limite in cui ciascuna lista possiede al più un unico elemento, per cui potremmo demplicemente che la lista o è vuota o contiene lo *zeroset*.
Sia $R_k(x)$ la relazione che lega un coefficiente $x \in O_k(r)$ ad r. Un ordinamento parziale che vincola l'ordinamento totale scelto e si basa sulle le seguenti regole:

- $R_{k+1}(x)$ precede $R_k(x)$;
- $Y_h(R_{k+1}(x))$ precede $Y_h(R_k(x))$;
- $Y_h(x)$ precede $Y_{h+1}(x)$.

Per maggior chiarezza lo schema di codifica è riportato nell'algoritmo 6. Stabilito l'ordinamento parziale definito in precedenza, lo schema di PROGRES può essere visto anche come un generale adattamento efficiente di *zerotree coder* alla *codifica lossless*. Nello pseudocodice la lista LSE è stata eliminata, mentre durante la scansione di LIE e LIS, come illustrato negli algoritmi 7 e 8, il primo *bitplane di significatività* è aggiornato nella matrice SB.

Algoritmo 6 Zeroset coder: implementazione lossless.

procedure LOSSLESSZEROSETCODER(X)

for each $I \in P_{min}(X)$ **do**
Inizializza($I, LIS[.], LIE[.], SB$);
for each I_k**generatedby**I **do**
for each $bitplane$ **do**
EsploraLIE$*$($I, LIE[.], SB$);
EsploraLIS$*$($I, LIE[.], LIS[.], SB$);
end for
end for
end for

for each $x \in X$ **do**
for each $bitplane > SB(x)$ **do**
code Segno(x);
code Raffinamento(x);
end for
end for

end procedure

Algoritmo 7 Scansione della LIE nel caso lossless.

procedure ESPLORALIE*****(x,$LIE[.]$,SB)
if then Presente(x)
codifica Signif(x);
if Signif(I) $= 0$ **then**
$SB(x) := SB(x) + 1$;
else
Elimina($I, LIE[.]$);
end if
end if
end procedure

Algoritmo 8 Scansione della LIS nel caso lossless.

```
procedure EsploraLIS*(I,LIE[.],LIS[.],SB)
    if thenPresente(I)
        t := Test(I)
        CodificaTest(t);
        if Signif(I) = 1 then
            {I_1, ..., I_N} := GeneraPartizione(I, t);
            for each k ∈ {1, ..., N} do
                if Dim(I_k) = 1 then
                    CodificaSignif(I_k);
                    if Signif(I_k) = 0 then
                        Accoda(LIE[I_k], I_k);
                        SB(I_k) = SB(I_k) + 1;
                    end if
                else !caso in cui Dim(I_k) > 1
                    Accoda(LIS[I_k], I_k);
                end if
            end for
            Elimina(I, LIS[I]);
        end if
    end if
end procedure
```

4.8.4 SPECK

SPECK (*Set Partitionred Embedded bloCK coder*)è un esempio di *zeroset coder* che non è uno *zerotree coder*. Il suo sistema di partizionamento è ottenuto mediante la tecnica ricorsiva. Supponiamo di voler codificare i coefficienti di una wavelet diadica bidimensionale. La *partizione primitiva* è costituita da due soli insiemi: tutti i coefficienti della banda passa basso LL, I_L, e tutti i coefficienti delle sottobande superiori, I_H. Vi sono due tipi di IS ammessi: quelli costituiti da coefficienti di una sola sottobanda (*Single Sub-Band* sets, **SSB**) e quelli composti da coefficienti di sottobande differenti, (*Many Sub-Band* sets, **MSB**). I_L è classificato come SSB, mentre I_H come MSB. Due regole ricorsive differenti stabiliscono la decomposizione degli *zeroset*:

- gli SSB si partizionano in quattro parti uguali, secondo la regola del *quadtree*;
- gli MSB si decompongono in tre SSB (le tre sottobande wavelet di ordine inferiore) ed un solo MSB (tutte le altre sottobande).

Il tipo di decomposizione è mostrato in fig. 4.22.

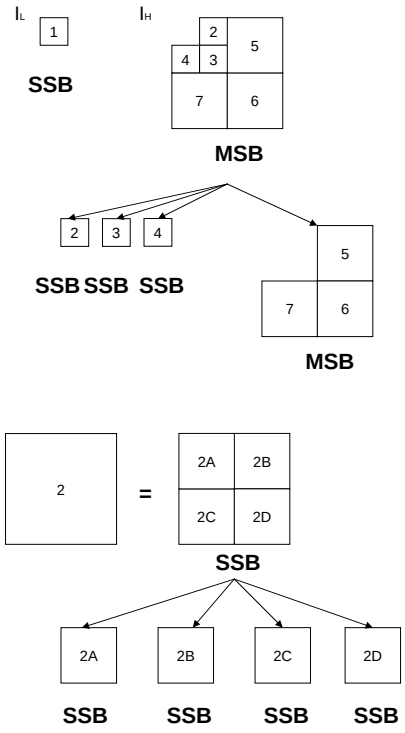


Figura 4.22: Regola di partizionamento di **SPECK**.

4.9 Conclusioni

In questo capitolo, facendo uno sforzo di sintesi dei vari lavori proposti in letteratura, si è cercato di formalizzare i concetti di *zeroset codec* e di *zerotree codec*.
A differenza di quanto già fatto da altri autori:

- si è formalizzato un modello di descrizione dei *codificatori gerarchici*;
- si è formalizzato il concetto di *zeroset coder*;
- è stato reso esplicito il legame tra *partizionamento gerarchico* ed *albero descrittore della trasformata*, per gli *zerotree coder*;
- si sono forniti degli strumenti formali per la descrizione degli *zerotree coder* come sistemi a stati finiti;
- è stato ampliato il numero di possibili transizioni di stato, grazie alla generalizzazione della *funzione di test*;
- le definizioni date sono state rese indipendenti dalla dimensionalità dei dati e dalla natura della trasformata.

Lo studio presentato nel capitolo ha come risultato una descrizione di tipo comportamentale dei codificatori. Non sono state però affrontate le due seguenti fondamentali questioni:

- come si misura l'efficienza dei codificatori;
- come si progetta un codificatore per essere efficiente.

Bisogna dunque ancora descrivere le proprietà dei codificatori dal punto di vista informazionale, avendone fornito finora soltanto una descrizione di tipo algoritmico. L'analisi informazionale degli *zeroset coder* tuttavia non è mai stata affrontata in letteratura in maniera completa e soddisfacente ed è ben più complessa di quella algoritmica. Solo in [12] vi è un timido tentativo di affrontare la questione. Sicuramente un'analisi rigorosa degli algoritmi, così come presentata in questo capitolo, offre maggiori possibilità di studio anche dal punto di vista della teoria dell'*informazione*.
Nel prossimo capitolo tenteremo di fornire alcuni spunti su queste tematiche, essendo comunque ben lungi da una formalizzazione vera e propria. L'attenzione verrà rivolta unicamente agli *zerotree coder*. Inevitabilmente lo studio, affinché sia possibile un riscontro sperimentale, sarà vincolato alla natura

statistica dei dati e alla trasformata. Per questo motivo vedremo anche, con maggior dettaglio, come costruire in maniera efficace gli *alberi descrittori*.

Bibliografia

[1] L. Cicala, G. Poggi, “A generalization of zerotree coding algorithms”, Picture Coding Symposium , Lisbona (Portogallo), n. 1050, novembre 2007.

[2] J.M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients”, *IEEE Trans. On Signal Processing*, vol. 41, n.12 pp. 3445-3463, dicembre 1993.

[3] A. Said, W. A. Pearlman: “New fast and efficient image codec based on set partioning in hierarchical tree”, *IEEE Trans. On Circuits and Systems on Video Technology*, vol. 6, pp. 243-250, giugno 1996.

[4] A. Islam and W. A. Pearlman, “An Embedded and Efficient Low-Complexity Hierarchical Image Coder”, *Visual Communications and Image Processing ’99, Proceedings of SPIE*, Vol. 3653, pp. 294-305, gennaio 1999.

[5] F. W. Wheeler, W. A. Pearlman: “SPIHT image compression without lists” , *IEEE Int. Conf. on Acoustics, Speech and Signal Processing* (ICASSP 2000), Istanbul, Turkey, giugno, 2000.

[6] B-J. Kim, Z. Xiong, and W. A. Pearlman, “Low Bit-Rate Scalable Video Coding with 3D Set Partitioning in Hierarchical Trees (3D SPIHT)”, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 10, pp. 1374-1387, dicembre 2000.

[7] P.L. Dragotti, G. Poggi, A.R.P. Ragozini: “Compression of miltispectral images by three-dimensional SPIHT algorithm”, *IEEE Trans. On Geoscience and Remote Sensing*, vol. 38, gennaio 2000.

[8] D.Taubman, “High performance scalable image compression with EBCOT”, *IEEE Trans. On Image Processing*, Volume 9, Issue 7, pp.1158 - 1170, luglio 2000.

[9] C. He, J. Dong, Y. F. Zheng. Z. Gao, “Optimal 3D coefficient tree structure for 3D wavelet video coding” - *IEEE Trans. On Circuits and Systems on Video Technology*, vol. 13, ottobre 2003.

[10] H. Danyali and A. Mertins, “A fully SNR, spatial and temporal scalable 3DSPIHT-based video coding algorithm for video streaming over heterogeneous networks ” , *Proc. TENCON*, Bangalore, India, vol. 4, pp. 1445-1449, ottobre 2003.

[11] H. Danyali, A. Mertins: “Flexible, highly scalable, object-based wavelet image compression algorithm for network applications” , *Vision, Image and Signal Processing, IEE Proceedings*, vol. 151, dicembre 2004.

[12] Y. Cho and W. A. Pearlman, “Quantifying the Performance of Zerotrees of Wavelet Coefficients: Degree-k Zerotree Model” , *IEEE Trans. on Signal Processing*, Vol. 55, Part 1, pp. 2425-2431, giugno 2007.

[13] E. Christophe and W. A. Pearlman, “Three-dimensional SPIHT Coding of Volume Images with Random Access and Resolution Scalability”, submitted to *Eurasip J. on Image and Video Processing*, settembre 2007.

[14] Y. Cho, W. A. Pearlman, and A. Said, “Hierarchical Dynamic Range Coding of Wavelet Subbands for Fast and Efficient Image Compression”, *IEEE Trans. on Image Processing*, Vol. 16, No. 8, pp. 2005-2015, agosto 2007.

Capitolo 5

Progetto di zerotree coder multidimensionali

Parole chiave

Nozioni propedeutiche

Codifica mediante trasformata, analisi multirisoluzione, DWT, KLT, stima e compensazione del movimento, campi di moto diretto e inverso, albero descrittore, zerotree, ordine di zerotree, zerotree coder, azioni elementari di codifica, diagrammi di evoluzione degli alberi, tabelle di evoluzione degli alberi, EZW, SPIHT.

Concetti introdotti nel capitolo

Decomposizione di Dragotti-Poggi-Ragozini, albero descrittore di Shapiro, albero descrittore di Said e Pearlman, albero descrittore di Dragotti-Poggi-Ragozini, albero descrittore diadico, albero descrittore packet.

Concetti innovativi illustrati nel capitolo

Leggi di evoluzione Ord0, Ord01, Ord012.

5.1 Introduzione

In questo capitolo ci occupiamo di analizzare sperimentalmente alcune scelte progettuali sugli *zerotree coder*. In particolare focalizziamo l'attenzione su due aspetti principali:

- la scelta dell'*albero descrittore*;
- il massimo *ordine* previsto per gli *zerotre*e.

La scelta dell'albero descrittore è stata già discussa in [9] per quel che riguarda le sequenze video. Noi estenderemo lo studio anche al caso delle immagini multispettrali. Inoltre approfondiremo il discorso su un'altro tipo di analisi multirisoluzione, quella per immagini multicanale proposta in [7], che è una combinazione di DWT bidimensionale nel piano trasverso e KLT lungo la dimensione spettrale.
La sapiente scelta degli *ordini di zerotree* effettuata in [5], ha permesso a SPIHT di diventare un importante *benchmark* per gli *zerotree coder* e più in generale per tutti i codificatori basati su analisi multirisoluzione. Per quanto in [12] il concetto di *ordine di zerotree* sia stato formalizzato, nessuna prova convincente finora è stata fornita sulla particolare scelta effettuata da SPIHT. L'unico vero confronto tra SPIHT e *zerotree coder* alternativi è quello con EZW [4]. Nessun'altra combinazione di *ordini di zerotree* diversa da quelle proposte in [4] (ordine 0) e in [5] (ordini 1 e 2) è stata proposta e valutata sperimentalmente. In questo lavoro di tesi si prenderanno in esame combinazioni di ordini 0 ed 1, e di ordini 0, 1 e 2. Si cerchera di capire dunque qual'è il reale vantaggio di considerare l'ordine 0 oltre all'ordine 1, e poi cosa cambia aggiungendo l'ordine 2.

5.2 Analisi multirisoluzione tridimensionale

Gli *zerotree coder* sono particolarmente indicati per codificare le *analisi multirisoluzione*. L'*albero descrittore* infatti è costruito in maniera tale da mettere in relazione coefficienti in posizioni spaziali omologhe e scale differenti; si cerca così di trarre giovamento dalla correlazione tra sottobande di scale diverse. Questa strategia differisce dalle tecniche che invece cercano di trarre vantaggio dalla correlazione tra coefficienti spazialmente contigui all'interno della stessa sottobanda, come ad esempio quella impiegata da EBCOT [8].
In questo paragrafo verranno descritte le analisi multirisoluzione impiegate nelle sperimentazioni i cui risultati sono riportati nel corso dell'intero capitolo. Nel caso bidimensionale è stata effettuata semplicemente una DWT-2D diadica. Nel caso tridimensionale sono state prese in esame invece due diverse trasformate multidimensionali. La prima è una DWT tridimensionale *packet*: prima è effettuato un filtraggio diadico lungo la terza dimensione, poi ciascun immagine trasformata è analizzata con una DWT-2D. La seconda è una KLT lungo la terza dimensione seguita da una DWT bidimensionale sulle immagini trasformate [7]. D'ora in avanti indicheremo sinteticamente queste due decomposizioni come DWT-3D *packet* e **DPR** (Dragotti-Poggi-Ragozini, gli autori di [7]).
I filtri wavelet utilizzati negli esperimenti sono quelli biortogonali e dispersivi *Daubachies 9-7* nelle due dimensioni spaziali e quello ortogonale e non dispersivo di *Haar* lungo la terza dimensione [1] .

5.2.1 DWT-3D e compensazione del movimento

Negli esperimenti su sequenze video abbiamo adottato lo schema proposto in [14] basato sul *lifting scheme compensati in movimento* (vedi fig. 5.1). Lo schema di analisi temporale è diadico ed il filtro è quello di Haar.
Il modello del moto è di tipo *traslazionale a blocchi*, come nello standard MPEG2. I blocchi e le *finestre di ricerca* sono quadrati e di dimensione identica per tutte le coordinate spaziali. La stima del movimento è effettuata con un algoritmo di ricerca esaustiva. Il campo *diretto* e quello *inverso* sono calcolati separatamente. Nel tracciamento delle *curve tasso-distorsione* non è incluso il costo della codifica dei *campi di moto*, i risultati in termine di PSNR sono dunque sovrastimati.
Le sequenze video selezionate per gli esperimenti con compensazione del movimento sono caratterizzate da un campo di moto traslazionale; il modello di stima e compensazione del movimento è dunque del tutto adeguato a

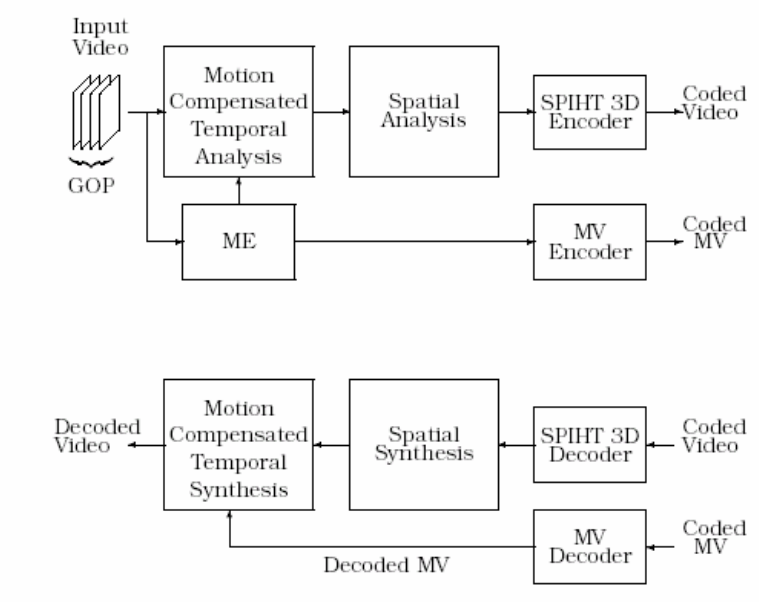


Figura 5.1: Schema di codifica per le sequenze video.

rappresentare gli spostamenti degli oggetti nel piano immagine e vi sono le condizioni perchè lo schema proposto in [14] sia davvero efficace.

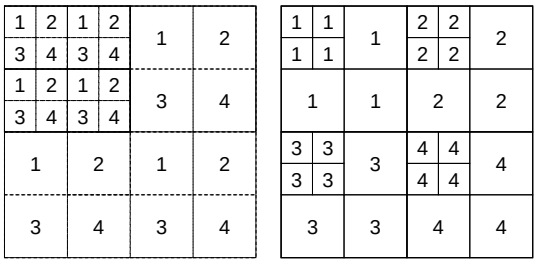


Figura 5.2: Decomposizione diadica bidimensionale. (A) Organizzazione per sottobande. (B) Organizzazione per **unità elementari di analisi**.

5.3 Progetto dell'albero descrittore

Sono state proposte fino ad ora diverse strategie di progetto dell'*albero descrittore*. In realtà le soluzioni di volta in volta presentate sono fortemente legate al tipo di analisi multirisoluzione, per cui non è semplice trarre una generalizzazione.
Con l'intento di avere ulteriori strumenti di studio del problema, definiamo informalmente come **unità elementare dell'analisi** l'insieme più piccolo di coefficienti trasformati che conserva le seguenti proprietà dell' analisi multirisoluzione: numero di sottobande, rapporti di scala tra le sottobande, numero e tipo dei filtraggi eseguiti su ciascuna delle sottobande. Inoltre i coefficienti appartenenti ad un'*unità elementare di analisi*, una volta antitrasformati, devono mantenere la localizzazione spaziale più stretta possibile, nel senso che l'energia deve essere concentrata in pochi coefficienti contigui. Per una decomposizione wavelet diadica bidimensionale mostriamo in fig. 5.2, sul lato sinistro (A), una classica organizzazione per sottobande, e, a destra (B), la disposizione dei coefficienti trasformati per *unità elementari di analisi*.

5.3.1 Alberi descrittori nel caso bidimensionale

Per quel che riguarda la *decomposizione diadica* bidimensionale, in [4], si propone di mettere in corrispondenza (attraverso relazioni genitore-figlio) i *pixel* appartenenti ad una generica sottobanda con quelli aventi la stessa localizzazione spaziale nella sottobanda di scala immediatamente inferiore. Un

Tasso	S-2D	P-2D
0.0625	26.30	25.91
0.125	28.27	27.89
0.25	30.30	30.10
0.5	32.59	32.68
1	35.90	36.00

Tabella 5.1: Confronto per l'immagine monocromatica Goldhill tra gli *alberi descrittori* S-2D e P-2D. La decomposizione effettuata è a 6 livelli nel primo caso e a 5 nel secondo. La figura di merito di riferimento è il PSNR.

ulteriore vincolo è che sono messe in relazione esclusivamente le sottobande ricavate dallo stesso numero di filtraggi passa-alto in una determinata direzione, in modo tale da privilegiare il fattore **direzionalità** (dettagli orizzontali, verticali e diagonali). Non capita mai che coefficienti corrispondenti a *direzionalità* diverse siano tra loro accomunati dallo stesso genitore, ad eccezione del primo livello di decomposizione, in cui la banda di trend genera le tre bande di dettaglio alla stessa scala, le quali hanno tre *direzionalità* ben distinte. Definiamo l'albero descrittore così costruito **albero di Shapiro**, indicandolo in seguito sinteticamente con la sigla **S-2D**. La sua struttura è riportata in fig. 5.3.
In [5] (fig. 5.4) viene invece proposto un *albero descrittore* leggermente differente. Questa volta si cercano di evitare del tutto raggruppamenti di coefficienti di *direzionalità* diversa. Per mantenere comunque alberi abbastanza grandi si combinano tra loro coefficienti appartenenti a quattro *unità elementari di analisi* contigue. In seguito ci riferiremo a questa struttura chiamandola **albero di Said e Pearlman** e denotandola con la sigla **P-2D**.
Poichè nelle applicazioni l'**unità elementare di codifica** è quella costituita dai coefficienti afferenti allo stesso *albero descrittore*, l'*albero di Said e Pearlman* presenta rispetto a quello di *Shapiro*, lo svantaggio di aver bisogno del quadruplo delle *unità elementari di analisi*. Per questo motivo, a parità di dimensioni dell'**unità elementare di codifica**, intesa come il più piccolo insieme di coefficienti che non può essere codificato separatmente, adottando la strategia di *Shapiro* è possibile eseguire un livello di decomposizione in più.
Il confronto prestazionale tra le due soluzioni, a valle di una DWT-2D diadica, è mostrato in tab. 5.1.

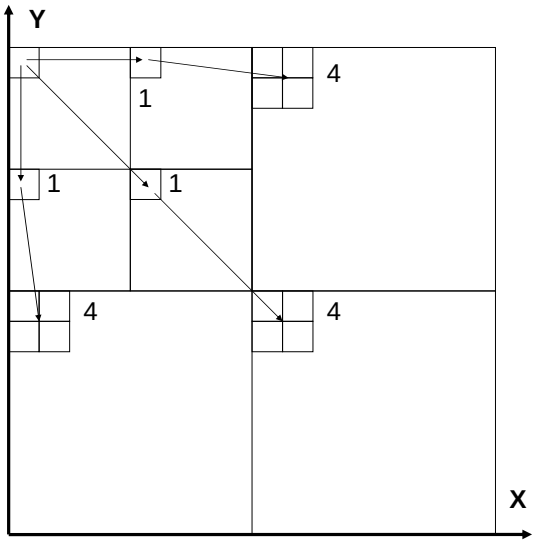


Figura 5.3: Albero descrittore di Shapiro (**S-2D**).

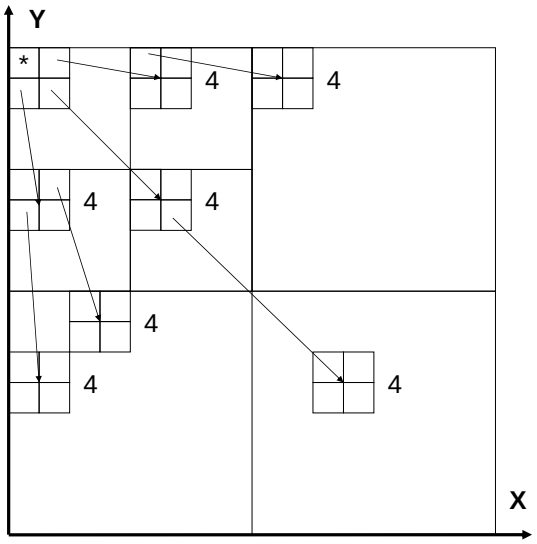


Figura 5.4: Albero descrittore di Said e Pearlman (**P-2D**).

5.3.2 Alberi descrittori nel caso 3D

Il più semplice sistema di alberi descrittori in tre dimensioni è costituito da un albero descrittore bidimensionale per ciascuna immagine trasformata. Questa struttura, per quanto banale, è molto interessante in quanto riduce di molto le dimensioni dell'*unità elementare di codifica* che diventa addirittura più piccola dell'*unità elementare di analisi*. Il vantaggio più grande di questa configurazione è il non dover accedere a più immagini trasformate contemporaneamente durante la codifica. La sperimentazione riportata in questo capitolo propone una struttura costituita da un *albero di Shapiro bidimensionale* riproposto per ciascuna immagine trasformata.
Quando si è provato ad adattare gli *zerotree coder* ad un'analisi tridimensionale, tuttavia si è subito pensato a creare strutture più complesse, anche esse tridimensionali.
Per le decomposizioni tridimensionali DWT-3D *packet*, la prima soluzione storicamente proposta è quella presentata in [6]. Tale soluzione non è progettata *ad hoc* per la DWT *packet*, piuttosto si adatta al caso *packet* un albero pensato per la decomposizione diadica tridimensionale. L'estensione dell'*albero di Said e Pearlman* ad un numero arbitrario di dimensioni è infatti immediata nel caso di decomposizione diadica. In fig. 5.5 viene mostrata la relazione di parentela dei coefficienti in un piano ortogonale alla terza dimensione (indicata con Z) e ad una delle due dimensioni spaziali (indicata con Y). Lo stesso albero di fig. 5.5 viene adattato all'analisi *packet* in [6] così come mostrato in fig. 5.6. Un *albero descrittore* costruito in questo modo è detto **albero simmetrico**. In seguito ci riferiremo a questa struttura con la sigla (**P-3D-s**), che sta per *albero di Said e Pearlman tridimensionale simmetrico*.
Allo stesso modo, come proposto in [9], è possibile impiegare per la decomposizione *packet* un *albero simmetrico* tridimensionale ottenendolo direttamente dalla strategia di Shapiro in due dimensioni, così come mostrato nello schema di fig. 5.7. Questo *albero descrittore* verrà d'ora in avanti denotato con la sigla (**S-3D-s**), che sintetizza la dicitura *albero di Shapiro tridimensionale simmetrico*.

Gli *alberi descrittori simmetrici* presentano tuttavia lo svantaggio di disporre sullo stesso livello coefficienti appartenenti a sottobande di ordine differente. Addirittura possiamo avere coefficienti di sottobande diverse generate direttamente dallo stesso nodo. Poichè i coefficienti di sottobande di ordine diverso hanno statistiche presumibilmente differenti, questo tipo

di soluzione potrebbe essere inefficiente. Sicuramente è possibile progettare un'*albero descrittore* più orientato alla natura della decomposizione. In [7] per la prima volta si propone un **albero asimmetrico** adattato alla particolare decomposizione non diadica (nello specifico la DPR). Per la DWT-3D *packet* un progetto analogo è proposto in [9], in cui si estende lo schema di Shapiro anche alla dimensione spettrale, ma questa volta per i soli coefficienti delle sottobande di *trend* di ciascuna immagine trasformata. In [9] si riportano esperimenti in cui, per la codifica video senza compensazione del movimento, *alberi asimmetrici* di questo tipo risultano più efficaci di *alberi simmetrici* come quello di fig. 5.6 e di 5.7. L'albero adottato negli esperimenti presentati nel capitolo è quello di fig. 5.8 (sigla **S-3D-a**, *albero di Shapiro tridimensionale asimmetrico*).
Abbiamo già accennato che per l'analisi DPR in [7] si propone un particolare *albero descrittore asimmetrico* adattato alla decomposizione. Esso è schematizzato in fig. 5.9 (**DPR**). La linea di principio è la stessa ripresa in [9]: solo i coefficienti della sottobanda passa-basso spaziale generano nella terza dimensione. Questa volta il sistema di generazione è 1:1 nella dimensione Z, essendo, lungo questa dimensione, l'analisi *monorisoluzione*.

Nei confronti sperimentali che saranno presentati nei successivi paragrafi si fisserà la dimensione dell'*unità elementare di codifica* ed il numero di immagini allocate congiuntamente dallo *zerotree coder* (**unità minima di allocazione**). Per quanto detto, le dimensioni dell'*unità minima di analisi* possono anche cambiare al variare dell'*albero descrittore*.
Detta $X/Y/Z$ la generica decomposizione (X livelli lungo le righe, Y lungo le colonne, Z nella terza dimensione), a ciascun albero descrittore saranno associate le seguenti decomposizioni della DWT-3D:

S-2D : $X/X/Z$;

S-3D-a : $X/X/Z$;

S-3D-s : $X/X/Z$;

P-3D-s : $X-1/X-1/Z-1$.

Nel caso di decomposizione DPR la cifra Z indicherà invece il numero di autobande prese in considerazione dalla KLT.
Con la dicitura **GOP** (*Group Of Picture*), si specificherà il numero di*unità elementari di allocazione* codificate nell'esperimento.

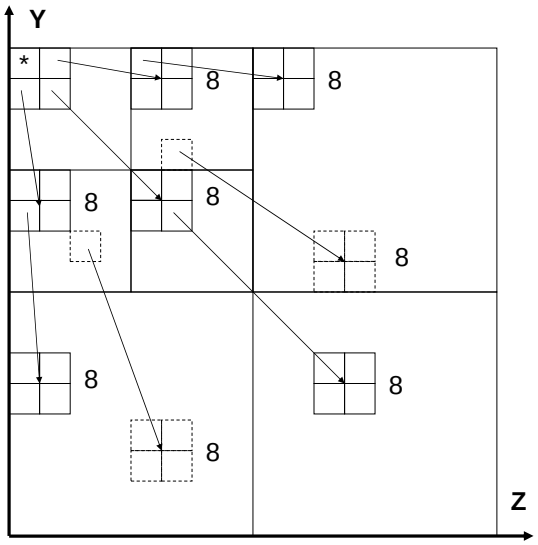


Figura 5.5: Albero descrittore di Said e Pearlman tridimensionale simmetrico.

Per tutti gli esperimenti sugli alberi descrittori la *legge di evoluzione degli alberi* è quella proposta in [5], la quale prevede l'impiego di *zerotree* di ordine 1 e 2 ed è già stata descritta nel capitolo precedente.

5.3.3 Codificatori video

Vediamo dunque quali sono le prestazioni dei diversi *alberi descrittori* introdotti, nel caso della codifica di sequenze video. L'analisi effettuata è di tipo DWT-3D *packet*. I confronti proposti in questo paragrafo sono effettuati senza compensazione del movimento e soltanto sul primo GOP delle sequenze video considerate. I GOP sono costituiti da 32 frame ciascuno, per cui l'analisi per l'albero descrittore S-2D è di $5/5/5$ livelli. La misura di qualità utilizzata è il PSNR ed è riportata in dB.

I video presi in esame sono a risoluzione CIF (288×352) ma si prende in considerazione la sola componente di luminanza. Le sequenze video sono quattro e si possono suddividere in due gruppi, in base alle loro peculiarità statistiche. Un gruppo è costituito dalle due sequenze **Mobile** e **Coastguard**. Esse sono caratterizzate da moto su tutta l'immagine, di tipo traslazionale. Un secondo gruppo invece dalle sequenze **Foreman** e **Akiyo**. In queste sequenze

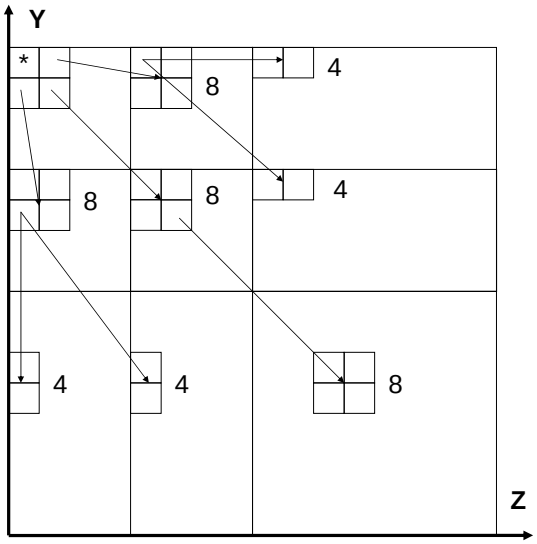


Figura 5.6: Albero descrittore di Said e Pearlman tridimensionale simmetrico, adattato alla decomposizione *packet* (**P-3D-s**).

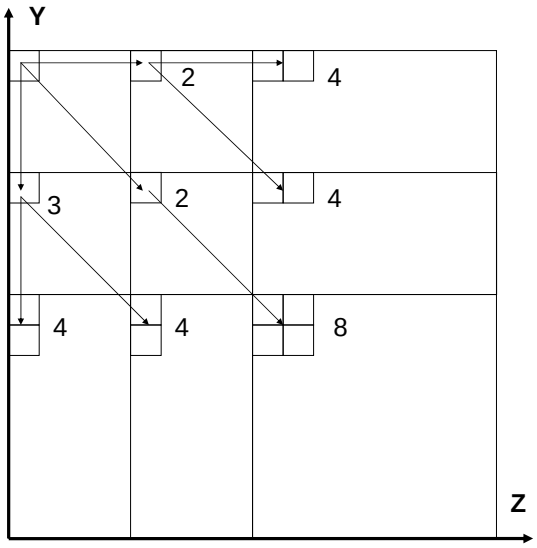


Figura 5.7: Albero descrittore di Shapiro tridimensionale simmetrico, adattato alla decomposizione *packet* **S-3D-s**.

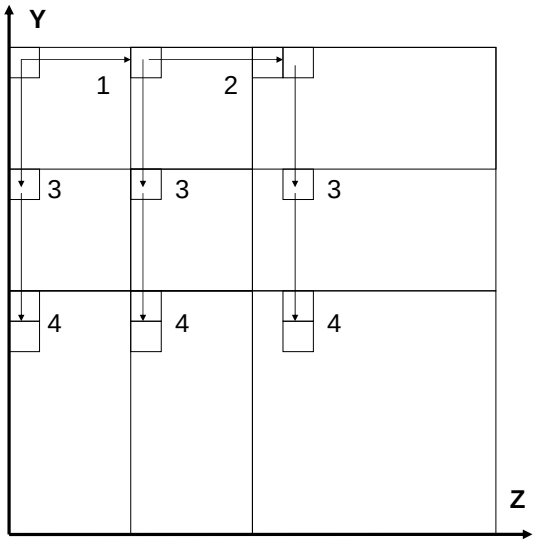


Figura 5.8: Albero descrittore di Shapiro tridimensionale asimmetrico **S-3D-a**.

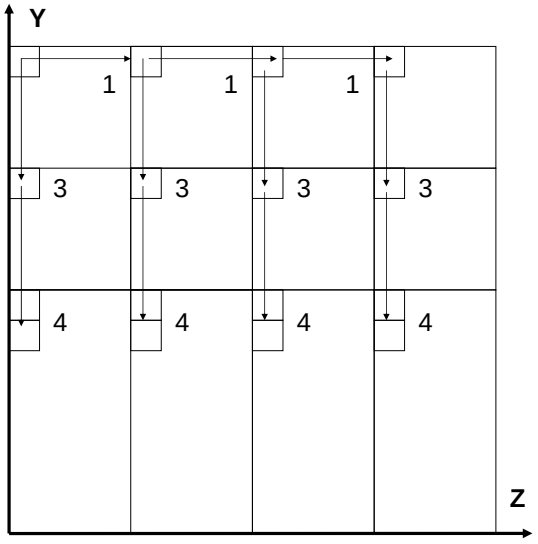


Figura 5.9: Albero descrittore asimmetrico per analisi **DPR**.

Tasso	P-3D-s	S-3D-s	S-3D-a	S-2D
0.0625	19.51	19.49	19.29	18.95
0.125	20.73	20.69	20.57	20.18
0.25	22.32	22.28	22.06	21.89
0.5	24.97	24.90	24.69	24.61
1	28.80	28.70	28.72	28.62

Tabella 5.2: Confronto tra *alberi descrittori* per la sequenza video Mobile.

Tasso	P-3D-s	S-3D-s	S-3D-a	S-2D
0.0625	26.67	26.71	26.74	26.18
0.125	28.16	28.17	28.11	27.89
0.25	29.83	29.87	29.88	29.64
0.5	32.26	32.26	32.12	31.95
1	35.75	35.74	35.60	35.54

Tabella 5.3: Confronto tra *alberi descrittori* per la sequenza video Coastguard.

vi è una persona in primo piano che parla. La scena sullo sfondo è immobile nel caso di Akiyo e soggetta ad un leggerissimo movimento di zoom nel caso di Foreman.

Per il primo gruppo di sequenze video (tab.5.2 e tab.5.3), quelle più mosse, i risultati sperimentali sono leggermente a favore dello schema P-3D-s. Seguono nell'ordine gli *alberi descrittori* S-3D-s, S-3D-a ed S-2D.
Per l'altro gruppo di sequenze video (tab.5.4 e tab.5.5), quelle più statiche, invece le prestazioni sono nettamente a favore dello schema S-3D-a. L'*albero descrittore* S-2D va comunque meglio di S-3D-s e di P-3D-s. P-3D-s è senza dubbio l'*albero descrittore* con il comportamento peggiore.
Si può concludere affermando che gli *alberi descrittori* più robusti rispetto al tipo di sequenza video sono S-3D-a ed S-2D. Nel caso di scarso movimento, le prestazioni degli stessi sono sensibilmente migliori rispetto agli alberi *simmetrici* P-3D-s e S-3D-s. L'albero S-3D-a va meglio di quello S-2D soprattutto a bassi *bit-rate*.

Questi risultati sperimentali presentano un valore aggiunto rispetto a quelli proposti in [9] in quanto le sequenze video analizzate hanno un comportamento statistico più vario e per la prima volta è sviluppato il paragone anche con un *albero descrittore* bidimensionale.

Tasso	P-3D-s	S-3D-s	S-3D-a	S-2D
0.0625	28.97	28.83	29.43	28.62
0.125	31.18	31.07	31.50	31.07
0.25	33.82	33.64	34.04	33.83
0.5	36.79	36.61	37.19	37.08
1	40.73	40.56	41.26	41.19

Tabella 5.4: Confronto tra *alberi descrittori* per la sequenza video Foreman.

Tasso	P-3D-s	S-3D-s	S-3D-a	S-2D
0.0625	38.23	39.57	40.47	38.41
0.125	41.89	43.03	43.95	43.07
0.25	45.83	47.01	48.14	47.62
0.5	50.32	51.84	52.97	52.65
1	55.88	57.49	58.94	58.62

Tabella 5.5: Confronto tra *alberi descrittori* per la sequenza video Akiyo.

5.3.4 Codificatori multicanale basati su analisi DWT-3D

In questo paragrafo sono riportati i risultati relativi alla codifica di immagini multicanale, in particolare immagini multispettrali ed iperspettrali telerilevate.

Negli esperimenti proposti una decomposizione DWT-3D *packet* è stata effettuata sia su un'immagine multispettrale a 4 bande, acquisita mediante il sensore satellitare **IKONOS**, [1]) di dimensioni 448×448 *pixel*, sia su un GOP di 16 bande di un'immagine iperspettrale acquisita dal sensore aerotrasportato **GER** [2]), di dimensioni 256×256 . Le bande spettrali sono state normalizzate a varianza unitaria prima di essere codificate, per pesare allo stesso modo l'informazione di ciascuna banda. Le decomposizioni DWT sono rispettivamente $6/6/2$ e $6/6/4$.

I risultati per questi dati sono decisamente a favore degli alberi S-3D-a e S-2D. A differenza che per le sequenze video, anche a bassi *bit-rate* l'albero S-2D è competitivo rispetto all'albero S-3D-a. Gli alberi simmetrici P-3D-s e S-3D-s risultano molto meno efficaci soprattutto quando le bande spettrali sono poche (vedi tab. 5.6).

[1] Sono considerate per semplicità solo le 4 bande multispettrali con la stessa risoluzione.
[2] Sono considerate le bande contigue da 9 a 24

Tasso	P-3D-s	S-3D-s	S-3D-a	S-2D
0.0625	3.54	4.87	5.82	5.83
0.125	4.34	6.26	7.18	7.20
0.25	5.32	8.12	9.25	9.26
0.5	6.21	10.35	12.25	12.26
1	7.22	13.92	16.43	16.45

Tabella 5.6: Confronto tra *alberi descrittori* per un'immagine multispettrale IKONOS. Decomposizione DWT-3D *packet* 6/6/2.

Tasso	P-3D-s	S-3D-s	S-3D-a	S-2D
0.625	11.20	11.35	12.32	12.37
0.125	13.03	13.08	13.74	13.77
0.25	14.78	14.78	15.21	15.22
0.5	15.93	15.95	16.67	16.69
1	18.05	18.13	18.82	18.83

Tabella 5.7: Confronto tra *alberi descrittori* per un'immagine iperspettrale GER. Decomposizione DWT-3D *packet* 6/6/4.

Tasso	DPR	S-2D
0.0625	6.02	6.01
0.125	7.52	7.51
0.25	9.56	9.55
0.5	12.83	12.83
1	17.45	17.45

Tabella 5.8: Confronto tra *alberi descrittori* per un'immagine multispettrale IKONOS. Decomposizione DPR 6/6/4.

5.3.5 Codificatori multicanale basati su analisi DPR

Un'immagine telerilevata multibanda può essere analizzata anche con una KLT spettrale ed una DWT-2D su ciascuna banda trasformata. Questo tipo di analisi, seguita da una codifica basata su albero descrittore DPR, è stata per la prima volta proposta in [7] ed è qui ripresa per un confronto con il più semplice schema S-2D.

Anche in questo caso riportiamo gli esperimenti relativi ad un'immagine IKONOS e ad uno spezzone di immagine GER, le stesse presentate nel paragrafo precedente. La KLT lungo la dimensione spettrale è eseguita in blocco su tutte le bande. Nella dimensione trasversa è invece applicata una DWT-2D a 6 livelli di decomposizione.

I risultati sperimentali sono riportati in tab. 5.8 e tab. 5.9. Come prima cosa si osservi che la decomposizione DPR offre migliori prestazioni rispetto alla DWT-3D *packet*, in quanto la KLT spettrale, a differenza della DWT che è a coefficienti fissi, è progettata sulle statistiche dei dati. Per quel che riguarda gli *alberi descrittori*, le differenze prestazionali tra le configurazioni DPR ed S-2D non sembrano essere per niente significative.

In definitiva, nel caso di immagini telerilevate, sia per la decomposizione DWT-3D *packet* che per quella DPR, l'albero descrittore più conveniente è S-2D, in quanto, a parità di prestazioni, è quello più semplice.

Tasso	DPR	S-2D
0.0625	13.50	13.45
0.125	14.96	14.94
0.25	16.28	16.31
0.5	18.10	18.10
1	20.21	20.22

Tabella 5.9: Confronto tra *alberi descrittori* per un'immagine iperspettrale GER. Decomposizione DPR 6/6/16.

5.4 Scelta degli ordini di zerotree

Le istanze storiche più note, EZW e SPIHT, prevedono l'una soltanto *zerotree* di *ordine* 0, l'altra *zerotree* di *ordine* 1 e 2. Le prestazioni di SPIHT, soprattutto senza codifica entropica, sono superiori a quelle di EZW. Ciò è stato evidenziato in vari articoli ed anche in [12], in cui si attribuisce la superiorità di SPIHT all'impiego di *zerotree* di *ordine* più alto. Gli esperimenti riportati in questo capitolo costituiscono, rispetto a [12], un'analisi più accurata, dando un'idea quantitativa del miglioramento prestazionale apportato dalla graduale introduzione di *zerotree* di *ordine* sempre più alto.

5.4.1 Configurazioni sperimentali

Le configurazioni storiche di *zerotree coder*, non sono adatte all'analisi sistematica che vogliamo compiere in quanto, EZW [4] presenta una codifica delle transizioni piuttosto inefficiente che non può esulare da uno stadio successivo di codifica entropica, SPIHT [5] invece non impiega *zerotree* di ordine 0, senza alcuna giustificazione teorica o sperimentale a supporto di questa scelta.
Verranno dunque introdotti tre nuovi codificatori. Gli ordini ammessi saranno rispettivamente 0, per il primo, sia 0 che 1, per il secondo, 0, 1 e 2, per il terzo. Sinteticamente indicheremo in seguito suddetti codificatori con le notazioni **Ord0**, **Ord01** e **Ord012**. Le *tabelle di evoluzione* per i tre codificatori sono rispettivamente la , la e la , mentre le leggi di evoluzione degli alberi sono riportate nelle figure , e .
Il codificatore Ord0 ha un comportamento simile ad EZW ad eccezione della codifica delle *transizioni* che è effttuata in maniera più razionale, il codificatore Ord1 ha una *legge di evoluzione* progettata sulla falsariga di SPIHT ma gli *ordini* previsti sono diversi, lo *zerotree coder* Ord012 invece si differenzia sia

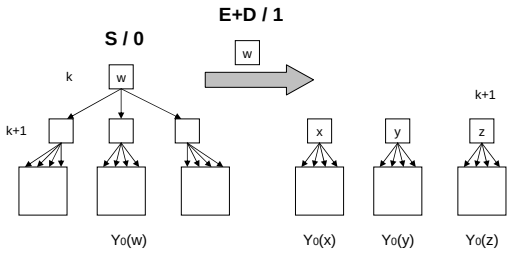


Figura 5.10: *Diagramma di evoluzione degli alberi* di **Ord0**.

da EZW che da SPIHT per ammettere anche tre possibili transizioni differenti per stato e per una funzione di test della significatività più complessa.

Negli esperimenti svolti è stato necessario fissare l'*albero descrittore*, indipendentemente dalla legge di evoluzione. Abbiamo dunque scelto, nel caso bidimensionale, l'albero di Shapiro S-2D, nel caso trimensionale, lo schema S-3D-a per la DWT-3D *packet* e lo schema DPR per la *decomposizione di Dragotti-Poggi-Ragozini*.

Ordine(k-1)	**Ordine(k)**	**Transizione**	**Codifica**
$= 0$	$= 0$	P	0
$= 0$	> 0	E+D	1

Tabella 5.10: *Tabella di evoluzione degli alberi* per **Ord0**.

Ordine(k-1)	**Ordine(k)**	**Transizione**	**Codifica**
$= 0$	$= 0$	S	0
$= 0$	> 0	E	1
$= 1$	$= 1$	S	0
$= 1$	> 1	D	1

Tabella 5.11: *Tabella di evoluzione degli alberi* per **Ord01**.

5.4.2 Codifica di immagini fisse

Come prima cosa valutiamo come variano le prestazioni al crescere del massimo *ordine di zerotree* per immagini monocromatiche. Le immagini naturali di

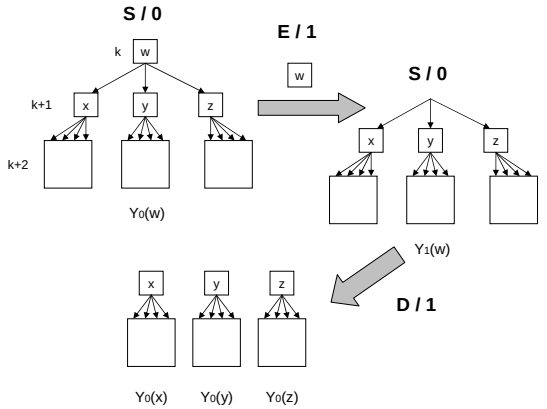


Figura 5.11: *Diagramma di evoluzione degli alberi* per **Ord01**.

Ordine(k-1)	Ordine(k)	Transizione	Codifica
$= 0$	$= 0$	S	0
$= 0$	$= 1$	E+S	10
$= 0$	> 1	E+E	11
$= 1$	$= 1$	S	0
$= 1$	$= 2$	E+S	10
$= 1$	> 2	D	11
$= 2$	$= 2$	S	0
$= 2$	> 2	D	1

Tabella 5.12: *Tabella di evoluzione degli alberi* per **Ord012**.

test sono Lena e Goldhill, di 512×512 *pixel*. Lena rappresenta il volto di una persona, mentre Goldhill un paesaggio cittadino, per cui le statistiche delle due immagini sono abbastanza differenti.
Si nota per entrambe le immagini (tab. 5.13 e tab. 5.15) un progressivo miglioramento aggiungendo *zerotree* di *ordine* via via più alto. La differenza di prestazioni è rilevabile soprattutto ai *bit-rate* più bassi ed è più alta tra i codificatori Ord01 e Ord0, rispetto ai codificatori Ord012 ed Ord01.
In tab. 5.14 e in tab. 5.16 si può osservare cosa succede effettuando la codifica aritmetica sugli elementi sintattici emessi dal codificatore. Lo schema di codifica aritmetica, per tutti gli esperimenti riportati nel corrente capitolo, è lo stesso illustrato nel capitolo precedente, in cui le statistiche sono calcolate per

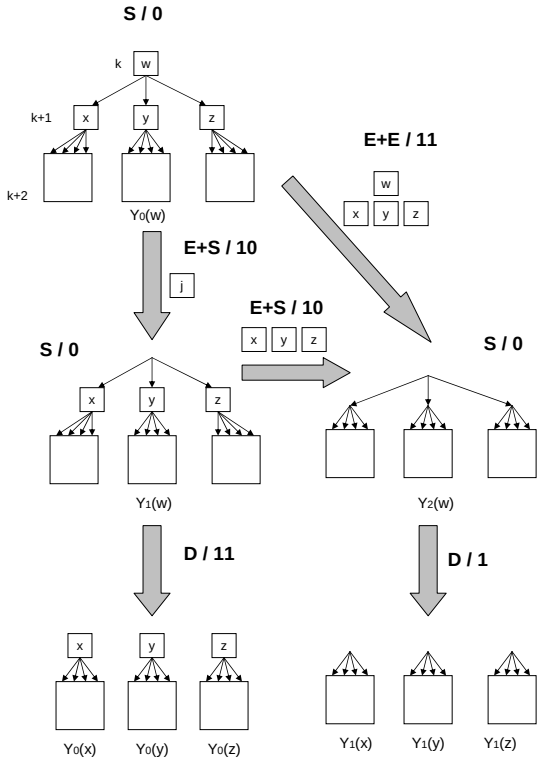


Figura 5.12: *Diagramma di evoluzione degli alberi* per **Ord012**.

ciascun tipo di transizione, per ciascun bitplane e per ciascuna sottobanda.
Con la codifica aritmetica le prestazioni migliorano per tutti i codificatori. Gli *zerotree coder* che se ne avvantaggiano di più sono in particolare quelli con *zerotree* di *ordine* più basso. Per alti *bit-rate* le prestazioni di tutti e tre gli schemi si avvicinano le une alle altre.

5.4.3 Codifica di immagini multispettrali

Per le immagini multispettrali sono stati effettuati esperimenti sia con la decomposizione DWT-3D *packet* che con quella Dragotti-Poggi-Ragozini.
Le immagini di test sono la IKONOS già presentata nei paragrafi precedenti ed un'immagine LANDSAT TM 512×512 di 6 bande spettrali [3], normalizzate

[3]Come nel caso della IKONOS sono prese in considerazione per semplicità solo le bande con la stessa risoluzione.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	26.87	27.39	27.72
0.125	29.16	30.08	30.51
0.25	31.82	33.18	33.56
0.5	35.05	36.43	36.72
1	38.67	39.74	39.93

Tabella 5.13: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per l'immagine monocromatica Lena. La decomposizione è DWT-2D 6/6.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	27.60	27.76	27.83
0.125	30.21	30.35	30.61
0.25	33.06	33.23	33.66
0.5	36.25	36.39	36.76
1	39.61	39.71	39.94

Tabella 5.14: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per l'immagine monocromatica Lena, a valle della codifica aritmetica. La decomposizione è DWT-2D 6/6.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	25.77	26.10	26.27
0.125	27.31	27.89	28.10
0.25	29.14	29.84	30.15
0.5	31.46	32.20	32.48
1	34.66	35.46	35.72

Tabella 5.15: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per l'immagine monocromatica Goldhill. La decomposizione è DWT-2D 6/6.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	26.34	26.22	26.32
0.125	27.88	27.93	28.18
0.25	29.78	30.13	30.32
0.5	32.09	32.28	32.60
1	35.37	35.51	35.89

Tabella 5.16: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per l'immagine monocromatica Goldhill, a valle della codifica aritmetica. La decomposizione è DWT-2D 6/6.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	5.33	5.62	5.79
0.125	6.53	6.96	7.14
0.25	8.48	8.98	9.14
0.5	11.22	11.97	12.21
1	15.11	16.16	16.34

Tabella 5.17: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un'immagine multispettrale IKONOS. La decomposizione è di tipo DWT-3D packet 6/6/2.

questa volta a potenza unitaria [4].

Anche per le immagini multispettrali, gli esperimenti confermano l'andamento visto nel caso bidimensionale. Risultati sperimentali sono riportati con e senza codifica aritmetica. Le tabelle relative ai suddetti esperimenti sono le seguenti: 5.17,5.18, 5.19, 5.20, per l'immagine IKONOS, 5.21 e 5.22 per l'immagine LANDSAT.

5.4.4 Codifica di immagini iperspettrali

Le immagini iperspettrali hanno una quantità di bande (decine, oppure una o due centinaia) molto più alta rispetto alle immagini multispettrali (sotto al decina). Inoltre le bande contigue sono tra loro molto simili. Per questo mo-

[4] La normalizzazione è effettuata rispetto alla potenza complessiva, pari al quadrato della media più la varianza.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	5.69	5.69	5.86
0.125	6.92	7.09	7.30
0.25	8.99	9.11	9.33
0.5	11.92	12.13	12.40
1	16.12	16.40	16.67

Tabella 5.18: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un'immagine multispettrale IKONOS, dopo la codifica aritmetica. La decomposizione è di tipo DWT-3D packet 6/6/2.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	5.58	5.87	6.03
0.125	6.83	7.29	7.48
0.25	8.85	9.39	9.57
0.5	11.94	12.60	12.81
1	16.39	17.24	17.44

Tabella 5.19: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un'immagine multispettrale IKONOS. La decomposizione è di tipo DPR 6/6/4.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	5.93	5.95	6.11
0.125	7.27	7.46	7.69
0.25	9.40	9.53	9.79
0.5	12.59	12.76	13.04
1	17.25	17.47	17.79

Tabella 5.20: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un'immagine multispettrale IKONOS, dopo la codifica aritmetica. La decomposizione è di tipo DPR 6/6/4.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	18.72	19.13	19.39
0.125	20.32	21.24	21.52
0.25	22.97	23.68	23.92
0.5	26.55	27.32	27.52
1	31.39	32.27	32.45

Tabella 5.21: Prestazioni, misurate come inverso dell'MSE, di codificatori che impiegano *ordini di zerotree* differenti per un'immagine multispettrale LANDSAT. La decomposizione è di tipo DPR 6/6/6.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	19.24	19.20	19.50
0.125	21.29	21.57	21.71
0.25	23.75	23.92	24.15
0.5	27.40	27.52	27.80
1	32.42	32.57	32.84

Tabella 5.22: Prestazioni, misurate come inverso dell'MSE, di codificatori che impiegano *ordini di zerotree* differenti per un'immagine multispettrale LANDSAT, dopo la codifica aritmetica. La decomposizione è di tipo DPR 6/6/6.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	11.50	12.17	12.36
0.125	13.00	13.55	13.75
0.25	14.69	15.06	15.17
0.5	16.35	16.59	16.65
1	18.41	18.40	18.42

Tabella 5.23: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un GOP di 16 bande di un'immagine iperspettrale GER. La decomposizione è di tipo DWT-3D packet 6/6/4.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	12.15	12.26	12.44
0.125	13.47	13.69	13.85
0.25	15.04	15.13	15.29
0.5	16.68	16.72	16.82
1	18.97	18.60	19.07

Tabella 5.24: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un GOP di 16 bande di un'immagine iperspettrale GER, dopo la codifica aritmetica. La decomposizione è di tipo DWT-3D packet 6/6/4.

tivo, così come si fa per le sequenze video, può valere la pena segmentare la pila di immagini in GOP.
Nei nostri esperimenti riportiamo i risultati relativi ad un GOP di un'immagine GER, la stessa già presentata nei paragrafi precedenti. I risultati (tab. 5.23, 5.24, 5.25, 5.26) confermano quanto già detto per le immagini monocromatiche e per le immagini multispettrali.

5.4.5 Codifica di sequenze video senza compensazione del movimento

Le sequenze video scelte negli esperimenti sono Mobile e Coastguard. Per esse infatti ha più senso uno schema di codifica e compensazione del movimento basato su modello di *moto traslazionale a blocchi*, così come sarà proposto nel seguente paragrafo. In questo paragrafo invece presentiamo

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	12.89	13.32	13.48
0.125	14.47	14.87	14.96
0.25	16.05	16.22	16.34
0.5	17.97	18.07	18.09
1	20.33	20.24	20.22

Tabella 5.25: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un GOP di 16 bande di un'immagine iperspettrale GER. La decomposizione è di tipo DPR 6/6/16.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	13.30	13.41	13.57
0.125	14.87	15.01	15.13
0.25	16.39	16.44	16.62
0.5	18.28	18.29	18.39
1	20.98	20.95	20.99

Tabella 5.26: Prestazioni in SNR di codificatori che impiegano *ordini di zerotree* differenti per un GOP di 16 bande di un'immagine iperspettrale GER, dopo la codifica aritmetica. La decomposizione è di tipo DPR 6/6/16.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	19.32	19.61	19.76
0.125	20.45	20.89	21.01
0.25	21.95	22.47	22.66
0.5	24.34	24.92	25.11
1	27.98	28.90	29.05

Tabella 5.27: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Mobile. La decomposizione è di tipo DWT-3D $5/5/4$. Non vi è compensazione del movimento.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	19.69	19.93	20.00
0.125	20.98	21.21	21.31
0.25	22.57	22.86	23.01
0.5	25.08	25.32	25.54
1	29.01	29.34	29.56

Tabella 5.28: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Mobile, dopo la codifica aritmetica. La decomposizione è di tipo DWT-3D $5/5/4$. Non vi è compensazione del movimento.

risultati per decomposizione DWT-3D *packet* senza compensazione del movimento.

Anche nel caso di sequenze video la *legge di evoluzione* Ord012 offre migliori prestazioni della legge Ord01. Allo stesso modo Ord01 presenta un PSNR più alto a tutti i *bit-rate*. L'aggiunta della codifica aritmetica ridimensiona il divario di prestazioni, ma tenere in conto *zerotree* di ordine superiore si mostra sempre una strategia vincente. I risultati sono riportati nelle tab. 5.27, 5.28, 5.29 e 5.30.

Tasso	**Ord 0**	**Ord 0,1**	**Ord 0,1,2**
0.0625	26.37	26.73	26.87
0.125	27.75	28.18	28.33
0.25	29.35	29.90	30.09
0.5	31.60	32.17	32.44
1	34.81	35.53	35.94

Tabella 5.29: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Coastguard. La decomposizione è di tipo DWT-3D $5/5/4$. Non vi è compensazione del movimento.

Tasso	**Ord 0**	**Ord 0,1**	**Ord 0,1,2**
0.0625	26.70	26.95	27.03
0.125	28.16	28.40	28.55
0.25	29.86	30.33	30.48
0.5	32.20	32.53	32.76
1	35.56	36.01	36.40

Tabella 5.30: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Coastguard, dopo la codifica aritmetica. La decomposizione è di tipo DWT-3D $5/5/4$. Non vi è compensazione del movimento.

Tasso	Ord 0	Ord 0,1	Ord 0,1,2
0.0625	20.93	21.20	21.28
0.125	22.70	22.99	23.07
0.25	24.65	25.09	25.04
0.5	27.22	27.50	27.51
1	31.03	31.38	31.37

Tabella 5.31: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Mobile. La decomposizione è di tipo DWT-3D $5/5/4$. E' effettuata la compensazione del movimento mediante lifting scheme.

5.4.6 Codifica di sequenze video con compensazione del movimento

La stima del movimento è effettuata su blocchi 8×8, su una *finestra di ricerca* 32×32, mediante ricerca esaustiva. La precisione è limitata a 1 *pixel* intero. Il costo dei campi di moto è assunto nullo, per cui le *curve tasso-distorsione* presentate non sono quelle reali ma una sorta di limite superiore per le prestazioni.

Per i video presi in considerazione il miglioramento in PSNR è netto rispetto all'assenza di compensazione del movimento. La differenza tra le varie leggi di evoluzione invece, pur conservandosi il trend visto negli altri casi, è meno marcata sia a bassi che ad alti *bit-rate*. I risultati degli esperimenti sono riportati sinteticamente nelle tab. 5.31, 5.32, 5.33 e 5.34.

Tasso	**Ord 0**	**Ord 0,1**	**Ord 0,1,2**
0.0625	21.19	21.31	21.45
0.125	23.09	23.10	23.23
0.25	25.24	25.35	25.41
0.5	27.84	27.94.	28.00
1	31.87	31.94	32.05

Tabella 5.32: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Mobile, dopo la codifica aritmetica. La decomposizione è di tipo DWT-3D $5/5/4$. E' effettuata la compensazione del movimento mediante lifting scheme.

Tasso	**Ord 0**	**Ord 0,1**	**Ord 0,1,2**
0.0625	25.89	25.86	26.05
0.125	27.51	27.83	28.05
0.25	29.44	30.03	30.09
0.5	32.27	32.67	32.79
1	36.00	36.47	36.69

Tabella 5.33: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Coastguard. La decomposizione è di tipo DWT-3D $5/5/4$. E' effettuata la compensazione del movimento mediante lifting scheme.

Tasso	**Ord 0**	**Ord 0,1**	**Ord 0,1,2**
0.0625	25.93	25.93	26.17
0.125	27.84	27.85	28.09
0.25	30.05	30.22	30.38
0.5	32.71	32.97	33.23
1	36.59	36.98	37.21

Tabella 5.34: Prestazioni in PSNR di codificatori che impiegano *ordini di zerotree* differenti per 2 GOP di 16 bande della sequenza video Coastguard, dopo la codifica aritmetica. La decomposizione è di tipo DWT-3D $5/5/4$. E' effettuata la compensazione del movimento mediante lifting scheme.

Tasso	Ord 0,1,2	Ord 1,2
0.0625	26.32	25.91
0.125	28.18	27.89
0.25	30.32	30.10
0.5	32.60	32.69
1	35.89	36.00

Tabella 5.35: Confronto tra la *legge di evoluzione* Ord012 e quella Ord12 di SPIHT per l'immagine monocromatica Goldhill.

Tasso	Ord 0,1,2	Ord 1,2
0.0625	19.76	19.75
0.125	21.01	20.99
0.25	22.66	22.64
0.5	25.11	25.10
1	29.05	29.04

Tabella 5.36: Confronto tra la *legge di evoluzione* Ord012 e quella Ord12 di SPIHT per la sequenza video Mobile.

5.4.7 Confronto con SPIHT

Infine presentiamo un confronto tra la *legge di evoluzione* Ord012 e la *legge di evoluzione* presentata in [5], che denotiamo, d'ora in avanti, con la sigla **Ord12**. I confronti sono effettuati senza codifica entropica. Al di là dei risultati preliminari incoraggianti, presentati in [3], non sono emerse conferme ulteriori su un valore aggiunto della strategia Ord012 rispetto alla strategia Ord12. Nella maggior parte dei casi il discostamento di prestazioni non è affatto significativo, confermando la ragionevolezza dell'approccio di Ord12 di trascurare gli alberi di ordine 0. Le prestazioni tipiche sono quelle delle tab. 5.35, 5.36, 5.36, 5.37 e 5.38.

Tasso	Ord 0,1,2	Ord 1,2
0.0625	12.36	12.33
0.125	13.75	13.77
0.25	15.17	15.20
0.5	16.65	16.67
1	18.42	18.43

Tabella 5.37: Confronto tra la *legge di evoluzione* Ord012 e quella Ord12 di SPIHT per l'immagine iperspettrale GER. La decomposizione è DWT-3D packet.

Tasso	Ord 0,1,2	Ord 1,2
0.0625	13.48	13.54
0.125	14.96	14.95
0.25	16.34	16.39
0.5	18.09	18.11
1	20.22	20.25

Tabella 5.38: Confronto tra la *legge di evoluzione* Ord012 e quella Ord12 di SPIHT per l'immagine iperspettrale GER. La decomposizione è DPR.

5.5 Conclusioni

In questo capitolo di analisi sperimentale sono state considerate diversi progetti di *zerotree coder* per differenti analisi multidimensionali e tipi di dati. Si sono individuati due fattori di rilievo che influenzano le prestazioni:

- l'*albero descrittore* della trasformata;
- l'*ordine* massimo di *zerotree* ammesso.

Per il primo fattore si è riscontrato un comportamento diverso a seconda della tipologia dei dati. Nello specifico, per sequenze video senza compensazione del movimento, l'albero migliore è risultato quello tridimensionale *asimmetrico* che abbiamo denominato S-3D-a. Nel caso di immagini multicanale telerilevate invece, l'albero tridimensionale S-3D-a, per quanto decisamente migliore delle sue alternative 3D, non ha dimostrato un *gap* prestazionale significativo rispetto alla sua versione bidimensionale S-2D. Gli alberi bidimensionali si sono dimostrati adeguati anche nel caso di analisi DPR.
Per quel che riguarda l'impiego di *zerotree* di diverso *ordine*, si è visto che *ordini* più alti garantiscono migliori prestazioni, anche se la codifica aritmetica può risollevare di molto le prestazioni degli schemi che utilizzano *zerotree* di *ordine* più basso. Per la prima volta in letteratura un'analisi sperimentale completa è stata presentata su questo argomento.

Bibliografia

[1] S. Mallat, “A wavelet tour into signal processing”, Elsevier, 1998.

[2] W.Sweldens, P.Shroeder: “Building your own wavelet at home” - *Wavelets in Computer Graphics*, ACM SIGGRAPH Course Notes, 1996.

[3] L. Cicala, G. Poggi, “A generalization of zerotree coding algorithms”, Picture Coding Symposium , Lisbona (Portogallo), n. 1050, novembre 2007.

[4] J.M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients”, *IEEE Trans. On Signal Processing*, vol. 41, n.12 pp. 3445-3463, dicembre 1993.

[5] A. Said, W. A. Pearlman: “New fast and efficient image codec based on set partioning in hierarchical tree’, *IEEE Trans. On Circuits and Systems on Video Technology*, vol. 6, pp. 243-250, giugno 1996.

[6] B-J. Kim, Z. Xiong, and W. A. Pearlman, “Low Bit-Rate Scalable Video Coding with 3D Set Partitioning in Hierarchical Trees (3D SPIHT)”, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 10, pp. 1374-1387, dicembre 2000.

[7] P.L. Dragotti, G. Poggi, A.R.P. Ragozini: “Compression of miltispectral images by three-dimensional SPIHT algorithm”, *IEEE Trans. On Geoscience and Remote Sensing*, vol. 38, gennaio 2000.

[8] D.Taubman, “High performance scalable image compression with EBCOT”, *IEEE Trans. On Image Processing*, Volume 9, Issue 7, pp.1158 - 1170, luglio 2000.

[9] C. He, J. Dong, Y. F. Zheng. Z. Gao, “Optimal 3D coefficient tree structure for 3D wavelet video coding” - *IEEE Trans. On Circuits and Systems on Video Technology*, vol. 13, ottobre 2003.

[10] H. Danyali and A. Mertins, “A fully SNR, spatial and temporal scalable 3DSPIHT-based video coding algorithm for video streaming over heterogeneous networks ” , *Proc. TENCON*, Bangalore, India, vol. 4, pp. 1445-1449, ottobre 2003.

[11] H. Danyali, A. Mertins: “Flexible, highly scalable, object-based wavelet image compression algorithm for network applications” , *Vision, Image and Signal Processing, IEE Proceedings*, vol. 151, dicembre 2004.

[12] Y. Cho and W. A. Pearlman, “Quantifying the Performance of Zerotrees of Wavelet Coefficients: Degree-k Zerotree Model” , *IEEE Trans. on Signal Processing*, Vol. 55, Part 1, pp. 2425-2431, giugno 2007.

[13] E. Christophe and W. A. Pearlman, “Three-dimensional SPIHT Coding of Volume Images with Random Access and Resolution Scalability”, submitted to *Eurasip J. on Image and Video Processing*, settembre. 2007.

[14] A. Secker and D. Taubman “Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression”, *IEEE Transactions on Image Processing*, vol. 12, n. 12, pp. 1530-1542, dicembre 2003.

[15] M. Cagnazzo, “Wavelet transform and three-dimensional data compression”, tesi di dottorato, Univ. Federico II di Napoli ed Univ. di Nizza-Sophia Antipolis (Francia), marzo 2005.

Capitolo 6

Codifica mediante trasformata classificata

Parole chiave

Nozioni propedeutiche

Quantizzazione vettoriale, codifica mediante trasformata.

Concetti introdotti nel capitolo

Apprendimento statistico, classificazione, clustering, apprendimento supervisionato, insieme di apprendimento, apprendimento non supervisionato, poggetto off-line, progetto on-line, apprendimento on-line, segmentazione, segmentazione contestuale, segmentazione non contestuale, codifica mediante classificazione, trasformata classificata.

6.1 Trasformata classificata per la codifica

La *quantizzazione scalare* è la più semplice tecnica di approssimazione dei dati per la codifica *lossy*. In fig. 6.1 è mostrato, in un esempio bidimensionale, come due quantizzatori scalari uniformi impongano un *reticolo* sullo spazio dei dati. I centroidi del *reticolo* sono i valori con cui i dati vengono approssimati dal processo di *quantizzazione*. Nell'esempio presentato tutti i dati sono racchiusi in un rettangolo di 24 celle, a cui corrispondono 24 indici di quantizzazione. Ciascun indice, senza ausilio di codifica entropica, può essere dunque codificato con $log_2 24$ bit.
Anche nel campo della codifica, come per altre discipline, l'approssimazione lineare delle dipendenze statistiche permette di trattare i problemi in maniera semplice ed efficace. Questo è principio alla base della *codifica mediante trasformata*. Nella fig. 6.1 per i dati mostrati già in fig. 6.1 è individuata la direzione principale. Si può pensare dunque di ruotare opportunamente gli assi del *reticolo di quantizzazione* in modo che uno di essi giaccia proprio su di essa. Questa non è altro che una trasformata ortogonale, il cui risultato (a meno di una traslazione), è rappresentato in 6.1. Grazie alla trasformazione, le celle necessarie a ricoprire i dati con un rettangolo (quantizzazione scalare) si riducono a 20; in questo modo sono necessari solo $log_2 20$ bit per ciascun indice di quantizzazione.
In alternativa, prima di eseguire la *quantizzazione scalare*, posso pensare di classificare i dati come in fig. 6.1. Posso poi effettuare una traslazione del *reticolo di quantizzazione* per ciascuna delle due *classi*. I vettori di traslazione sono detti *vettori template* e rappresentano l'approssimazione dei dati appartenenti ad una certa *classe* che si ha attraverso la *quantizzazione vettoriale* così descritta. I due nuovi reticoli per l'esempio di fig. 6.1 sono mostrati in fig. 6.1. Grazie alla *quantizzazione vettoriale*, nell'esempio mostrato, riusciamo a ridurre il numero di celle di quantizzazione da 20 a 18. Il numero di bit per indice di quantizzazione è ora solo $log_2 18$.
Le tecniche di *codifica mediante trasformata* e di *quantizzazione vettoriale* sono le più diffuse nell'ambito della compressione dati a causa della loro riconosciuta efficacia. Si può pensare dunque di combinare le due tecniche, sperando di ottenere una rappresentazione dei dati ancora più efficiente. L'analisi da effettuare, nel caso bidimensionale, è quella in fig. 6.1: oltre ad imporre una *classificazione* sui dati, viene stabilita una direzione principale per ciascuna *classe*. In fig. 6.1 è mostrato l'effetto di una **trasformata classificata** basata sull'analisi di fig. 6.1. Abbiamo di nuovo due reticoli, come per la VQ, ma questa volta abbiamo anche effettuato un'opportuna trasformazione

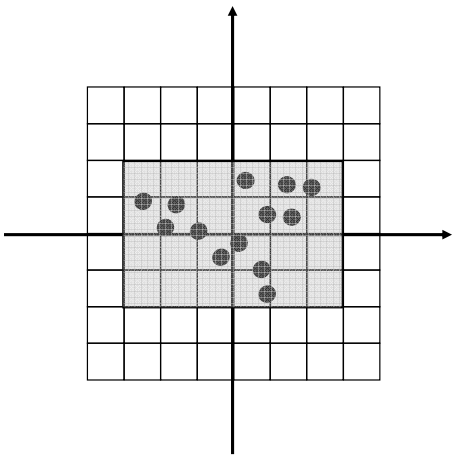


Figura 6.1: Dati bidimensionali immersi in un reticolo di quantizzazione uniforme.

ortonormale per ciascuna classe. Il numero complessivo di celle di quantizzazione è ridotto a 16. Il costo di codifica per indice è dunque ridotto a $log_2 16$ bit contro i $log_2 24$ iniziali.

In questo capitolo vedremo come applicare il principio della *trasformata classificata* alla compressione di immagini multicanale, ed in particolare alle immagini telerilevate. Inoltre integreremo questa famiglia di tecniche nel più ampio *framework* della *codifica classificata*. Prima di addentrarci però nelle problematiche di codifica, introduciamo rapidamente qualche concetto fondamentale relativo alla *classificazione statistica* e alla *segmentazione di immagini*. Queste poche nozioni risulteranno molto utili quando si parlerà di *codifica mediante classificazione*.

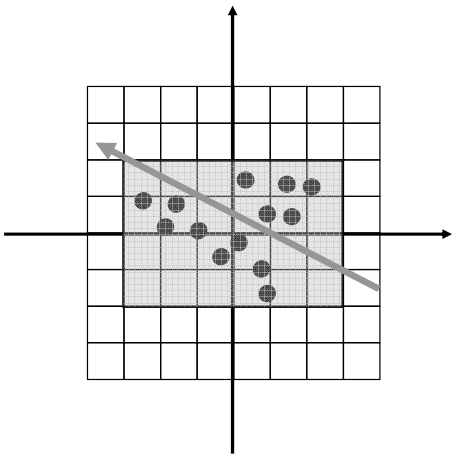


Figura 6.2: Direzione principale dei dati di fig. 6.1.

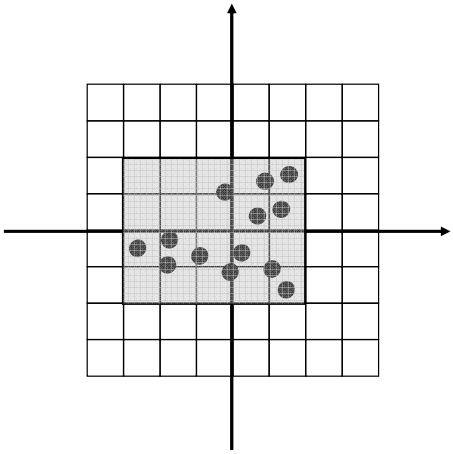


Figura 6.3: Trasformata lineare sui dati di fig. 6.1 .

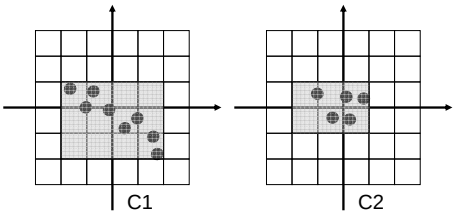


Figura 6.4: Classificazione in due classi dei dati di fig. 6.1.

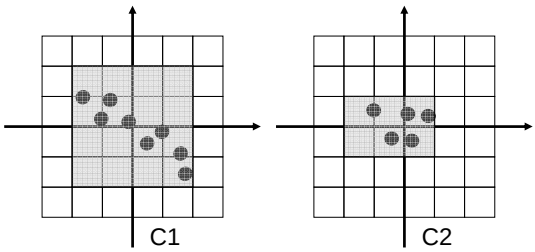


Figura 6.5: Reticoli uniformi sui dati classificati di fig. 6.1.

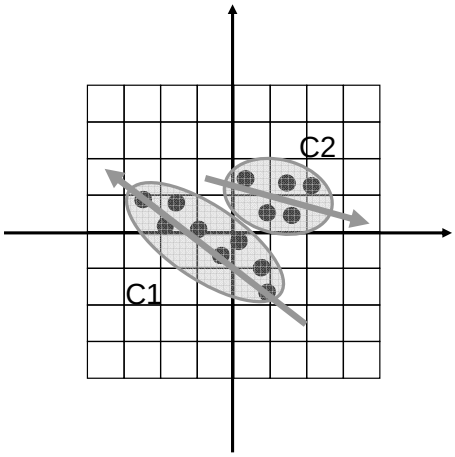


Figura 6.6: Classificazione ed individuazione delle direzioni principali locali per i dati di fig. 6.1.

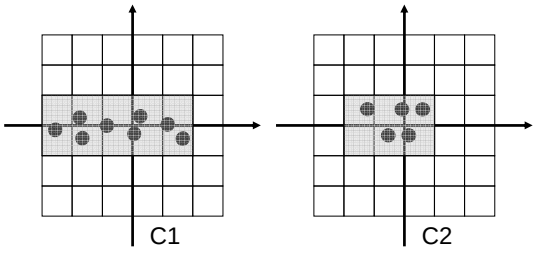


Figura 6.7: Trasformata classificata sui dati di 6.1.

6.2 Classificazione statistica e segmentazione di immagini

Un **classificatore statistico** [2] è un sistema ad n ingressi ed un'uscita. Gli ingressi rappresentano le caratteristiche (o **features**) $\underline{f}(x)$ di un oggetto x appartenente ad un certo insieme X. Le *features* possono essere continue o discrete. L'uscita y è invece un numero intero che rappresenta l'etichetta di una delle possibili **classi** Y a cui x appartiene. La funzione $y = d(\underline{f})$ che lega le *features* ad una particolare *classe* è detta **regola di decisione**.
Il progetto di un classificatore può essere effettuato in due modalità. La prima, detta di classificazione **supervisonata**, prevede di avere a disposizione degli esempi, cioè delle coppie $(\underline{f}, y)$ che rappresentano delle classificazioni corrette, le quali costituiscono l'*insieme di apprendimento* o **training set**. Tramite degli algoritmi di *apprendimento statistico* è possibile per induzione ricavare una regola generale di decisione partendo dagli esempi. Una seconda modalità è invece quella **non supervisionata**, detta anche **clustering**. In questa modalità non si hanno esempi a disposizione, ma si procede cercando di minimizzare una figura di merito che ha caratteristiche dipendenti dall'applicazione.
Il progetto del classificatore può inoltre avvenire **off-line** oppure **on-line**. Nel primo caso esso è progettato prima di essere messo in funzione, nel secondo caso la sua struttura si modifica automaticamente man mano che esso opera su nuovi dati. Per *classificatori supervisionati* la modalità di progetto *off-line* è quella più naturale, per quelli *non supervisionati* invece entrambe le strategia di progetto sono diffuse.

6.2.1 La segmentazione di immagini

La *segmentazione* di immagini è un particolare processo di *classificazione* mediante cui si assegna a ciascun *pixel* una determinata *classe*. L'obiettivo della *segmentazione* è la minimizzazione di un certo funzionale che cambia a seconda del metodo e dell'applicazione. Solitamente un requisito importante è quello di avere regioni connesse ed omogenee.
Per gli algoritmi di *segmentazione* sussiste la stessa tassonomia che vige per i classificatori. Un'ulteriore distinzione si può fare a seconda che le feature prese in considerazione siano legate alle proprietà del singolo *pixel* o anche a quelle di *pixel* vicini. Si parla dunque, a seconda, di segmentazione **non contestuale** oppure **contestuale**.

Esistono diversi modelli di *segmentazione* per immagini. Molti di loro

sono formalizzabili attraverso il paradigma dell'*apprendimento statistico* [2], altri sono storicamente formalizzati secondo altri approcci [1]. Ci si limiterà in seguito a descrivere alcuni algoritmi, tra i più celebri afferenti al paradigma della *pattern analysis*.

Una strategia semplice di *segmentazione non contestuale* è quella di associare a ciascuna *classe* la media statistica dei vettori ad essa afferenti, detta **vettore template**, e dunque minimizzare l'*errore quadratico medio* per ciascuna *classe*. Questo approccio è detto a minima distanza. Esso è ideale per un clustering molto fitto dei dati.

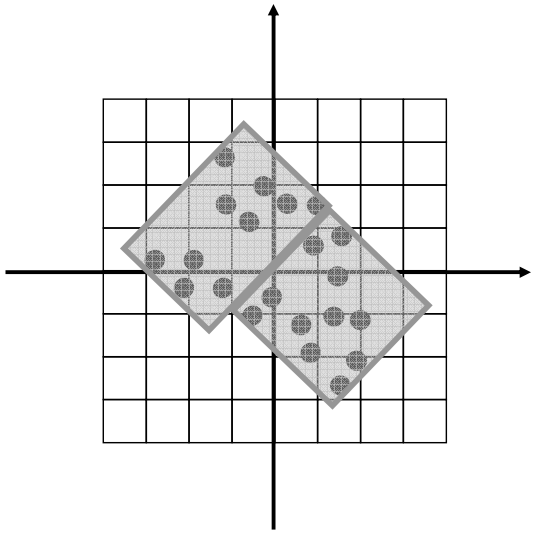
Un'altro approccio è quello di rappresentare ciascuna *classe* non solo con la media ma anche con la matrice di covarianza. Si fa dunque l'ipotesi che la distribuzione di ciascuna *classe* sia gaussiana. Diversi algoritmi di apprendimento statistico (es. *massima verosimiglianza*, *expectation maximization*) possono essere impiegati allo scopo di scegliere i parametri che definiscono il classificatore.

Un metodo di *segmentazione contestuale* di successo è quello basato su *campi markoviani*. Nei *campi di Markov* [5] l'etichetta di *classe* di ciascun *pixel* dipende dal valore del *pixel* stesso e da quello di pochi *pixel* circostanti, secondo un particolare modello di vicinato. Ancora una volta diverse strategie di apprendimento statistico determinano la segmentazione dell'immagine.

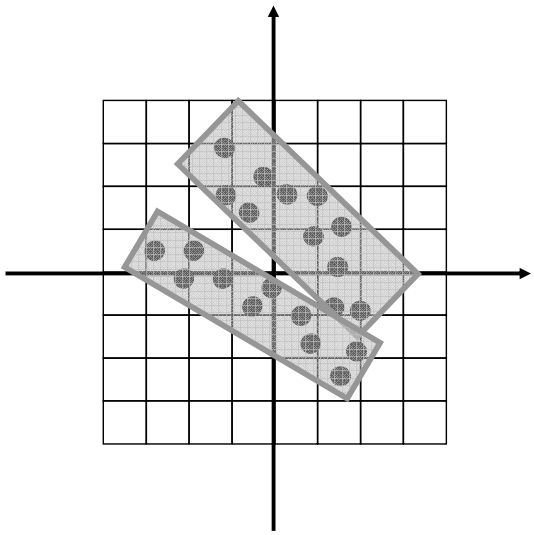
Come per la quantizzazione vettoriale, anche per le strategie di segmentazione è possibile un approccio strutturato, che presenta il vantaggio di ridurne la complessità. Per i campi di Markov ad esempio è stato proposta una struttura ad albero, che è stata formalizzata in [6].

6.3 La KLT classificata

Una *trasformata classificata* in linea teorica è un processo congiunto di *classificazione* e di *trasformata adattativa*, che ha come obiettivo quello di permettere la costruzione del reticolo scalare ottimo per i dati da codificare. Questo problema, espresso qui informalmente, è estremamente complesso ed affascinante, tuttavia non è ancora stato affrontato in letteratura.
Per adesso, il problema che trattato e risolto, è ben più semplice. Supposto di avere dati già classificati, è possibile trovare una trasformata lineare *adattata alla classe* (coefficienti che variano a seconda della stessa), che, sotto l'ipotesi di distribuzione gaussiana per i dati di ciascuna *classe*, fornisca il risultato migliore di compattazione dell'energia. Questa trasformata non è altro che una KLT con coefficienti *adattati alla classe*, detta **KLT classificata**. Se la distribuzione statistica dei dati di ciascuna *classe* è ancora simile ad una gaussiana, si può essere ben fiduciosi che la *KLT classificata* garantisca un reticolo di quantizzazione migliore rispetto alla versione *flat*. La *KLT classificata* viene utilizzata in pratica nel seguente modo: prima si effettua una *classificazione* dei dati con un criterio di minimizzazione dell'MSE, poi si progetta la *trasformata classificata* sulle *classi* così ottenute.
Naturalmente, questo problema è ben diverso e ben più semplice di quello generale di trovare la combinazione di *classificazione* e *trasformata adattativa*, che garantisca l'ottimalità del *reticolo di quantizzazione*. Mostriamo per semplicità in fig. 6.8, il confronto tra due diverse soluzioni, l'una ottenuta mediante una *classificazione* che minimizza l'MSE sui dati ed una *trasformata classificata* progettata a valle della stessa, l'altra realizzata mediante un'ipotetico progetto congiunto. Si può notare che il reticolo ottenuto mediante progetto congiunto è più efficiente, presentando un'area minore.
In seguito, parlando di *trasformata classificata*, si farà sempre riferimento a schemi di progetto separato. Infatti, per ora, gli strumenti teorici non consentono ancora di affrontare il problema del progetto congiunto.



(a)



(b)

Figura 6.8: (a) Classificazione e trasformata classificata progettate separatamente. (b) Ipotetico progetto congiunto.

6.4 Codifica mediante classificazione di immagini multicanale

Negli ultimi anni è stata rivolta dagli standard, JPEG2000 ed MPEG4, una grande attenzione alla **codifica di oggetti**. In questo paradigma di codifica [3] le immagini o le *frame* di una sequenza video, vengono segmentate in **oggetti** che rappresentano la proiezione sul piano immagine di effettivi oggetti della scena. Gli oggetti così ottenuti, di natura spaziale o spazio-temporale, vengono codificati autonomamente. Gli obiettivi possono essere due. Il primo riguarda una questione di efficienza e punta a migliorare le prestazioni di codifica in termini di qualità, solitamente ai *bit-rate* più bassi. Il secondo riguarda invece una questione di *funzionalità*: in questo modo, da un lato, è possibile avere *accesso diretto* agli oggetti nel flusso di codifica, dall'altro, vi è l'opportunità di decodificare soltanto gli oggetti di interesse (si parla dunque di **scalabiltà di contenuti**).
La partizione di un'immagine in *oggetti* è memorizzata in una *mappa di segmentazione*, la quale di per sè costituisce un'informazione aggiuntiva da codificare. Tecniche pensate appositamente per la *codifica di oggetti* possono tuttavia sfruttare la *mappa* per codificare meglio gli *oggetti* e controbilanciare il costo della codifica della stessa [11].

Per alcuni tipi di contenuti visuali non è tanto facile parlare di *oggetti*. Si pensi ad esempio alle immagini telerilevate. Gli oggetti della scena possono proiettare sul piano immagine rappresentazioni davvero piccole e confuse. Per le immagini a bassa risoluzione (ordine delle decine di metri per *pixel*) si possono al più identificare delle regioni di oggetti affini: aree urbane, bacini idrici, vegetazione, suolo coltivato. Per le immagini a più alta risoluzione (ordine del metro per *pixel*), in cui, ad esempio, è possibile chiaramente distinguere degli edifici, sorge invece il problema che gli oggetti sono innumerevoli ed occupano soltanto pochi *pixel*. In definitiva, per immagini come queste, la categoria *oggetto* è davvero di scarsa applicabilità; ciò che invece ha più senso è l'appartenenza dei *pixel* ad una specifica *classe di oggetti* affini.
La **codifica classificata**, rispetto a quella *di oggetti*, nasce dunque con gli stessi obiettivi e sulla base di considerazioni simili. Cambia la tipologia dei dati, che di solito sono immagini o *frame* in cui non è possibile o opportuno individuare degli *oggetti* indipendenti, quanto più delle regioni di *pixel* appartenenti ad una stessa *classe*.

La *codifica classificata*, come la *codifica ad oggetti*, può prevedere strumenti specifici volti a migliorarne l'efficienza. In primo luogo, è possibile

sfruttare l'eventuale omogeneità delle regioni per evitare l'effetto negativo sulle prestazioni delle discontinuità spaziali nette (*bordi*); questo approccio, denominato *codifica orientata alla forma* o **shape adaptive**, è stato proposto per diversi tipi di trasformate spaziali bidimensionali, tra cui la DWT [10], ed è stato ormai sottoposto a standardizzazione. Inoltre, come proposto per la prima volta in [13], è possibile adattare l'eventuale trasformata spettrale alla specifica *classe*, secondo il paradigma della *trasformata classificata*.

Sostituendo il concetto di *classe* a quello di *oggetto*, vengono meno alcuni vincoli concettuali che aprono nuove prospettive di applicazione. Una *classe*, a differenza di un *oggetto*, non deve avere necessariamente un'interpretazione nella scena. Ad esempio essa può essere semplicemente un *cluster* di *pixel* che posseggono analoghe proprietà statistiche, il cui raggruppamento consente comunque di adottare gli strumenti tipici dell'approccio classificato. Si può pensare, in quest'ottica, di effettuare una *segmentazione orientata alla compressione*, per ottenere regioni connesse e poter impiegare le trasformate *shape adaptive*, e/o per ricavare *classi* omogenee spettralmente, per consentire l'impiego proficuo di *trasformate classificate*. La classificazione dei *pixel*, o *segmentazione*, può essere guidata dunque da due obiettivi distinti: o la definizione di regioni con una semantica ben precisa nella scena, per cui si parlerà di **segmentazione orientata alla semantica**, oppure la definizione di una suddivisione in *classi* efficiente per la codifica, per cui si potrà parlare di **segmentazione orientata alla compressione**.

6.4.1 Codifica classificata orientata alla semantica

La *codifica classificata orientata alla semantica* della scena punta soprattutto alle funzionalità proprie della *codifica di oggetti*. Solitamente la *mappa di segmentazione* è ottenuta con tecniche *supervisionate*. Questo tipo di codifica, allo stato dell'arte, non è molto indicata per applicazioni *real-time*, a causa della pesantezza computazionale e della scarsa affidabilità dei sistemi di *segmentazione*. Invece le sue funzionalità possono essere di forte attrattiva quando non ci sono vincoli di operatività in tempo reale e si hanno a disposizione strumenti di *segmentazione* complessi ed efficaci.

6.4.2 Codifica classificata orientata all'efficienza

La *codifica classificata orientata all'efficienza* non presenta tutte le problematiche applicative di quella *orientata alla semantica*. Le principali sfide sono invece di carattere teorico. Non è infatti facile stabilire un funzionale di costo

che serva ad ottenere, non semplicemente regioni connesse oppure omogeneità spettrale, ma, la *segmentazione* ottimale per le prestazioni di codifica.
Le tecniche di *segmentazione* più adatte a questo tipo di codifica sono naturalmente quelle *non supervisionate*. Per adesso sono state proposte, in vari lavori, tecniche del tipo di quelle già descritte brevemente nel paragrafo 6.2.1.

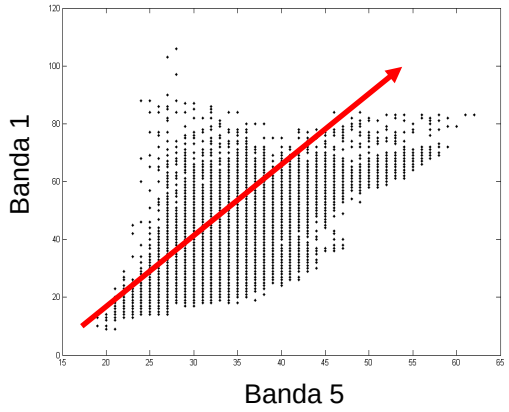


Figura 6.9: Direzione principale per dati di due bande di un'immagine LANDSAT.

6.5 Il caso di studio delle immagini telerilevate

Per quel che riguarda la *codifica classificata* un ambito applicativo di notevole interesse è quello della compressione di immagini telerilevate multicanale. Infatti, da un lato, le caratteristiche di questa categoria di dati ben si presta all'approccio classificato, dall'altro, i più importanti lavori proposti in letteratura sono rivolti proprio a questa particolare applicazione. Lo schema generale di un codificatore classificato per immagini multicanale è mostrato in 6.5. In primo luogo l'immagine viene segmentata con una strategia *contestuale* o *non contestuale*. Dopo la segmentazione, viene effettuata una trasformata spettrale, che eventualmente può essere *classificata*. Infine è effettuata una trasformata spaziale su ciascuna regione della mappa, eventualmente con tecniche specifiche per la codifica per classi. In figura è mostrato anche un blocco funzionale preposto all'allocazione delle risorse *a posteriori*. Ciò non è vincolante in quanto possono essere impiegati anche altri tipi di allocazione.

Il primo lavoro presentato a rivista su questo argomento è [13]. In esso viene proposto uno schema di *codifica mediante classificazione* di immagini multispettrali *orientato all'efficienza*. La *classificazione* dei dati avviene a *minima distanza* su un *codebook* noto sia al codificatore che al decodificatore, ottenuto *off-line* su un *training-set* di immagini. Una *trasformata classificata* è applicata quindi lungo la dimensione spettrale sui residui della quantizzazione.

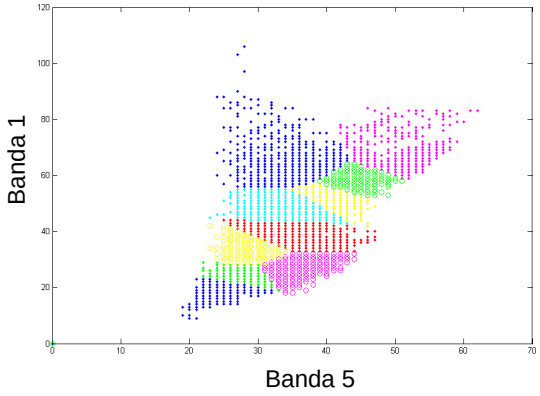


Figura 6.10: Clustering dei dati di fig. 6.9.

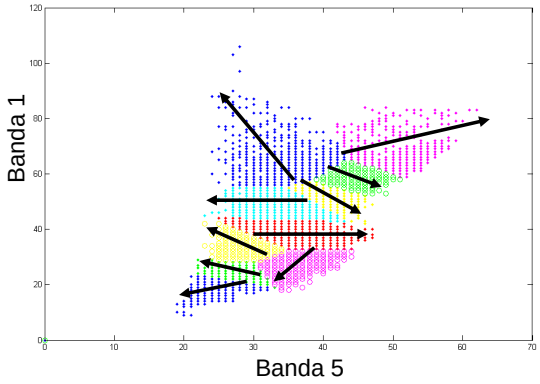


Figura 6.11: Direzioni principali per le classi di fig. 6.11.

Essa è una trasformata ortogonale, la cui matrice cambia a seconda della *classe* ed è ottenuta *off-line* sull'*insieme di addestramento*, come KLT della specifica *classe* considerata. Sui coefficienti trasformati, separati per *classe* e per banda, è effettuata una DCT spaziale monodimensionale.
Lo schema di codifica, per quanto detto, adotta un criterio di *classificazione* trasparente alla *trasformata classificata* e alla trasformata spaziale *orientata alle classi*. Il criterio di segmentazione è comunque finalizzato alla minimizzazione dell'MSE, per cui in qualche modo tiene conto, anche se in maniera *greedy*, delle problematiche di compressione.

In [8] si dimostra sperimentalmente che tecniche di segmentazione *contestuali*, non specificamente orientate alla minimizzazione dell'MSE, non garantiscono solitamente buone *performance* di codifica. Una moderata regolarizzazione può comunque giovare alla codifica efficiente della mappa si segmentazione.

In [15] e poi più approfonditamente [12], lo schema iniziale viene migliorato. La *segmentazione* rimane una semplice VQ. Adesso però sia il classificatore che la conseguente KLT classificata, sono progettati direttamente sulle statistiche dei dati da codificare. Lo schema in questo modo raggiunge prestazioni migliori, del tutto confrontabili con quelle dei più avanzati sistemi di codifica allo stato dell'arte (DWT-3D + SPIHT).

Lo schema di [12] viene successivamente migliorato in [16], adottando diverse tecniche di segmentazione *contestuali* e *non contestuali*. Per le tecniche *contestuali*, che mirano ad ottenere regioni piuttosto connesse mediante opportune strategie di regolarizzazione della mappa, viene introdotta una trasformata spaziale *shape adaptive*. Le prestazioni si dimostrano ottime per tutti gli schemi proposti. La *classificazione* si rivela uno strumento efficace e la *KLT classificata* ne migliora ulteriormente le *performance*.
In fig. 6.13, viene proposto il confronto tra *codificatori classificati* che impiegano, da un lato, tecniche di *segmentazione contestuale* e trasformate *shape adaptive* (codifica **region-adaptive**), dall'altro, strategie basate su *segmentazione non contestuale* e trasformata spaziale monodimensionale (codifica **class-adaptive**). Le due diverse famiglie di tecniche, negli esperimenti presentati, si mostrano alla pari. La *KLT classificata* garantisce migliori risultati di ricostruzione rispetto al caso in cui la trasformata spaziale è *shape adaptive* ma la trasformata spettrale non è classificata (si veda il lavoro [18]).

In [14] viene riproposto, infine, lo schema di [13] sostituendo la DWT-2D *shape adaptive* alla DCT monodimensionale. Le prestazioni migliorano ulteriormente ed il confronto con KLT spettrale + JPEG2000 è a favore dello

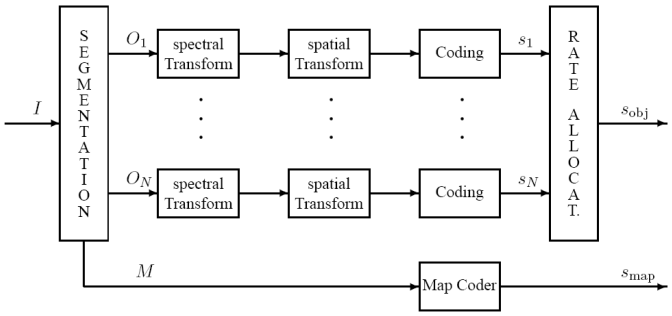
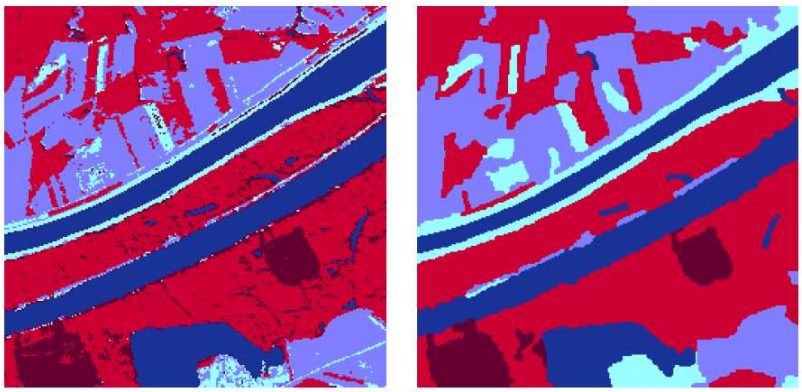


Figura 6.12: Schema generale di un *codificatore classificato* per immagini multispettrali ed iperspettrali.

schema proposto. C'è da dire, che rispetto ad altri schemi di codifica visti in precedenza (compreso quello di [12]), l'allocazione delle risorse è effettuata *a posteriori*, non con tecniche *greedy* oppure analisi *a priori* sicuramente meno efficaci. In definitiva, rispetto a [13], la tecnica di codifica viene migliorata

- dalla strategia di progetto *non supervisionata* per *segmentazione* e *trasformata classificata*,
- dall'*allocazione delle risorse a posteriori*,
- dalla trasformata bidimensionale *shape adaptive*.

In fig. 6.14 è presentato il confronto definitivo tra la tecnica di codifica classificata proposta e KLT+JPEG2000.



(a) (b)

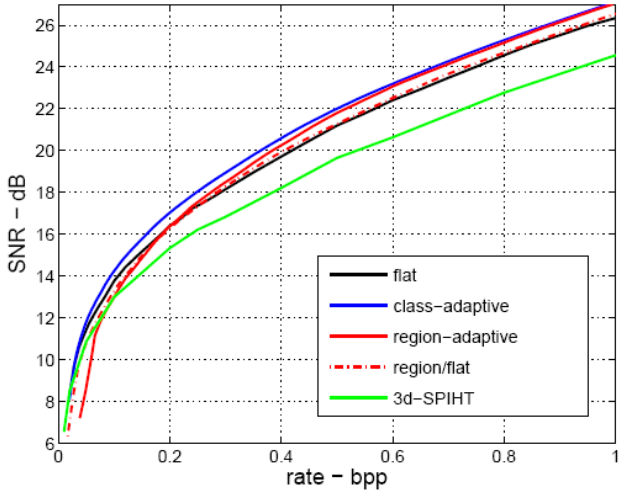
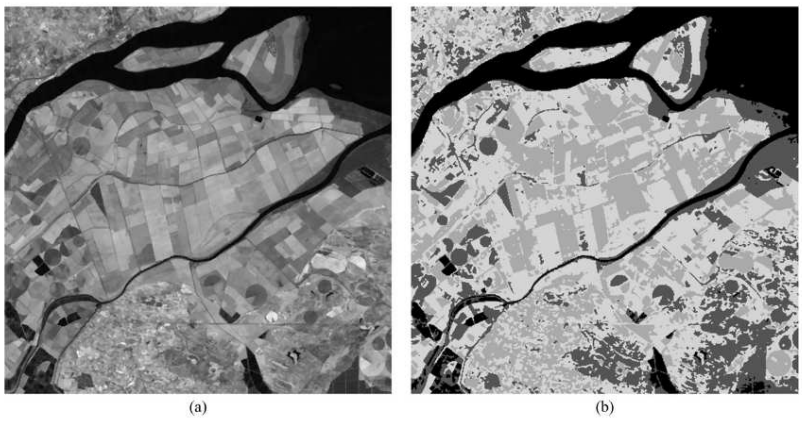


Figura 6.13: a) Mappa di segmentazione *non contestuale* di un'immagine iperspettrale GER. (b) Mappa di segmentazione *contestuale* della stessa immagine. In basso: confronto tra tecniche di codifica mediante classificazione con trasformata spettrale *classificata* e con trasformata spettrale *flat*.



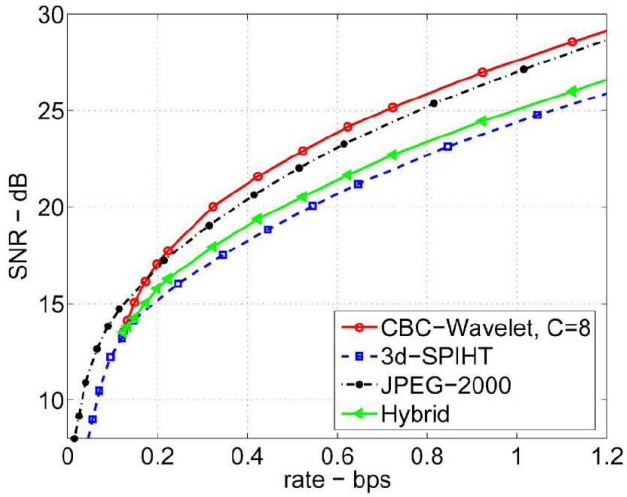


Figura 6.14: (a) Immagine multispettrale LANDSAT. (b) Mappa di segmentazione TSVQ. In basso: prestazioni di una tecnica classificata basata sui più moderni strumenti di codifica, confrontate con quelle di schemi di riferimento allo stato dell'arte, tra cui KLT+JPEG2000.

6.6 Conclusioni

I risultati sperimentali ottenuti sulle immagini multispettrali dimostrano l'efficacia della *codifica mediante classificazione orientata all'efficienza*. Inoltre le *mappe di segmentazione* innestate nel flusso di codifica, costituiscono una funzionalità aggiuntiva di interesse. E' vero infatti che esse rappresentano solo un *clustering* dei dati e non una *classificazione orientata alla semantica* della scena, ma il *clustering* è un'elaborazione intermedia impiegata comunemente da tecniche di segmentazione più complesse, per cui averla già disponibile rappresenta un vantaggio non trascurabile.
Qualche obiezione potrebbe sorgere per la complessità di queste tecniche rispetto ad i codificatori tradizionali. Il problema della complessità viene trattato nel capitolo successivo, come contributo di questo lavoro di tesi allo sviluppo della ricerca sul tema.
Si segnala infine l'importanza teorica del paradigma della *trasformata classificata*, le cui potenzialità applicative sono per ora soltanto intraviste ed il cui studio vale sicuramente la pena di approfondire in lavori futuri.

Bibliografia

[1] M. Sonka, V. Hlavac, R. Boyle, “Image processing, analysis and machine vision”, PWS Publishing, 1999.

[2] C. M. Bishop, “Pattern recognition and machine learning”, Springer, 2006.

[3] L. Torres, M. Kunt, “Video coding, the second generation approach”, Kluwer Academic Publishers, 1996.

[4] J. A. Richards, X. Jia, “Remote sensing digital image analysis, an introduction”, IV Ed. , Springer 2005.

[5] S. German, D. German, “Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 721-741, novembre 1984.

[6] C. D’Elia, G. Poggi, G. Scarpa, “A tree-structured Markov random field model for bayesian image segmentation”, *IEEE Transactions on Image Processing*, vol. 12, pp. 1259-1273, ottobre 2003.

[7] G. Poggi, G. Scarpa, J. Zerubia, “A binary tree-structured mrf model for multispectral satellite image segmentation”, Technical report, *INRIA Sophia Antipolis*, dicembre 2003.

[8] G. Poggi, G. Scarpa, J. Zerubia, “Supervised segmentation of remote-sensing images based on a tree-structured MRF model”, IEEE Transactions on Geoscience and Remote Sensing, vol. 43, n. 8, pp. 1901-1911, agosto 2005.

[9] L. Cicala, G. Poggi, G. Scarpa, “Supervised segmentation of remote-sensing multitemporal images based on the tree-structured markov random field model”, *IEEE International Geoscience and Remote Sensing Symposium*, vol. 3, pp. 1569-1572, Anchorage (USA), settembre 2004.

[10] S. Li, W. Li, “Shape adaptive dicrete wavelet transforms for arbitrarely shaped visual object coding”. IEEE Trans. on Circuits and System for Video Technology, pp. 725-743, agosto 2000.

[11] M. Cagnazzo, S. Parrilli, G. Poggi, L. Verdoliva, “Costs and advantages of shape-adaptive wavelet transform for region-based image coding”, *EURASIP Journal on Image and Video Processing*, 2007.

[12] M. Cagnazzo, L. Cicala, G. Poggi, L. Verdoliva, “Low-complexity compression of multispectral images based on classified transform coding”, *Elsevier Signal Processing: Image Communication*, vol. 21, n. 10, pp. 850-861, novembre 2006.

[13] G. Gelli. G. Poggi, “Compression of multispectral images by spectral classification and transform coding” *IEEE Transactions on Image Processing*, 8:476-489, April 1999.

[14] M. Cagnazzo, S. Parrilli, G. Poggi, L. Verdoliva, “Improved class-based coding of multispectral images with shape-adaptive wavelet transform”, IEEE Geoscience and Remote Sensing Letters vol. 4, pp. 566-570, ottobre 2007.

[15] M. Cagnazzo, L. Cicala, G. Poggi, G. Scarpa, L. Verdoliva, “An unsupervised segmentation-based coder for multispectral images.” *European Signal Processing Conference* (EUSIPCO), settembre 2005.

[16] M. Cagnazzo, R. Gaetano, S. Parrilli, and L. Verdoliva. “Region based compression of multispectral images by classified klt”, *European Signal Processing Conference* (EUSIPCO), Firenze, settembre 2006.

[17] G. Poggi and A. R. P. Ragozini. “Image segmentation by tree-structured Markov random fields”, *IEEE Signal Processing Letters*, vol. 6, pp. 155-157, giugno 1999.

[18] M. Cagnazzo, R. Gaetano, S. Parrilli, and L. Verdoliva. “Adaptive region-based compression of multispectral images.” *IEEE International Conference on Image Processing* (ICIP), Atlanta, GA (USA), ottobre 2006.

Capitolo 7

Trasformata classificata multispettrale

Parole chiave

Nozioni propedeutiche

Clustering, classificazione, codifica mediante classificazione, trasformata classificata, TSVQ, KLT, DCT, quantizzazione scalare, DWT, SPIHT, curve tasso-distorsione, matrici di confusione.

Concetti introdotti nel capitolo

CBC.

Concetti innovativi illustrati nel capitolo

U-CBC.

7.1 Introduzione

I sensori satellitari hanno sempre una più alta risoluzione spaziale, spettrale e radiometrica. Con tale ricchezza di informazioni sorge il problema dover gestire volumi molto grandi di dati, in ogni stadio della loro elaborazione. La fase più critica è sul satellite, dove i dati acquisiti superano facilmente la capacità di *downlink* del canale di trasmissione, e spesso gran parte delle immagini deve essere semplicemente scartata, ma problemi analoghi sorgono anche nel segmento di terra, dove l'archiviazione e la distribuzione dell'immagine sono seriamente compromesse dalla quantità di dati da amministrare. Per evitare questi problemi si può ricorrere alla *compressione* dati, che permette di ridurre il volume di dati di uno o perfino due ordini di grandezza senza seri effetti sulla qualità dell' immagine al fine delle elaborazioni successive.

In tal senso, comunque, non si possono utilizzare tecniche di impiego generico perchè esse non sono in grado di sfruttare adeguatamente le peculiarità delle immagini multispettrali telerilevate, e per questo motivo negli ultimi anni sono stati proposti vari schemi di codifica ad hoc. L'approccio più diffuso è quello della *codifica mediante trasformata*, per varie ragioni. Infatti, le tecniche di codifica tramite trasformata sono ben radicate e profondamente comprese, hanno eccellenti prestazioni nella *compressione* di immagini, video, e altre sorgenti, hanno una ragionevole complessità e, non ultima motivazione, sono il nucleo di noti ed efficienti standard come ad esempio JPEG e JPEG2000, largamente usati e disponibili alla comunità scientifica in diverse implementazioni [9].
Un approccio di comune impiego per la codifica di immagini multispettrali [10, 11] è utilizzare trasformate decorrelanti lungo la dimensione spettrale, seguite da JPEG2000 sulle bande trasformate. Soluzioni alternative includono l'utilizzo della trasformata di wavelet (WT) seguita da SPIHT [12, **?**], schemi basati sulla più tradizionale trasformata coseno-discreta (DCT) [14, 15],o su altre trasformate orientate all'applicazione, e.g., [16].
Minor attenzione è stata rivolta alle tecniche basate sulla *quantizzazione vettoriale* (VQ) perchè, a dispetto della loro ottimalità teorica [17], la VQ è troppo onerosa computazionalmente per essere di qualsiasi uso pratico. Nondimeno quando si ha a che fare con immagini multicanale, la VQ è un candidato naturale, perchè l'unità semantica elementare di tali immagini è il vettore di *risposta spettrale* (o concisamente "*spettro*") che rappresenta la risposta dell'immagine, in una determinata collocazione spaziale, su tutte le bande spettrali. Il valori dello spettro a bande differenti non sono semplicemente correlati ma fortemente dipendenti, perchè sono univocamente determinati (ad ec-

cezione di un certo rumore sovrapposto) fissata la natura del terreno della cella ripresa. Questa osservazione ha spinto la ricerca verso le tecniche di *quantizzazione vettoriale* vincolata [18, 19, 20], che sono sub-ottime ma anche ben più semplici della VQ a ricerca esaustiva, ed hanno prestazioni incoraggianti.

In seguito, esamineremo con attenzione sullo schema di codifica ibrida originariamente proposto in [4], dove la VQ è usata esclusivamente solo nella prima e più critica fase di codifica, per essere seguita più tardi da più semplici tecniche di trasformata, al fine di codificare i residui.

In questo approccio la VQ, oltre a costituire un primo passo di codifica, ha il ruolo centrale di segmentare l'immagine in classi omogenee: i residui VQ sono quindi raggruppati secondo la loro *classe* di appartenenza, cosicché le successive tecniche di *codifica mediante trasformata* possano operare separatamente su ogni *classe*, ed adattarsi alle specifiche statistiche della *classe* considerata.

Grazie all'uso congiunto della VQ, che si serve pienamente delle forti dipendenze interbanda delle immagini multispettrali, e delle trasformate classificate, che concentrano in pochi coefficienti la maggior parte dell'energia dei residui, questo schema di codifica (chiamato da ora in poi CBC, che sta per **Class-Based Coder**) raggiunge ottime prestazioni.

Va sottolineato che questo è un codificatore che indicheremo come di tipo *supervisionato*, cioè che sfrutta per i modelli di codifica (ad esempio il classificatore VQ e le matrici di co-varianza per la trasformata classificata), informazioni note a priori. Tali informazioni sono ricavate a partire da un database molto ampio, che copre virtualmente tutti i tipi di terreno di possibile interesse, in tutte le condizioni atmosferiche e di illuminazione, e sono computate prima dell'effettiva fase di codifica ed immagazzinate a terra e a bordo. In questo modo si da un lato si evita il computo a bordo delle statistiche necessarie ai modelli, essendo esso oneroso e difficilmente effettuabile sotto specifiche di real time, dall'altro non vi è bisogno di inviare le stesse al segmento di terra, risparmiando il conseguente ingombro informazionale.
Fino a pochi anni fa tale accorgimento era senz'altro ragionevole, ma i costanti aumenti di potenza di calcolo ora aprono le porte al computo on board e real time delle statistiche.

Verificate le interessanti prestazioni del CBC, e le opportunità offerte dall'approccio basato sulla *classificazione*, in questo capitolo svilupperemo la problematica della modifica del codice originario così da renderlo pienamente

non supervisionato e adatto all'implementazione a bordo. Come già accennato, lo schema *non supervisionato*, comporta che i parametri di codifica debbano essere calcolati a bordo al momento della *compressione* ed inviati come *informazione* collaterale assieme ai dati codificati. Nel passaggio dallo schema *supervisionato* a quello *non supervisionato*, sono due dunque le criticità:

1. la complessità congiunta del progetto dei modelli e della codifica dei dati deve essere tale da poter permettere l'implementazione real time,

2. l'*informazione* associata ai modelli di codifica deve essere molto limitata, tanto da non deteriorare significativamente le prestazioni in termini di tasso-distorsione.

Tenendo in conto sia dei vincoli di complessità che di *informazione* collaterale, introdurremo un nuovo codificatore classificato per immagini multispettrali completamente *non supervisionato* che garantisce una bassa complessità computazionale e prestazioni altamente competitive.
Sebbene il progetto di un vero e proprio codificatore in tempo reale sia al di fuori dello scopo di questo studio, l'analisi sperimentale mostra che la complessità generale di codifica è chiaramente all'interno delle possibilità dei moderni processori [21]. Inoltre, esperimenti sulle immagini multispettrali dimostrano che, anche in termini di prestazioni tasso-distorsione, il nuovo codificatore *non supervisionato* è superiore all'originario CBC, essendo del tutto comparabile alle tecniche di stato dell'arte come 3d-SPIHT [13].

Il prossimo paragrafo riesamina il CBC, evidenziando sia i punti di forza che quelli di debolezza per le operazioni in tempo reale, e stimando attentamente la complessità di codifica. Il paragrafo 7.3 descrive nel dettaglio le variazioni rispetto al codice originario, stima la complessità di progetto dei modelli e valuta il peso delle informazioni collaterali. Il paragrafo 7.4 è rivolta all'analisi sperimentale di alcune immagini telerilevate. In primo luogo, la procedura semplificata del progetto e la tecnica di codifica delle informazioni collaterali sarà convalidata, quindi, le prestazioni tasso-distorsione della tecnica proposta saranno comparate con quelle del CBC originale e di un codificatore basato sul wavelet allo stato dell'arte. Infine, nella sezione 7.5 traiamo le conclusioni sul lavoro presentato.

7.2 Il codificatore supervisionato originale

In questa sezione anzitutto descriviamo i dettagli dello schema di codifica CBC, quindi passiamo ad un'accurata valutazione della sua complessità di codifica in relazione all'impiego in applicazioni real time.

7.2.1 Schema di codifica

Il CBC effettua tre principali operazioni, come sintetizzato nella Fig. 7.2.1:

1. *segmentazione* dell'immagine;

2. codifica della *mappa di segmentazione* senza perdita di *informazione*;

3. codifica della tessitura con perdita di *informazione*.

La *segmentazione* consiste in un semplice clustering spettrale: ogni *classe* è rappresentata da un unico spettro template e ciascun pixel (inteso come vettore dei pixel omologhi di ciascuna banda spettrale) è associato ad una *classe* in accordo ad un criterio di minima distanza. Le associazioni indice di *classe* - spettro costituiscono un dizionario VQ, mentre la *segmentazione* stessa può essere vista come una *quantizzazione vettoriale*. Il passo di *segmentazione* appena descritto può anche essere considerato come un primo livello di *compressione* dell'immagine, a cui seguirà la *codifica mediante trasformata* dei residui.

Il principale obiettivo della *segmentazione*, ad ogni modo, è quello di classificare i pixel dell'immagine in base al loro spettro, così da raggrupparli e permettere la codifica congiunta dei residui VQ appartenenti alla stessa *classe*.

Ogni insieme di residui, infatti, mostra statistiche omogenee e può essere compresso molto efficientemente impiegando trasformate convenzionali [1]. Nel CBC una Trasformata di Karhunen-Loeve (KLT) è applicata lungo la dimensione spettrale impiegando una differente matrice di trasformazione per ogni *classe*. Quindi, dalle bande trasformate si ricavano varie sequenze di coefficienti, una per ogni banda e per ogni *classe*, sulle quali viene ulteriormente

[1] Algoritmi più sofisticati possono produrre *mappe di segmentazione* più connesse rispetto alla VQ, ma conducono eventualmente a peggiori prestazioni sotto tutti i punti di vista. Nel rispetto di ciò, infatti, la VQ rappresenta la scelta ottimale, siccome è progettata appunto per ottimizzare le prestazioni tasso-distorsione, e cioè per minimizzare l'energia dei residui ad un tasso di codifica assegnato.

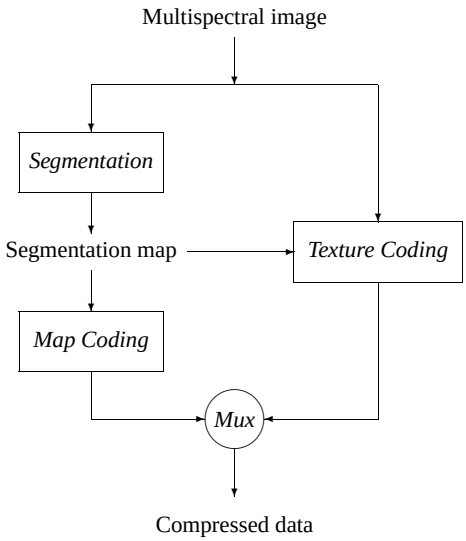


Figura 7.1: Schema di principio della codifica mediante classificazione.

effettuata una Trasformata Coseno Discreta (DCT) che elimina le rimanenti ridondanze spaziali. Per massimizzare la correlazione dei coefficienti, essi vengono raggruppati secondo una scansione non riga per riga, ma lungo una curva di Peano. In fine, ogni coefficiente della trasformata è memorizzato in una **classe di quantizzazione** indicizzata dalla classe spettrale, dalla banda KLT e dalla frequenza DCT. A ciascuna specifica classe di quantizzazione è assegnato uno quantizzatore Lloyd-Max opportunamente progettato off-line partendo dalle statistiche delle immagini afferenti alla base di dati di riferimento. Nella fase di codifica a ciascun quantizzatore è assegnata una certa risoluzione, stabilita da un algoritmo di allocazione miope delle risorse. Ovviamente, la *mappa di segmentazione*, che è un insieme degli indici VQ, deve essere trasmessa allo stesso modo della codifica dei residui. Dato che i pixel adiacenti sono altamente correlati, la mappa può essere significativamente compattata, ricorrendo ad uno schema predittivo seguito dalla *codifica di Huffman*. Poichè lo schema di codifica è *supervisionato* il dizionario VQ e le matrici di covarianza non devono essere inviate in quanto già memorizzate nel decodificatore.

Gli esperimenti mostrano che il codificatore classificato così strutturato garantisce un miglioramento di 2-3 dB a tutti i tassi di interesse nel rispet-

to allo stesso codificatore non classificato (lo stesso codificatore con una singola *classe* onnicomprensiva), nonostante il costo aggiuntivo richiesto per la trasmissione della *mappa di segmentazione*. In aggiunta, la mappa di *segmentazione* stessa è un prezioso pezzo dell'*informazione*, ottenuto dai dati originali non compressi ed automaticamente innestata sul flusso di codifica, che contribuisce a rendere questo schema di codifica un interessante strumento per la *compressione* delle immagini telerilevate. Questo rapido esame del CBC (il lettore faccia riferimento alla nota[4] per ulteriori dettagli) ci ha permesso di comprenderne le fasi principali, cioè, la *classificazione* VQ, la trasformata spettrale KLT, la trasformata spaziale e la quantizzazione.

Nello schema originale, i dati statistici necessari al progetto (classificatore VQ, matrici KLT, quantizzatori scalari) sono già disponibili al decodificatore, ma nella versione *non supervisionata* devono essere ricavati on-line dai dati da codificare, aumentando la complessità di calcolo, e devono essere inviati come side-infomation, pesando sul *tasso di codifica*.
Questa nuova **f**ase di progetto sarà descritta approfonditamente nella prossima sezione, ma prima di ciò, sarà utile valutare la complessità della **f**ase di codifica, in modo da avere un riferimento significativo della complessità e raggiungere una maggiore comprensione del funzionamento del codificatore.

7.2.2 Stima della complessità di codifica

Consideriamo un immagine multisprettrale con N pixels (il numero esatto di linee e colonne è irrilevante per ora) e B bande spettrali. Per ogni pixel, il segmentatore deve trovare la minima distanza spettrale tra i C modelli spettrali disponibili, dove per C si intende il numero di classi. Con una ricerca esaustiva VQ, questa operazione richiede BC moltiplicazioni per ogni spettro, o C mps (moltiplicazioni per campione, unità convenzionali della misura per complessità). Anche se solo piccoli dizionari potranno essere usati (diciamo, $C < 32$), questo è un livello non trascurabile di complessità. Il CBC ricorre, dunque, a struttura ad albero VQ (TSVQ) [17], C che riduce la complessità a circa $\log_2 C$ mps (il valore esatto non è predicibile e dipende dalla specifica codifica) con un effetto trascurabile sull'accuratezza della *segmentazione* Nella TSVQ, infatti, le classi sono organizzate in un albero binario, e la *classe* desiderata è ottenuta da una successione di scelte binarie, ognuna delle quali richiede il computo di un prodotto scalare tra vettori. [2]

[2]Esistono un gran numero di tecniche di VQ veloce, che potrebbero ulteriormente ridurre la complessità, ma il loro approfondimento va oltre i propositi di questa trattazione.

Dopo la VQ, ogni vettore residuo subisce una trasformazione KL, e questo richiede B^2 moltiplicazioni per vettore, cioè B mps, piuttosto poche per una tipica immagine multispettrale. La complessità può diventare rilevante per grandi valori di B (cioè per immagini iperspettrale), In questo caso tuttavia, solo una piccola frazione $B' \ll B$ delle bande trasformate contiene un'*informazione* utile, per cui si può usare una matrice trasformata rettangolare $B \times B'$, con una conseguente complessità inferiore a B' mps.

A valle della KLT spettrale viene applicata una trasformata coseno discreta che, per una sequenza di lunghezza K, ha una complessità di $\frac{1}{2}K \log_2 K$ moltiplicazioni. Visto che tutti i campioni devono essere trasformati, a prescindere dalla *classe* e dalla banda, vanno eseguite circa NB/K DCT, per cui la complessità normalizzata (su cassi e bande) è soltanto $\frac{1}{2}\log_2 K$ mps, non particolarmente elevata per un tipico valore di K.

L'ultimo passo è la *quantizzazione scalare*, la cui complessità può essere tralasciata. Infatti, essa richiede soltanto confronti, il cui numero esatto dipende dal tasso assegnato allo specifico insieme da quantizzare e dal *tasso di codifica* totale. Come limite superiore, il numero dei confronti deve essere inferiore al numero totale di bit inviati (perchè alcuni bit sono invece assegnati alla codifica della mappa) ed è dunque dell'ordine di un bit per campione, quantità decisamente alta per la codifica di immagini multispettrali.

Sommando tutti questi risultati, la complessità della fase di codifica può essere stimata come

$$Q_{\text{coding}} \simeq \log_2 C + B + \frac{1}{2}\log_2 K \tag{7.1}$$

Di certo, questa è soltanto una stima approssimata della complessità, siccome molte fasi che impegnano la CPU, come il trasferimento di dati etc., sono completamente trascurate, ma ci consente di studiare in quali condizioni il CBC è utilizzabile nelle implementazioni a bordo.

In primo luogo una prima specifica da soddisfare è

$$Q_{\text{coding}} < P_{\text{CPU}}/R_{\text{aq}} \tag{7.2}$$

dove P_{CPU} è la potenza della CPU (in moltiplicazioni al secondo), e R_{aq} il tasso di acquisizione dei dati (in campioni al secondo). Supponendo, ad esempio, un'unità di calcolo di 2.5 Gflops [21], e un tipico tasso di acquisizione dati di 10^8 campioni al secondo, il codificatore dovrebbe essere già capace di funzionare fluidamente. In aggiunta, il CBC si presta naturalmente al calcolo parallelo, perchè dopo la VQ, che lavora contemporaneamente su tutti i dati, C

elaborazioni possono funzionare in parallelo, una per ciascuna *classe*. Quindi, se è prevista un'implementazione parallela, il vincolo di complessità è di gran lunga minore ed è indubbiamente possibile un'implementazione in tempo reale.

7.3 Schema on line non supervisionato

Per realizzare una versione del CBC completamente *non supervisionata* dobbiamo semplicemente progettare on-line il classificatore VQ, le matrici KLT, ed i quantizzatori scalari. D'altra parte il risultante incremento di complessità e le informazioni collaterali devono essere attentamente controllati al fine di mantenere il coder efficiente e realizzabile. Comunque, poichè tutte queste componenti di *informazione* sono ricavate dagli stessi dati da codificare, ci si aspetta un miglioramento prestazionale che potrebbe bilanciare o persino superare la riduzione del tasso dovuta alle informazioni collaterali.

7.3.1 Il classificatore VQ

Il progetto di un dizionario VQ può essere davvero oneroso in termini di potenza computazionale ma, siccome soltanto un limitato numero di tipi di terreno sono comunemente presenti in una data immagine, basterà ricavare un dizionario piuttosto piccolo. Inoltre, poichè i nostri classificatori sono strutturati ad albero, non bisogna effettivamente progettare un dizionario di dimensione C, ma piuttosto $C-1$ dizionari di dimensione 2, che sono molto meno onerosi da calcolare. Infine, il progetto non necessariamente deve essere effettuato su tutti i dati da codificare, piuttosto può avvenire su un insieme di addestramento di dimensione adeguata M_{VQ}. Si propone dunque di effettuare il progetto del classificatore VQ mediante il celebre algoritmo generalizzato di Lloyd(GLA)[17], che richiede poche interazioni, durante le quali ogni vettore d'addestramento è codificato dal dizionario corrente. Se supponiamo, per semplicità, che l'albero di *classificazione* sia bilanciato, che i vettori d'addestramento siano regolarmente ripartiti tra ramo destro e sinistro ad ogni nodo, e che il GLA richieda non più di I_{VQ} iterazioni per convergere, la complessità può essere stimata come $M_{\mathrm{VQ}}I_{\mathrm{VQ}}B$ moltiplicazioni per il progetto di un dizionario base, più altri $2\times(M_{\mathrm{VQ}}/2)I_{\mathrm{VQ}}B$ per il progetto dei due dizionari al livello uno dell'albero, e così via fino alla fine dell'albero, per un totale di $M_{\mathrm{VQ}}I_{\mathrm{VQ}}B\log_2 C$ moltiplicazioni o $(M_{\mathrm{VQ}}/N)I_{\mathrm{VQ}}\log_2 C$ mps.
Con $M_{\mathrm{VQ}}=N$ il costo complessivo potrebbe essere significativo, ma attraverso un sottocampionamento non eccessivamente spinto, impiegando cioè un insieme di addestramento sufficientemente grande, la complessità di progetto può essere portata ben al di sotto di quella di codifica. Per esempio, assumendo abbastanza prudentemente $I_{\mathrm{VQ}}=10$, $C=32$, $M_{\mathrm{VQ}}=100\times C$, e $N=512\times 512$ si può raggiungere una complessità totalmente gestibile di 1 mps.

Per quel che riguarda l'*informazione* collaterale, il dizionario VQ conta C vettori [3], di B componenti ciascuno. Per qualsiasi ragionevole combinazione di parametri (dimensioni dell'immagine, etc.) il costo di codifica di un siffatto dizionario è trascurabile anche impiegando 16 bit per ciascuna componente (e può essere ulteriormente ridotto se in qualche modo è presa in considerazione la correlazione inter-banda), per cui esso non deteriora le prestazioni. Al contrario, un buon dizionario progettato on-line può essere decisamente più efficace della sua versione off-line, in quanto nel secondo caso non è garantito che l'insieme di addestramento rappresenti adeguatamente i dati reali. Dal progetto on-line del classificatore VQ ci si aspetta per questo motivo un miglioramento delle prestazioni.

7.3.2 Le matrici KLT adattate alla classe

Per calcolare le matrici KL lungo la direzione spettrale, dobbiamo prima stimare matrici di correlazione di dimensione $B \times B$ sui dati, e dunque calcolarne gli autovettori.

Dal momento che la trasformata applicata è orientata alle classi, abbiamo bisogno di calcolare C matrici di covarianza, una per ciascuna *classe*. Per il calcolo di esse ricorriamo nuovamente al sottocampionamento dei dati. Con un insieme di addestramento di M_{KLT} vettori, bisogna effettuare $M_{\mathrm{KLT}}B(B+1)/2$ moltiplicazioni, cioè $(M_{\mathrm{KLT}}/N)(B+1)/2$ mps, che non è significativo per un training set di dimensione ragionevolmente contenuta, ad esempio $M_{\mathrm{KLT}} = 100C$.

Calcolare gli autovettori con una procedura di inversione standard (tridiagonalizzazione/Jacobi), ha invece una complessità di circa $2B^2(B+1)I_{\mathrm{KLT}}$ per ciascuna matrice KLT [23], dove I_{KLT} è il numero di iterazioni richieste, per un totale di $2B(B+1)CI_{\mathrm{KLT}}/N$ mps.

Per le immagini multispettrali, dove B è piuttosto contenuto, tale calcolo è sicuramente di complessità accettabile; con $C = 32$, $N = 512 \times 512$, e $I_{\mathrm{KLT}} = 10$ ad esempio, la complessità rimane sotto 1 mps anche per un immagine di 20 bande spettrali. Per le immagini iperspettrali, d'altra parte, poichè sono richiesti ragionevolmente solo $B' \ll B$ autovettori dominanti, si possono impiegare tecniche più veloci, come il *power method* [23], in modo da tenere la complessità sotto controllo.

Riguardo l'*informazione* collaterale, per ciascuna delle C matrici KLT è

[3] I vettori TSVQ intermedi non sono richiesti dal ricevitore.

necessario inviare $B(B+1)/2$ parametri, il cui tasso è significativo solo in condizioni eccezionali (immagini piccole, basso tasso, molte classi, molte bande). Evitando questi casi e facendo attenzione a codificare i parametri con il numero minore di bit possibile, l'effetto negativo sul tasso complessivo non è significativo. Ancora una volta è da considerarsi un aspetto decisamente positivo per le prestazioni tasso-distorsione il fatto che le statistiche sono calcolate sui dati stessi da codificare.

7.3.3 I quantizzatori parametrici adattativi

Ora dobbiamo distaccarci decisamente dallo schema di codifica originale, non solo per la complessità di progetto ma soprattutto perchè l'*informazione* collaterale da trasmettere nel caso di quantizzatori di Lloyd-Max, è ovviamente inaccettabile. Infatti nello schema originale, un quantizzatore ad hoc è impiegato per il primo coefficiente DCT della prima autobanda KLT della prima *classe* e così via fino a raggiungere un totale di CBK quantizzatori. Anche considerando che alla maggioranza degli insiemi non viene assegnato nessun bit per la codifica, per cui non vi è bisogno di inviare il modello del quantizzatore, rimangono ancora così tanti quantizzatori da progettare e da inviare, che questo approccio, pur avendo senso nel contesto di uno schema *supervisionato*, diventa del tutto inappropriato per la nuova versione *non supervisionata* per applicazioni real-time.

Per abbattere il costo della side-information bisogna dunque ricorrere a quantizzatori parametrici: ciascun insieme di coefficienti è modellato a seconda come un processo gaussiano (per le basse frequenze) o laplaciano (per le alte frequenze), e caratterizzato con la sua varianza, la quale univocamente determina il quantizzatore parametrico corrispondente. Per preservare la scalabilità dello schema originale, i quantizzatori sono ancora una volta strutturati ad albero. Inoltre essi sono mid-tread (centrati), cioè, per questioni di robustezza, in ciascun livello, oltre ad i tradizionali nodi binari, vi è un nodo centrale ternario, così come illustrato in fig. 7.3.3. Le varianze dei coefficienti sono impiegate per assegnare le risorse ai quantizzatori così previsto dall'algoritmo di Huang-Schultheiss [17].

La complessità del progetto risiede unicamente nella stima delle varianze. Senza sottocampionamento, tutti i campioni compaiono nella stima di un solo parametro, per cui la complessità è solo di 1 mps. Schemi di sottocampionamento possono abbattere ulteriormente questo costo, che chiaramente non costituisce un serio problema per la complessità.

L'*informazione* collaterale è rappresentata da ciascuna varianza, per un to-

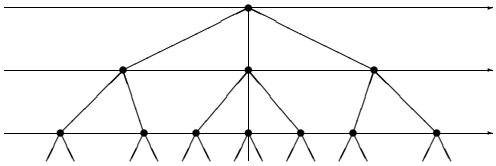


Figura 7.2: Quantizzatore scalare mid-tread strutturato ad albero.

tale di CBK parametri da inviare, apparentemente molto di più che per le matrici KLT. Tuttavia, come già accennato prima, una grande quantità di questi insiemi ha bassissima varianza, per cui ad essi non è assegnato alcun bit per la codifica.
Si può dunque pensare di inviare due informazioni: un flag binario che indica se la varianza deve essere rappresentata o meno (a seconda che all'insieme siano o no assegnati bit), ed un certo numero di bit per l'eventuale rappresentazione della varianza. Il costo complessivo è dunque pari a $CBK(1+\alpha(R)\beta)$ dove la frazione di bit attivi $\alpha(R)$ è piccola per bassi bit-rate R, semplificando i possibili problemi, e β indica il numero di bit impiegati per codificare il parametro.

In definitiva la complessità totale del progetto è stimabile come

$$Q_{\mathrm{design}} \simeq \frac{M_{\mathrm{VQ}}}{N} I_{\mathrm{VQ}} \log_2 C + \frac{M_{\mathrm{KLT}}}{N}(B+1)/2 + \frac{2}{N} C I_{\mathrm{KLT}} B(B+1) + 1 \tag{7.3}$$

La complessità relativa del progetto rispetto alla codifica, per quanto detto, è regolata dalle equazioni 7.1 e 7.3. Supponendo che il training set impiegato nella fase di progetto sia molto più piccolo dell'intera immagine, e che $I_{\mathrm{VQ}} < 10$, come sempre è verificato nei nostri esperimenti, il primo ed il secondo termine nell'espressione di Q_{design} sono trascurabili rispetto ad i rispettivi termini presenti nell'espressione di Q_{coding}. In più il contributo relativo alla quantizzazione scalare è sempre ininfluente. Allo stesso modo il terzo termine dell'espressione di Q_{design}, è anch'esso trascurabile, fatta eccezione per il caso in cui il numero delle bande è confrontabile con il numero di pixel, come avviene nel caso iperspettrale. In questa evenienza, comunque, solo una matrice di trasformazione $B' \ll B$, come già accennato, deve essere calcolata, e l'equazione 7.3 viene modificata in modo che la complessità sia ancora perfettamente gestibile. In conclusione, se la complessità di progetto possa essere o meno considerata trascurabile, dipende essenzialmente dalle dimen-

sioni del training set sul quale sono calcolati il dizionario VQ e le matrici di trasformazione KL.

7.4 Analisi sperimentale

In questa sezione, ci interroghiamo sulla reale complessità e sulle vere prestazioni dello schema di codifica proposto, attraverso i risultati degli esperimenti numerici su immagini telerilevate acquisiti da sensori aerei o satellitari, con lo scopo di dimostrare che

- la complessità aggiuntiva richiesta dalla nuova fase di progetto è praticamente trascurabile;
- il nuovo codificatore è in grado di lavorare in tempo reale e a bordo di aerei o satelliti;
- le prestazioni in termini di tasso-distorsione sono anche superiori a quelle del codificatore originale.

I test iniziali sono stati condotti sulla stessa immagine usata in [4] in cui CBC viene per la prima volta presentato, un' immagine multispettrale Landsat TM a 6 bande, di una regione rurale vicino Lisbona in Portogallo, in modo da poter confrontare direttamente i risultati.

Esperimenti simili sono stati eseguiti su altre immagini campione. In questo capitolo riporteremo i risultati ottenuti su un'immagine multispettrale IKONOS a 4 bande dell'area di San Diego (California, USA), che ha caratteristiche che sono marcatamente differenti da quelle della prima immagine, ed infine su un segmento di 32 bande di un'immagine iperspettrale GER, di una regione sulle rive del Reno in Germania.

7.4.1 Complessità

In precedenza, abbiamo visto che la complessità del progetto dipende essenzialmente dalla dimensione degli insieme dei dati su cui operano gli algoritmi di apprendimento statistico. Dunque, eseguiamo alcuni esperimenti per stabilire quale tasso di sottocampionamento può essere considerato accettabile per il progetto di classificatori VQ, trasformata KLT e quantizzatori scalari. Più precisamente, per diverse cardinalità dell'insieme delle classi C e per diversi tassi di codifica, abbiamo realizzato il progetto completo su insiemi di dati campione sempre più piccoli ed, effettuata la codifica, abbiamo misurato il conseguente degrado delle prestazioni. Le Fig.7.4.1 e Fig.7.4.1 mostrano i risultati di due esperimenti di questo tipo condotti su un'immagine TM, con 20 classi, e tassi di codifica di 0.25 e 0.5 bit/sample. Sulle ascisse è riportato il tasso di sottocampionamento , che va da 0 (nessun sottocampionamento) a

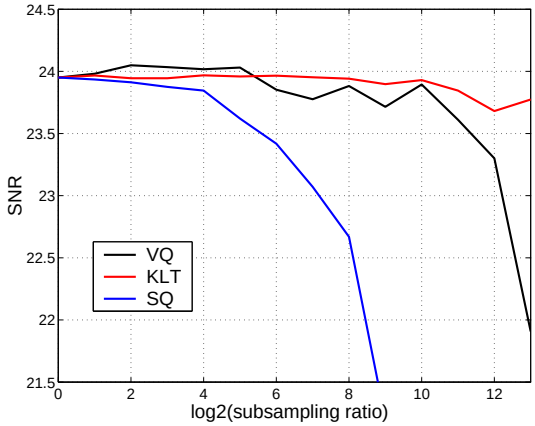


Figura 7.3: Effetto del sottocampionamento a 0.25 bit/campione. Il rapporto di sottocampionamento va da 1:1 to 1:8192.

13 (solo un campione su 8192 è incluso nell'insieme di apprendimento). Le ordinate riportano il rapporto al segnale/rumore (SNR) [4] ottenuto quando la sola grandezza d'interesse è progettata su un insieme sottocampionato mentre le altre sono progettate sull'intero insieme dei dati.

Osserviamo un comportamento comune in questi due casi così come in altri qui non riportati. Riducendo le dimensioni dell'insieme di addestramento per la VQ, l'effetto sulle prestazioni è trascurabile fino a $M_{\mathrm{VQ}} = 256$ (che corrisponde ad un rapporto di sottocampionamento di 1:1024 è cioè log2(sub)=10), un valore molto più piccolo di quello da noi stabilito prudentemente in fase di analisi. Da questo punto in poi, le prestazioni iniziano significativamente a calare, anche perchè la TSVQ ha delle difficoltà nell' identificazione di 20 classi distinte nell'insieme di addestramento.
La KLT sembra essere anche più robusta al sottocampionamento con prestazioni stabili (eccetto per alcune ovvie oscillazioni casuali) in un più ampio intervallo.
Le cose sono decisamente differenti per i quantizzatori scalari, dove anche un moderato sottocampionamento causa una netta perdita. Ciò accade sia perchè i dati disponibili sono in questo caso più scarsi, sia perchè questo è l'ultimo

[4] In tutti gli esperimenti su questa immagine, abbiamo seguito le convenzioni adoperate in [4], cioè le bande sono tutte normalizzate a potenza unitaria e l'SNR è definito come $10\log_{10}(1/\mathrm{MSE})$, dove MSE è l'errore quadratico medio. Per le altre immagini invece useremo la definizione più diffusa sostituendo alla potenza del segnale la varianza.

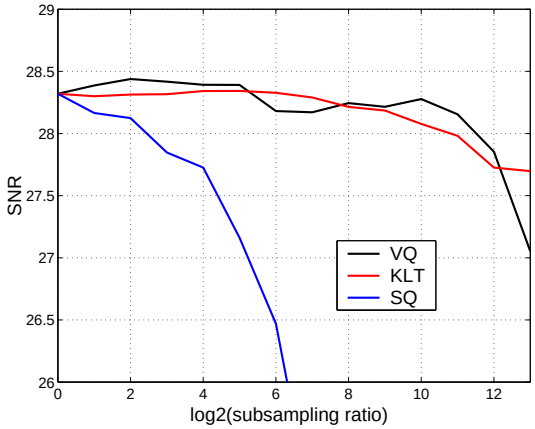


Figura 7.4: Effetto del sottocampionamento a 0.50 bit/campione. Il rapporto di sottocampionamento va da 1:1 to 1:8192.

passo dell'elaborazione, e gli errori in questa fase non possono essere recuperati dalle elaborazioni successive. Tuttavia, ricordiamo che la SQ ha una complessità così limitata che il sottocampionamento non è indispensabile.
Sebbene gli effetti congiunti qui non siano analizzati, gli esperimenti mostrano che un moderato tasso di sottocampionamento sia per la VQ che per la KLT, ad esempio prevedendo un insieme di addestramento di $100 \times C$ campioni, comporta un deterioramento delle prestazioni del tutto trascurabile. Per questo motivo il suddetto criterio di sottocampionamento sarà d'ora in avanti quello adottato nei nostri esperimenti.

In Fig. 7.4.1 il grafico riporta la complessità del progetto (linee continue) e della codifica (linee piane) per un'immagine di 512×512 pixel in funzione del numero di bande B e di classi C. Le curve sono calcolate attraverso le equazioni (7.3) e (7.1), considerando $I_{\mathrm{VQ}} = I_{\mathrm{KLT}} = 10$ e $K = 64$, e supponendo che $M_{\mathrm{VQ}} = M_{\mathrm{KLT}} = 100C$. In tutte le situazioni la complessità del progetto è uno o due ordini di grandezza più piccola della complessità di codifica, quindi, chiaramente trascurabile. E' superfluo dire che il comportamento mostrato varia con i valori dei parametri e la situazione diviene certamente più critica in alcuni casi limite, che tuttavia non sono significativi nelle applicazioni. Invece la nostra scelta dei parametri è ragionevole per la codifica di immagini telerilevate.

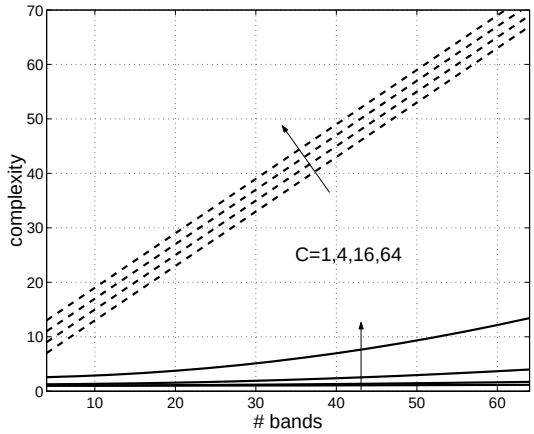


Figura 7.5: Commplessità di progetto e di codifica per lo schema U-CBC.

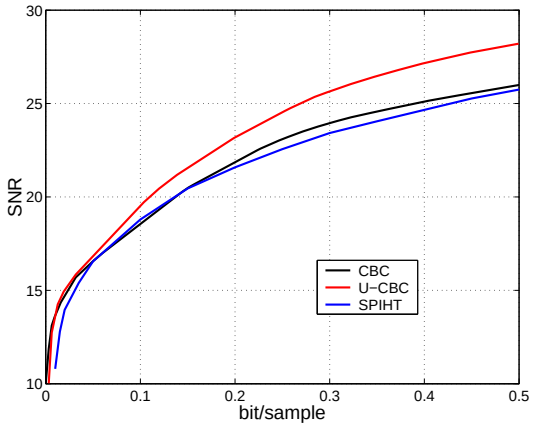


Figura 7.6: Prestazioni tasso distorsione per l'immagine LANDSAT TM.

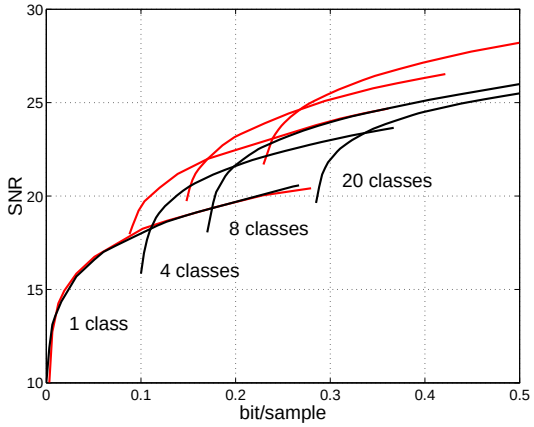


Figura 7.7: Prestazioni tasso distorsione per l'immagine LANDSAT TM, per una differente quantità di classi.

7.4.2 Prestazioni tasso-distorsione

Possiamo ora passare ad esaminare le prestazioni tasso-distorsione. Nella Fig. 7.4.1 riportiamo le prestazioni relative alla nostra immagine di test sia del originale CBC (in nero), sia della nostra versione *non supervisionata* U-CBC (in rosso), sia dallo schema basato SPIHT-3D (in blu) [24] adattato alle immagini multispettrali [13], e reso disponibile dagli autori su web [25]. In questo esperimento, abbiamo in realtà impiegato un codificatore SPIHT modificato [12] dove la KLT è usata al posto della WT nella direzione spettrale, la qual cosa comporta prestazioni migliori e permette di trattare immagini con un numero arbitrario di bande.
Prima di tutto, possiamo osservare che entrambe le versioni del CBC funzionano meglio della tecnica di riferimento basata su SPIHT, giustificando il nostro interesse verso l'approccio basato sulla *classificazione*. Ciò che è più interessante è tuttavia che, malgrado l'informazione aggiuntiva richiesta dal dizionario VQ, dalle matrici KLT e dai parametri di quantizzazione, e malgrado il sottocampionamento introdotto nella fase di progetto, la nostra versione *supervisionata* funziona meglio del codificatore originale CBC (e di SPIHT naturalmente) di circa 1 o 2 dB a tutti i tassi di interesse. In altre parole, usando gli strumenti di codifica progettati direttamente sui dati da codificare, si ottiene un miglioramento in compressione superiore al costo di invio delle statistiche. Questo approccio presenta inoltre il vantaggio non trascurabile di

poter effettuare la codifica a bordo senza bisogno di avere a disposizione un data base di statistiche esterno.
E' da notare che le curve nelle Fig. 7.4.1 sono calcolate come inviluppo superiore delle reali curve tasso-distorsione ottenute per differenti valori del numero di classi C. Nella Fig.7 riportiamo alcune di queste curve con C=1,4,8,20, sia per CBC (in nero) che per U-CBC (in rosso), al fine di ottenere una migliore percezione delle differenze prestazionali. Innanzitutto, la curva per C=1, dove nessuna *classificazione* è eseguita, rivelano prestazioni peggiori eccetto per i tassi di bit estremamente bassi. Tutte le altre curve partono da un tasso iniziale non nullo, corrispondenti al costo della side-information, ed un valore iniziale relativamente alto di SNR, corrispondente al primo passo di codifica VQ. Osservando questi punti iniziali, si constata che il CBC *non supervisionato* beneficia di un vantaggio significativo, sia per la distorsione sia per il tasso di codifica, a causa rispettivamente della migliore attendibilità e della maggior regolarità delle *mappe di classificazione*. Tale vantaggio cresce all'aumentare del numero di classi. Infatti, è relativamente facile scegliere 20 classi significative all'interno di un'immagine, ma è molto difficile che le stesse identiche classi siano presenti in altre immagini, anche molto simili. Quindi, l'approccio off-line di CBC mostra alcune difficoltà per ampi valori di C, al punto che la curva della *classe* 20 non supera mai la curva della *classe* 8, mentre il progetto on-line è valido in tutte le situazioni. In aggiunta, sebbene la VQ espliciti già molte ridondanze nell'immagine, la KLT classificata è ancora in grado di migliorare la qualità della codifica. L'uso dei quantizzatori parametrici piuttosto che ad-hoc non sembra produrre degradazioni apprezzabili.

Per completare l'analisi, la fig.7.4.2 mostra la banda 5 della nostra immagine test e la stessa banda compressa con un CBC *non supervisionato* a 0.4 bit/campione, usando una *segmentazione* a 20 classi. Persino ad un fattore di compressione così elevato (20:1) non sono visibili artefatti di codifica.

Simili risultati sono stati ottenuti per tutte le immagini multispettrali a nostra disposizione. Ad esempio, consideriamo una sezione 448x448 dell'immagine test IKONOS di San Diego, che comprende soltanto quattro bande ed ha una risoluzione spaziale più elevata dell'immagine TM, 4m invece di 20m. La Fig. 7.4.2 riporta le curve tasso-distorsione [5] dello schema U-CBC(rosso), ottenuto di nuovo come inviluppo superiore delle curve per un numero prefissato di classi, e le paragona con le curve di prestazioni di SPIHT, sia nella versione originale (linea tratteggiata in blu) che usa la trasformata wavelet nella dimensione spettrale e nella versione modificata (linea continua in blu) con la KLT

[5] Da ora in poi, noi adottiamo l'usuale definizione di SNR come $10\log_{10}(\mathrm{VAR}(X)/\mathrm{MSE})$.

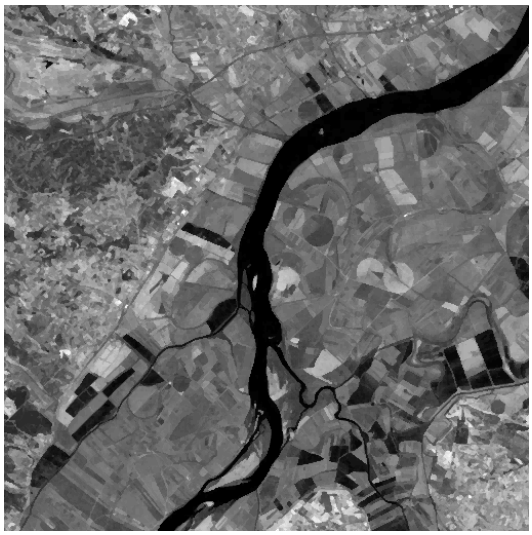


Figura 7.8: La banda 5 dell'immagine LANDSAT TM prima e dopo la codifica a 0.40 bit/campione.

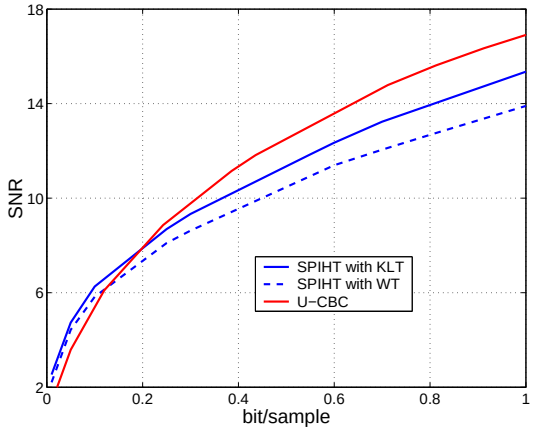


Figura 7.9: Prestazioni tasso-distorsione per l'immagine IKONOS.

Figura 7.10: La banda verde dell'immagine IKONOS prima e dopo la codifica a 0.80 bit/campione.

nella dimensione spettrale.
I risultati confermano il comportamento generale osservato per l'immagine TM: eccetto per tassi di bit molto bassi, che sono di scarso interesse dato il basso livello di SNR, il paragone si esprime sempre in favore dell'U-CBC, che guadagna 1-2 dB rispetto allo SPIHT basato sul KLT e fino a 3 dB rispetto alla versione originale basata sul WT.
La fig. 7.4.2 rappresenta la banda verde originale dell'immagine di test e la stessa banda codificata dallo schema U-CBC ad 0.8 bit/campione, a valle di una *segmentazione* in 20 classi, confermando ancora le buone prestazioni del processo di codifica. Notiamo che, rispetto alla precedente immagine LANDSAT TM, è necessario un maggiore tasso di codifica per ottenere un risultato pienamente soddisfacente, sia perchè l'immagine è molto più ricca di dettagli sia perchè vi è un minor numero da poter codificare congiuntamente.

Infine, la fig. 7.4.2 riporta il risultato del medesimo esperimento effettuato su un segmento di 32 bande contigue della sezione 512X512 di un'immagine di test acquisita con il sensore iperspettrale GER. Il comportamento in questo caso è differente. Infatti, mentre U-CBC garantisce un consistente guadagno rispetto allo SPIHT convenzionale effettuato su wavelet 3D, più di 1 dB a 0.5 bit/campione, la versione che impiega SPIHT a valle della KLT funziona

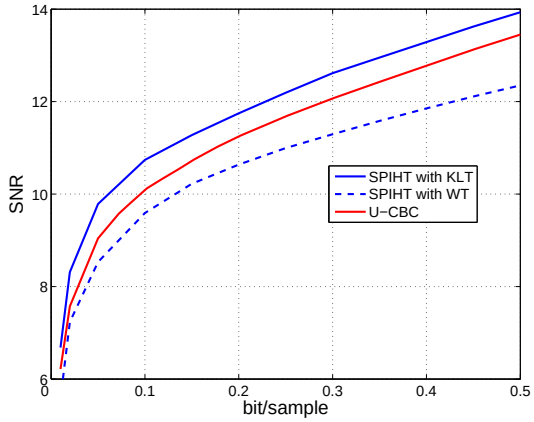


Figura 7.11: Curve tasso distorsione per la sezione di bande contigue dell'immagine GER.

meglio di U-CBC, guadagnando circa 0.5 dB a tutti i tassi. Una possibile spiegazione per questo risultato sta nella natura iperspettrale dell'immagine test considerata (vedi Fig. 7.4.2) che è più omogenea spettralmente rispetto alle immagini multispettrali. Quindi la KLT classificata garantisce soltanto un guadagno limitato rispetto alla KLT globale la quale rappresenta abbastanza bene la dipendenza tra le bande contigue. Tale guadagno non è sufficiente a compensare il costo dell'informazione collaterale, e la ridotta efficienza di una codifica DCT mono-dimensionale dei coefficienti, invece di una codifica trasformata wavelet discreta (DWT) bidimensionale.

Come per le altre immagini, completiamo la nostra analisi nel mostrare in Fig. 7.4.2 una banda dell'immagine originale di 10 bit/campione e la sua controparte compressa, ottenuta con lo schema CBC *non supervisionato* a 0.3 bit/campione, usando una *segmentazione* a 16 classi. Ancora, malgrado una percentuale di compressione maggiore di 30:1, la qualità soggettiva (come pur quella oggettiva) è così buona che la codifica è da considerarsi quasi senza perdita di informazione.

7.4.3 Valutazione della ricostruzione per le elaborazioni successive

In tutte le precedenti analisi, abbiamo usato l'SNR come misura oggettiva e sintetica della qualità di codifica. Tuttavia, nel campo del telerilevamento, la

Figura 7.12: La banda 24 prima e dopo la codifica del segmento a 0.30 bit/campione.

qualità di un'immagine compressa è meglio misurata in funzione delle elaborazioni successive come la *segmentazione*, la *classificazione*, il riconoscimento etc. Per questo motivo, ci sembra opportuno concludere l'analisi presentando un esperimento di *segmentazione*, eseguito sulla nostra prima immagine test, la sezione Landsat TM mostrata nella Fig. 7.4.2, per valutare il grado di preservazione del valore diagnostico delle immagini a valle della compressione.
A tale scopo abbiamo segmentato l' immagine originale in 6 regioni, usando un algoritmo di clustering (*non supervisionato*) a minima distanza. Non avendo a disposizione per l'immagine considerata la ground-truth, questa *segmentazione* è stata assunta come quella corretta. Quindi, usando i vettori template delle classi ricavate dalla *segmentazione* così effettuata, l' immagine è stata di nuovo segmentata, questa volta in maniera *supervisionata*, dopo essere stata compressa e ricostruita a 0.4 bit per campione sia con lo schema U-CBC che con il miglior algoritmo di riferimento, SPIHT con KLT. Infine sono state calcolate le matrici di confusione tra la *classificazione* a valle della codifica lossy e la *classificazione* dei dati originali; esse vengono riportate in Fig. 7.4.3 ed in figura 7.4.3 rispettivamente per ciascuno dei due algoritmi.

I risultati si esprimono chiaramente in favore dell' algoritmo proposto, con un tasso complessivo di errore di circa 6 punti percentuale per U-CBC e ben

CBC	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	Total	User's acc
class 1	26714			7				234	26955	99.1%
class 2		3222	52						3274	98.4%
class 3		164	31648		260				32072	98.7%
class 4	36			40560		3020	1141	1835	46592	87.0%
class 5			1763		48439	507			50709	95.5%
class 6				445	2566	59410			62421	95.2%
class 7				893			14522	456	15871	91.5%
class 8	143			1663			598	21846	24250	90.1%
Total	26893	3386	33463	43568	51265	62937	16261	24371		
Prod.'s acc	99.3%	95.2%	94.6%	93.1%	94.5%	94.4%	89.3%	89.6%		

Figura 7.13: Matrice di confusione per U-CBC.

SPIHT	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	Total	User's acc
class 1	25970			104				881	26955	96.3%
class 2		2886	388						3274	88.1%
class 3		333	28098		3630	11			32072	87.6%
class 4	137			35844	20	6563	1262	2766	46592	76.9%
class 5			3843	12	40179	6675			50709	79.2%
class 6			6	6351	7090	48946	26	2	62421	78.4%
class 7	2			1703		53	12808	1305	15871	80.7%
class 8	1250			2865		1	630	19504	24250	80.4%
Total	27359	3219	32335	46879	50919	62249	14726	24458	0	
Prod.'s acc	94.9%	89.7%	86.9%	76.5%	78.9%	78.6%	87.0%	79.7%		

Figura 7.14: Matrice di confusione per SPIHT.

18 per SPIHT (risultati che confermano questa tendenza sono stati ottenuti per tutte le immagini a qualunque tasso). E' chiaro che questo enorme miglioramento è soltanto parzialmente dovuto ad una migliore qualità di ricostruzione, considerando che l' SNR mostra solo un miglioramento di circa 2.5 dB a quel tasso. La principale ragione, invece, è che il CBC include un passo di clustering che, seppure indipendente dall'applicazione finale di *segmentazione supervisionata*, tende a preservare una qualche forma di aggregazione dei dati e conduce dunque a prestazioni migliori.

Un approccio orientato all'applicazione finale, in alternativa, potrebbe *esplicitamente* impiegare la *mappa di segmentazione* ricavata dai dati originali come supporto per la codifica. Questa procedura potrebbe rivelarsi utile in taluni casi applicativi, in cui si vuole ad esempio preservare la bontà della *segmentazione* o in cui si vuole memorizzare anche l'informazione relativa ad una mappa specifica, tuttavia ne tralasciamo una trattazione più approfondita in quanto perderemmo inevitabilmente di generalità. D'altra parte il clustering spettrale da noi effettuato, soprattutto se le classi impiegate sono in quantità sufficiente, è comunque un passo tipico di preprocessing per un gran numero di altre tecniche di *segmentazione*.

7.5 Conclusioni

In questo capitolo è stato analizzato lo sviluppo di un adattamento totalmente *non supervisionato* del codificatore per immagini multispettrali basato su *classificazione* e proposto in [4], così da permetterne l'impiego in tempo reale e a bordo di un velivolo o di un satellite.
Oltre alla modifica di alcuni blocchi costitutivi del codificatore originale, abbiamo dovuto gestire il progetto on line del cassificatore e della trasformata oltre che la trasmissione della risultante informazione collaterale. Un saggio (ed attentamente collaudato) uso di tecniche sottocampionamento ci ha permesso di limitare l' aumento nella complessità (di calcolo e di codifica) e quindi di ottenere un codificatore funzionale.
I numerosi esperimenti eseguiti hanno provato che il nuovo schema *non supervisionato*, U-CBC, presenta prestazioni tasso-distorsione migliori della versione originale *non supervisionata* CBC. Questa è stata parziamlente una sorpresa, infatti i molti parametri prima reperibili in una base di dati ora devono essere trasmessi come informazione collaterale, ma l'adeguamento dei parametri alle statistiche della stessa immagine da codificare compensa abbondantemente questo svantaggio.
U-CBC ha fornito anche migliori risultati rispetto allo schema di codifica di riferimento basato su wavelet 3D e SPIHT e, per le immagini multispettrali (ma per il momento non per quelle iperspettrali), anche rispetto all'analoga versione che impiega la KLT spettrale invece della wavelet.
Il codificatore classificato originale si era rivelato già un interessante strumento, specialmente per la sua peculiarità di fornire un *clustering* spettrale dei dati oltre che le statistiche ciascuna *classe*, come informazione innestata nel flusso di codifica. La versione *non supervisionata* qui sviluppata, grazie ad un attento progetto a bassa complessità, presenta un miglioramento deciso delle prestazioni in termini di tasso-distorsione e, ad un tempo, la possibilità di imbarco del sistema a bordo di un velivolo o di un satellite per il telerilevamento, essendo in grado di soddisfare le specifiche richieste di elaborazione in tempo reale e di bassa complessità di calcolo.

Bibliografia

[1] M. Cagnazzo, L. Cicala, G. Poggi, L. Verdoliva. “Low-complexity compression of multispectral images based on classified transform coding”, *Elsevier Signal Processing: Image Communication*, vol. 21, n. 10, pp. 850-861, novembre 2006.

[2] M. Cagnazzo, L. Cicala, G. Poggi, G. Scarpa, and L. Verdoliva. “An unsupervised segmentation-based coder for multispectral images.” *European Signal Processing Conference* (EUSIPCO), settembre 2005.

[3] L.Cicala, “Ottimizzazione delle risorse nella classificazione e codifica di immagini multispettrali.”, Tesi di Laurea, Univ. degli Studi Federico II di Napoli, 2003.

[4] G.Gelli, G.Poggi, “Compression of multispectral images by spectral classification and transform coding”, *IEEE Transactions on Image Processing*, pp.476-489, Apr. 1999.

[5] M. Finelli, G. Gelli, and G. Poggi. Multispectral image coding by spectral classification. *IEEE International Conference on Image Processing* (ICIP), pp. 605-608, Losanna (CH), settembre 1996.

[6] M. Cagnazzo, S. Parrilli, G. Poggi, and L. Verdoliva. “Improved class-based coding of multispectral images with shape-adaptive wavelet transform”, IEEE Geoscience and Remote Sensing Letters vol. 4, pp. 566-570, ottobre 2007.

[7] M. Cagnazzo, R. Gaetano, S. Parrilli, and L. Verdoliva. “Region based compression of multispectral images by classified klt”, *European Signal Processing Conference* (EUSIPCO), Firenze, settembre 2006.

[8] G. Poggi and A. R. P. Ragozini. ”Image segmentation by tree-structured Markov random fields”, *IEEE Signal Processing Letters*, vol. 6, pp. 155-157, giugno 1999.

[9] http://www.kakadusoftware.com/

[10] S.S.Shen, J.H.Kasner, “Effects of 3D wavelets and KLT-based JPEG-2000 hyperspectral compression on exploitation”, *Proceedings of the SPIE*, pp.167-176, 2000.

[11] J.A.Saghri, A.G.Tescher, A.M.Planinac, “KLT/JPEG 2000 multispectral bandwidth compression with region of interest prioritization capability”, *Proceedings of the SPIE*, pp.226-235, 2003.

[12] P.L.Dragotti, G.Poggi, A.R.P.Ragozini, “Compression of multispectral images by three-dimensional SPIHT algorithm”, *IEEE Transactions on Geoscience and Remote Sensing*, pp.416-428, Jan. 2000.

[13] X.Tang, W.A.Pearlman, J.W.Modestino, “Hyperspectral Image Compression Using Three-Dimensional Wavelet Coding”, SPIE *Electronic Imaging*, Vol. 5022, gennaio 2003.

[14] J.A.Saghri, A.G.Tescher, J.T.Reagan, “Practical transform coding of multispectral imagery”, *IEEE Signal Processing Magazine*, pp.32-43, Jan. 1995.

[15] G.P.Abousleman, M.W.Marcellin, B.R.Hunt, “Compression of hyperspectral imagery using the 3-D DCT and hybrid DPCM/DCT”, *IEEE Transactions on Geoscience and Remote Sensing*, pp.26-34, Jan. 1995.

[16] Q.Du, C.Chang, “Linear mixture analysis-based compression for hyperspectral image analysis”, *IEEE Transactions on Geoscience and Remote Sensing*, pp.875-891, April 2004.

[17] A.Gersho, R.M.Gray, *Vector quantization and signal compression*, Kluwer Academic Publisher, 1992.

[18] G.R.Canta, G.Poggi, “Kronecker-product gain-shape vector quantization for multispectral and hyper-spectral image coding”, *IEEE Transactions on Image Processing*, pp.668-678, May 1998.

[19] S.E.Qian, A.B.Hollinger, D.Williams, D.Manak, "Vector quantization using spectral index-based multiple subcodebooks for hyperspectral data compression", *IEEE Transactions on Geoscience and Remote Sensing*, pp.1183-1190, May 2000.

[20] S.E.Qian, "Hyperspectral data compression using a fast vector quantization algorithm", *IEEE Transactions on Geoscience and Remote Sensing*, pp.1791-1798, Aug. 2004.

[21] http://www.globalsecurity.org/space/systems/nemo.htm

[22] G.Gelli, G.Poggi, A.R.P.Ragozini, "Multispectral-image compression based on tree-structured Markov random field segmentation and transform coding", *Proc. 1999 IEEE International Geoscience and Remote Sensing Symposium*, vol.2, pp.1167-1170, June 1999.

[23] G.Golub, C.van Loan, *Matrix computations, 3rd ed.*, Johns Hopkins University Press, 1996.

[24] A.Said, W.A.Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees", *IEEE Transactions on Circuits and Systems for Video Technology*, pp.243-250, June 1996.

[25] http://www.cipr.rpi.edu/research/SPIHT/

Conclusioni e sviluppi futuri

Nel corso di questo lavoro di tesi sono state affrontate due strategie di codifica di segnali visuali multidimensionali: *analisi multirisoluzione multidimensionale* e *classificazione*.
Per l' analisi multirisoluzione si è studiata una particolare famiglia di algoritmi di quantizzazione e codifica, gli *zerotree coder*, di notevole importanza storica e applicativa. Per essi è stata proposta, per la prima volta con questo lavoro di tesi, una generalizzazione. Dello *zerotree coder* generalizzato è stato poi fornito un modello del tipo automa a stati finiti. Da un lato, la generalizzazione permette ora di confrontare in maniera più chiara le soluzioni già proposte in letteratura. Dall'altro, la formalizzazione consente un progetto più semplice di nuove soluzioni.
Gli sviluppi futuri più immediati della ricerca presentata sono l'impiego di *zerotree* del terzo *ordine*, lo studio di diverse topologie di *alberi descrittori* per vecchie e nuove applicazioni, l'introduzione di forme più efficaci di codifica aritmetica adattativa. Un passo ulteriore potrebbe essere quello di completare il modello teorico con strumenti statistici adeguati alla descrizione informazionale degli *alberi descrittori* e delle *leggi di evoluzione*.
Della *codifica classificata* sono state finora presentate in letteratura applicazioni per le immagini multicanale telerilevate. In questo lavoro di tesi si è studiata la fattibilità del progetto della classificazione e della trasformata sui dati stessi, dimostrando che la complessità dell'approccio è accettabile. Questo è stato un punto importante anche per i successivi avanzamenti della ricerca nel settore.
Gli sviluppi futuri di questa tematica possono essere, sul lato applicativo, l'implementazione delle diverse funzionalità e l'impiego in nuovi domini applicativi, sul lato teorico, l'approfondimento del concetto di *trasformata classificata*. Formalizzato lo stesso, si potrebbe provare a formulare il problema di ottimizzazione congiunta della classificazione e della trasformata.
Le possibili applicazioni degli *zerotree coder* sono le stesse dello standard

per immagini fisse JPEG2000: codifica di immagini monocromatiche e multicanale, codifica *near losselss* e a bassa latenza di sequenze video. Se da un lato, rispetto a JPEG2000, le soluzioni basate su *zerotree coder* presentano lo svantaggio di non essere standardizzate e di avere prestazioni leggermente inferiori, dall'altro, la semplicità di progetto degli stessi permette una maggiore adattabilità ad esigenze specifiche, non previste dallo standard. Ad esempio al progettista possono essere richieste una decomposizione o una trasformata alternativa, una funzionalità nuova, l'efficacia su dati non convenzionali, una particolare implementazione *hardware*.
Per quel che riguarda la *codifica mediante classificazione*, il dominio di applicazione privilegiato è quello delle immagini telerilevate. Da un lato, le strategie di segmentazione non supervisionate ed orientate all'efficienza di codifica, fanno dei *codificatori classificati* una valida alternativa alle tecniche di codifica convenzionali, così come dimostrato sperimentalmente in questa tesi. Dall'altro, esse presentano la funzionalità di una classificazione innestata nel flusso di codifica. Se i *cluster* spettrali sono molti, algoritmi più complessi possono sfruttare la tassellazione spettrale così ottenuta per una successiva segmentazione orientata alla semantica della scena e finalizzata ad applicazioni di *pattern recognition* (elaborazione di mappe tematiche, riconoscimento di disastri naturali, *change detection*, etc.).
Sell'applicazione non è la codifica a bordo, con specifici vincoli di *real-time*, ma l'immagazzinamento dei dati in un sistema informativo geografico, è possibile utilizzare mappe di segmentazione non finalizzate alla codifica, ma di tipo semantico. Vi è l'opportunità, ad esempio, di impiegare mappe tematiche, in cui si distinguono le diverse tipologie di terreno all'interno della scena rappresentata. Da un lato, si è visto che queste mappe non sono sempre efficaci per la codifica, ma questo è un problema risolvibile magari con un ulteriore passo di *clustering* sulle classi semantiche. Dall'altro, il *data base* geografico può essere in questo modo organizzato, in forma codificata, in due strati: uno di dati tematici (strato *geo-tipico*) ed un altro di dati reppresentanti le risposte spettrali originali (strato *geo-specifico*). In questo modo, ad esempio, è possibile implementare un particolare tipo di scalabilità di contenuto. Utenti con minori risorse o con maggiori esigenze di velocità potrebbero voler navigare soltanto all'interno dello strato *geo-tipico*, utenti con maggior esigenze di informazione invece potrebbero voler decodificare anche la parte *geo-specifica*.
Un'altra funzionalità interessante è l'accesso diretto alle classi. Ad esempio l'utente o l'applicazione *client*, possono essere interessati unicamente a decodificare una specifica classe geografica. Si può pensare che magari si vuole

accedere solo ai bacini idrici o alle aree boschive, per misurarne i cambiamenti nel tempo. Un'algoritmo automatico, grazie alla codifica classificata, potrebbe accedere solo ai dati utili, senza bisogno di dover decodificare dati invece non necessari alla particolare applicazione. Le funzionalità accennate sono di estrema importanza se si pensa a data base geografici distribuiti su rete, in cui al tempo di decodifica bisogna aggiungere quello di *download*.