Principi ed applicazioni dell'elaborazione digitale delle immagini
La Trasformata di Walsh

Nel campo dell'elaborazione digitale delle immagini, sono diverse le funzioni di trasformazione di dominio, dette Trasformate, di uso comune. Si è già trattato della più nota ed importante di queste, la trasformata di Fourier; ne esistono altre meno note, tuttavia utili ed interessanti che descriveremo nel prosieguo e nei prossimi appuntamenti. Questa volta parleremo della trasformata di Walsh, successivamente verranno trattate quella di Hadamard e quella del coseno, meglio nota come DCT. Abbiamo deciso di approfondire gli aspetti teorici ed applicativi di questi argomenti, per il grande interesse che il problema della compressione delle immagini sta suscitando. Le tecnologie multimediali hanno posto all'attenzione di tutti, il problema della memorizzazione della grande quantità di immagini, base di qualsiasi applicazione multimediale. In particolare si inizia a sentire la diffusione dello standard JPEG e dell'imminente rilascio di quello per le sequenze d'immagini, MPEG. Questi standard come altri metodi di compressione proposti fanno largo uso delle trasformate suddette, perciò è nostra intenzione svelare ai lettori i fondamenti delle tecnologie che si apprestano ad utilizzare o di cui fanno già uso

di Giuseppe Cardinale Ciccotti

Trasformate separabili e kernel di trasformazione

Tutte le trasformate che ci accingiamo a trattare di cui abbiamo già parlato, appartengono ad una classe di trasformate molto importante soprattutto per le implicazioni pratiche che se ne ricavano.

Ognuna di esse può essere espressa in forma generale come una relazione tra due funzioni:

$$T(u)=\sum_{x=0}^{N-1} f(x)\, g(x,u)$$

dove T(u) è la trasformata di f(x) e g(x,u) è detto «kernel di trasformazione diretta», con u che assume valori interi da 0 a N-1.

Similarmente, la trasformata inversa è data dalla relazione:

$$f(x)=\sum_{u=0}^{N-1} T(u)\, h(x,u)$$

dove h(x,u) è il «kernel di trasformazione inversa» e x va da 0 a N-1.

Il kernel è ciò che caratterizza la trasformata e varia con essa, tuttavia comprenderete che essendo medesima la formula che restituisce la trasformata, l'algoritmo che adopereremo per il calcolo sarà il medesimo e ciò torna indubbiamente molto utile.

Nella stessa forma possiamo mettere le trasformate bidimensionali di array quadrati:

$$T(u,v)=\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\, g(x,y,u,v)$$

per la trasformata diretta, e:

$$f(x,y)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)\, h(x,y,u,v)$$

per quella inversa con g(x,y,u,v) e h(x,y,u,v) detti kernel di trasformazione diretta ed inversa, rispettivamente.

Consideriamo ora più attentamente il kernel g(x,y,u,v); esso è detto separabile se:

$$g(x,y,u,v)=g1(x,u)\, g2(y,v)$$

Inoltre, il kernel g è simmetrico se g1 restituisce la stessa funzione di g2, cioè se g1 è «funzionalmente equivalente» a g2, come si usa dire. In tal caso la relazione precedente può essere espressa nella forma:

$$g(x,y,u,v)=g1(x,u)\, g1(y,v)$$

Queste ultime proprietà valgono anche per il kernel di trasformazione inversa, sostituendo h(x,y,u,v) a g(x,y,u,v).

La trasformata di Fourier bidimensionale verifica proprio queste condizioni, infatti il kernel è:

$$g(x,y,u,v)=1/N \exp[-j2\pi(ux+vy)/N]$$

che è separabile e simmetrico perché:

$$g(x,y,u,v) = g1(x,u)\, g1(y,v) = 1/N\{ \exp[-j2\pi ux/N] \exp[-j2\pi vy/N]\}$$

allo stesso modo si può dimostrare che la trasformata inversa è separabile e simmetrica.

Il risvolto pratico della proprietà di separabilità è che una trasformata bidimensionale può essere calcolata tramite la trasformata monodimensionale in due passi successivi ed indipendenti. Per esempio trasformando prima lungo ciascuna riga di f(x,y), ottenendo:

$$T(x,v)=\sum_{y=0}^{N-1} f(x,y)\, g2(y,v)$$

per x e v compresi tra 0 e N-1. Successivamente la trasformata monodimensionale è calcolata lungo le colonne di T(x,v) con il risultato:

$$T(u,v)=\sum_{x=0}^{N-1} T(x,v)\, g1(x,u)$$

con u e v tra 0 e N-1. Con il medesimo metodo è possibile calcolare la trasformata inversa h(x,y,u,v).

Figura 1
Kernel della Trasformata di Walsh per N=8.

u\x	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	+	+	+	-	-	-	-
2	+	+	-	-	+	+	-	-
3	+	+	-	-	+	+	-	-
4	+	-	+	-	+	-	+	-
5	+	-	+	-	-	+	-	+
6	+	-	-	+	+	-	-	+
7	+	-	-	+	-	+	+	-

Se il kernel diretto è separabile e simmetrico allora la trasformata diretta può anche essere espressa con la relazione matriciale:

T=AFA

dove **F** è la matrice quadrata **NxN** contenente i pixel dell'immagine, **A** è una trasformazione simmetrica **NxN** di elementi

$a_{ij}=g1(i,j)$

e **T** è la risultante trasformata **NxN** per valori di u e v compresi tra 0 e **N**-1.

Per ottenere la trasformata inversa basta moltiplicare a destra e a sinistra la matrice **T** per una matrice di antitrasformazione **B**. Vale a dire:

BTB=BAFAB

Se **B** è la matrice inversa di **A**, allora ne consegue che:

F=BTB

poiché **BA=I** che la matrice identità, quindi l'immagine digitale **F** può essere recuperata completamente dalla sua trasformata.

Se invece **B** non è esattamente uguale all'inversa di **A**, la matrice

F'=BAFAB

sarà un'approssimazione dell'immagine originaria.

Quali sono i vantaggi derivanti dall'utilizzo della forma matriciale?

Il primo è di tipo computazionale, in quanto è possibile utilizzare gli efficientissimi algoritmi sviluppati in decenni di ricerca sul calcolo matriciale; inoltre dal punto di vista implementativo, si può attingere alle vastissime librerie di calcolo sulle matrici disponibili per qualsasi tipo di macchina e linguaggio. Senza dimenticare che esistono processori specializzati nel calcolo di matrici!

Il secondo vantaggio è relativo invece al fatto che la trasformazione di dominio riduce la ridondanza, nel senso che la matrice **T** risultato della trasformata può essere decomposta in prodotto di matrici che in totale hanno un numero di elementi pari a zero maggiore della matrice originale: questo implica la possibilità di compattare in maniera semplice (e quindi veloce) le immagini.

La trasformata di Walsh

La trasformata discreta di Walsh di una funzione f(x) è definita quando N=2 2^n.

Si ottiene ponendo il kernel pari a:

$$g(x,u)=1/N\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}$$

nell'espressione che restituisce T(u). In definitiva la trasformata W(u) è data da:

$$W(u)=1/N\sum_{x=0}^{N-1}f(x)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}$$

dove $b_k(z)$ è il k-esimo bit nella rappresentazione binaria di z. Per esempio, se n=3 e z=6 (110 in binario), avremo $b_0(z)=0$, $b_1(z)=1$, e $b_2(z)=1$.

I valori di g(x,u), non considerando il termine costante 1/N, sono listati in figura 1 quando N=8. Come potete vedere, il kernel della trasformata di Walsh è una matrice simmetrica nelle quali le righe e le colonne sono ortogonali fra loro. Ciò significa che la riga i-esima è uguale alla colonna i-esima, vale a dire che ogni riga è la colonna omologa trasposta e viceversa.

Queste proprietà, che valgono in generale, comportano anche l'importante risultato che il kernel della trasformata inversa è identico al kernel della trasformata diretta, tranne che per un fattore moltiplicativo 1/N. Il kernel inverso è quindi:

$$h(x,u)=\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}$$

e la trasformata inversa vale:

$$f(x)=1/N\sum_{u=0}^{N-1}W(u)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}$$

Interessante è notare che la trasformata diretta ed inversa differiscono soltanto per il termine 1/N; è facile allora utilizzare il medesimo algoritmo sia per il calcolo della trasformata diretta che per quello della trasformata inversa. Basta moltiplicare per N il risultato ottenuto dall'algoritmo della trasformata diretta calcolato con W(u) al posto di f(x).

Nel caso bidimensionale i kernel della trasformata di Walsh sono dati da:

$$g(x,y,u,v)=1/N\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$$

$$h(x,y,u,v)=1/N\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$$

Queste sono proprio le forme più interessanti per le applicazioni di image processing, dove la trasformata diretta e quella inversa hanno la medesima importanza. Conla formulazione precedente le trasformate hanno lo stesso kernel

```
      SUBROUTINE FWT(F,LN)
      REAL  F(1024),T
      N=2**LN
      NV2=N/2
      NM1=N-1
      J=1
      DO 3 I=1,NM1
         IF(I.GE.J) GO TO 1
         T=F(J)
         F(J)=F(I)
         F(I)=T
1        K=NV2
2        IF(K.GE.J) GO TO 3
         J=J-K
         K=K/2
         GO TO 2
3        J=J+K
      DO 5 L=1,LN
         LE=2**L
         LE1=LE/2
         DO 5 J=1,LE1
            DO 4 I=J,N,LE
               IP=I+LE1
               T=F(IP)
               F(IP)=F(I)-T
4              F(I)=F(I)+T
5           CONTINUE
      DO 6 I=1,N
6        F(I)=F(I)/FLOAT(N)
      RETURN
      END
```

Figura 2 - Routine Fortran per il calcolo della Trasformata Veloce di Walsh.

e perciò sono anche identiche nella forma:

$$W(u,v)=1/N \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \prod_{i=0}^{n-1} (-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$$

$$f(x,y)=1/N \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} W(u,v) \prod_{i=0}^{n-1} (-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]}$$

Perciò qualsiasi algoritmo usato per calcolare la trasformata di Walsh diretta può essere adoperato senza alcuna modifica per la trasformata inversa.

Come già anticipato il kernel della trasformata di Walsh è separabile, infatti:

$$g(x,y,u,v) = g1(x,u)\, g1(y,v) = h1(x,u)\, h1(y,v) =$$
$$=1/N\left[\prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}\right] \left[\prod_{i=0}^{n-1} (1)^{b_i(x)b_{n-1-i}(u)}\right]$$

Come a suo tempo fu verificato per la trasformata di Fourier, così è facile eseguire la trasformata bidimensionale di Walsh come due successive applicazioni della trasformata monodimensionale.

L'algoritmo Fast Walsh Transform

La somiglianza con la trasformata di Fourier si riflette anche nell'algoritmo che si può progettare per un calcolo efficiente della trasformata di Walsh. Senza dimenticare che è sempre possibile e spesso assai efficiente calcolare la trasformata tramite moltiplicazioni di matrici, come mostrato nei paragrafi precedenti, soprattutto se si hanno a disposizione librerie per il calcolo matriciale, un modo rapido e semplice di calcolare la trasformata di Walsh, è quello di adottare lo stesso algoritmo della FFT, Fast Fourier Transofrm.

L'unica differenza è che i termini esponenziali sono posti pari ad 1.

Ricordiamo che la FFT si basa sulla tecnica dei «raddoppi successivi» e sull'osservazione che la trasformata per una certa u è la semisomma della trasformata della parte pari e della parte dispari moltiplicata per un fattore che nel caso della FFT è un esponenziale e nel caso della FWT è 1.

La trasformata della parte pari è, come dice il nome, ottenuta eseguendo il calcolo della trasformata solo sui campioni della f(x) di ordine pari, a distanza 2x. La trasformata della parte dispari viceversa è la trasformata sui campioni a distanza 2x+1. In formule, la FWT:

$$W(u)=1/2\{W_{pari}(u)+W_{dispari}(u)\}$$

$$W(u+M)=1/2\{W_{pari}(u)-W_{dispari}(u)\}$$

$$W_{pari}(u)=1/M \sum_x f(2X)$$

$$W_{dispari}(u)=1/M \sum_x f(2X+1)$$

dove M=N/2, x,u=0,1,...,M-1, e W(u) è la trasformata monodimensionale di Walsh.

Facciamo ora un esempio ponendo N=4 ed applichiamo le relazioni che abbiamo ricavato finora. Si ottiene:

$$W(0) = 1/4 \sum_{x=0}^{3} f(x) \prod_{i=0}^{1} (-1)^{b_i(x)b_{n-1-i}(0)} = 1/4[f(0)+f(1)+f(2)+(3)]$$

$$W(1) = 1/4 \sum_{x=0}^{3} f(x) \prod_{i=0}^{1} (-1)^{b_i(x)b_{n-1-i}(1)} = 1/4[f(0)+f(1)-f2)-f(3)]$$

$$W(2) = 1/4 \sum_{x=0}^{3} f(x) \prod_{i=0}^{1} (-1)^{b_i(x)b_{n-1-i}(2)} = 1/4[f(0)-f(1)+f(2)-(3)]$$

$$W(3) = 1/4 \sum_{x=0}^{3} f(x) \prod_{i=0}^{1} (-1)^{b_i(x)b_{n-1-i}(3)} = 1/4[f(0)-f1)-f(2)+f(3)]$$

Questa è la trasformata di Walsh, vediamo se parimenti otteniamo gli stessi risultati applicando il metodo FWT, calcoliamo prima la parte pari e quella dispari:

$W_{pari}(0)=1/2\{f(0)+f(2)\}$ $\quad$ $W_{pari}(1)=1/2\{f(0)-f(2)\}$

$W_{dispari}(u)=1/2\{f(1)+f(3)\}$ $\quad$ $W_{dispari}(1)=1/2\{f(1)-f(3)\}$

Ricordiamo che le trasformate della parte pari e di quella dispari in questo caso si ottengono al primo passo perché N=4, in ogni caso sarebbero stati necessari $lg_2(N)$-1 proprio per la tecnica dei raddoppi successivi, per ulteriori chiarimenti si può far riferimento ad Appunti di Informatica del numero 127 di MC, dove è sviluppata passo per passo questa tecnica, nel caso di FFT.

Continuando nell'applicazione manuale dell'algoritmo si ricava:

$$W(0)=1/2\{W_{pari}(0)+W_{dispari}(0)\}= 1/4[f(0)+f(1)+f(2)+(3)]$$

$$W(1)=1/2\{W_{pari}(1)+W_{dispari}(1)\}= 1/4[f(0)+f(1)-f(2)-(3)]$$

$$W(2)=1/2\{W_{pari}(0)-W_{dispari}(0)\}= 1/4[f(0)-f(1)+f(2)-(3)]$$

$$W(3)=1/2\{W_{pari}(1)-W_{dispari}(1)\}= 1/4[f(0)-f(1)-f(2)+(3)]$$

Come si vede i risultati dei due algoritmi sono identici, come doveva essere. In figura 2 allora trovate l'implementazione Fortran dell'algoritmo di FWT che non è altro che la versione modificata della routine di FFT, con i coefficienti pari ad 1. C'è da notare che proprio per questo motivo la trasformata di Walsh è reale così per i dati è sufficiente la metà dello spazio necessario alla FFT.

Conclusioni

Gli appuntamenti di questa rubrica sono sempre necessariamente teorici, d'altra parte gli algoritmi e le tecnologie non nascono per caso, ma alla base ci sono sempre idee, intuizioni ed osservazioni. Speriamo che spiegandoli per esteso, in modo essenziale, strumenti che sembrano astrusi risultino ora più «abbordabili» e magari, perché no, più utili. Avevamo promesso su queste pagine che la trattazione teorica sarebbe stata supportata da esempi con Photoshop, tuttavia ci sembra molto più nello spirito della rubrica, approfondire gli aspetti teorici delle tecniche dell'elaborazione di immagini, utilizzando procedure da noi stessi progettate. Tuttavia nell'attesa che chi scrive riceva Photoshop per Windows, con cui visualizzare gli effetti di ciò che trattiamo con le immagini, sarebbe gradito ricevere dai lettori implementazioni, diversi metodi ed informazioni di qualsiasi genere che trattino degli argomenti che analizziamo, in modo da poterli portare a conoscenza di tutti gli affezionati lettori che ci seguono. Ai prossimi appuntamenti.

MC