Il Formato JPEG

Introduzione

Un comitato ISO/CCITT noto come JPEG (Joint Photographic Experts Group) ha definito il primo standard internazionale di compressione per immagini a tono continuo, sia a livelli di grigio che a colori. Lo standard proposto vuole essere il più generico possibile. Lo standard JPEG definisce due metodi di compressione di base, uno basato sull' uso della trasformata DCT con compressione di tipo "lossy" cioè con perdita di informazione, l'altro sull'uso di un metodo predittivo con compressione di tipo "lossless" cioè senza perdita di informazione. L'algoritmo base dello JPEG di tipo "lossy" viene detto "baseline", inoltre sono state definite delle estensioni opzionali del metodo "lossy" per la compressione di tipo gerarchico e progressivo.

La codifica di compressione "baseline"

Trasformazione dello spazio cromatico

Generalmente risulta conveniente trasformare l'immagine dallo spazio cromatico RGB a quello YCrCb, in questo spazio l'immagine viene rappresentata con una componente di luminosità (livelli di grigio) e due componenti cromatiche. La ragione principale di questa trasformazione è che posso permettermi di scartare più informazione nelle componenti cromatiche che in quella della luminosita', questo in quanto l'occhio umano è meno sensibile variazioni di frequenze cromatiche che a quelle di luminosità. Questa trasformazione introduce degli errori di arrotondamento che però sono ininfluenti rispetto alla quantità di errore introdotta con l'applicazione dell'algoritmo JPEG. Nel caso di immagini a livelli di grigio la trasformazione dello spazio cromatico non è necessaria. Anche per le immagini a colori l'applicazione una trasformazione dello spazio cromatico può essere omessa e quinsi si può usare l'algoritmo direttamente sull'immagine rappresentata nello standard RGB, in quanto l'algoritmo lavora indipendentemente su ogni componente cromatica, senza analizzare i tipi di dati contenuti in essa.

Riduzione delle matrici delle componenti cromatiche

Pur non essendo necessaria, l'algoritmo prevede un'operazione di riduzione delle matrici delle componenti cromatiche attraverso la media dei pixel vicini. La

componente relativa alla luminosità resta invariata, mentre le due componenti cromatiche possono venire ridotte 2:1 orizzontalmente e 2:1 o 1:1 (nessuna modifica) verticalmente. Questi due tipi di riduzioni vengono di solito indicate con 2h2v e con 2h1v. Questa operazione riduce la quantita' di informazione di metà o un terzo, numericamente indica una grande perdida di dati ma, per la maggior parte delle immagini, non influenza in modo significativo la qualita' percettiva dell'immagine, questo in quanto l'occhio umano non è sensibile a piccole variazioni cromatiche. Questa operazione non è applicabile ad immagini a livello di grigio, questo è anche uno dei motivi per cui, a parità di qualità, con un'immagine a colori si ottiene un fattore di compressione maggiore.

Trasformata discreta del coseno (DCT)

Prima di applicare la traformata si effettua una operazione di shift a sinistra di 128: I(i,j)=I(i,j)-128. In questo modo il range dei valori dell'immagine non è più da 0 a 255 ma da -128 a 127. La matrice cosı ottenuta viene suddivisa in blocchi 8x8 su cui si applicherà la trasformata , se la dimensione della matrice non è divisibile per otto si aggiungono delle copie dell'ultima riga o colonna sino a rendere la dimensione (sia verticale che orizzontale) divisibile per otto, questo perché valori nulli nei blocchi 8x8 introducono errori nel risultato della trasformata, che deve sempre essere applicata a blocchi 8x8 completi. La trasformata usata è la trasformata discreta del coseno (DCT), che applicata ai valori del blocco 8x8 li restituisce come valori in frequenza (similmente alla trasformata di Fourier). In questo modo si potrà eliminare i dati ad alta frequenza senza toccare quelli a bassa frequenza. La DCT è invertibile senza perdita significante di informazioni, a parte qualche errore di arrotondamento dovuto ai termini del coseno.

Quantizzazione

Si passa poi alla fase di quantizzazione: ogni elemento (64) del blocco 8x8 viene diviso per un coefficente di quantizzazione distinto ed il risultato cosı ottenuto arrotondato all'intero più vicino. Questo è il procedimento fondamentale che porta all'eliminazione dell'informazione meno significativa. Le matrici di quantizzazione che contengono i 64 coefficienti possono essere definite a piacere, lo standard non pone limitazioni, comunque dopo alcuni esperimenti sono state rese note delle matrici di quantizzazione che danno i risultati migliori, le più usate le posso trovare in [1] e [3]. Queste matrici variano a seconda della componente a cui andranno applicate (es.: Y,Cr,Cb,R,G,B).

Codifica di Huffman

Per codificare i coefficenti dopo la quantizzazione si usa prima una codifica del tipo Run Length Encoding in cui vengono codificate le sequenze di zeri e poi applicata la codifica di Huffman. In questo processo non ho perdita di informazione.

Scrittura del file fisico

Nel file fisico verranno inglobati i seguenti dati: grandezza dell'immagine originale, le tabelle di quantizzazione usate (per le immagini a colori sono almeno due), i codici di Huffman usati per la codifica dei dati dell'immagine ed i dati dell'immagine stessa codificata.

Estensioni

La codifica progressiva

La codifica progressiva è stata realizzata per la trasmissione real-time di immagini. Permette di trasmettere i coefficienti della DCT in scan multipli dell'immagine. Con ogni scan il decodificatore può produrre una rappresentazione di qualita' migliore dell'immagine. In questo modo posso trasmettere una immagine a bassa definizione molto velocemente e poi raffinarla in trasmissioni sucessive. Lo spazio utilizzato per la memorizzazione dell'immagine è circa uguale a quello usato per codificare, alla stessa qualità, l'immagine con l'algoritmo "baseline". Il decodificatore deve produrre un ciclo completo di decodifica JPEG per ogni "scan" dell'immagine.

La codifica gerarchica

La codifica gerachica rappresenta l'immagine con diverse risoluzioni. Per esempio: si possono creare le seguenti versioni dell'immagine: 512x512, 1024x1024 e 2048x2048. Le immagini a risoluzione più grande vengono codificate come differenza dall'immagine a risoluzione più bassa. I singoli frame di una codifica gerarchica possono venire ulteriormente codificati con il metodo progressivo.

Ulteriori aggiunte

Una nuova estensione dello standard JPEG approvata in un tempo sucessivo introduce la seguente variante: l'utilizzo di una quantizzazione variabile, che permette una scalatura della marice di quantizzazione differente in parti distinte

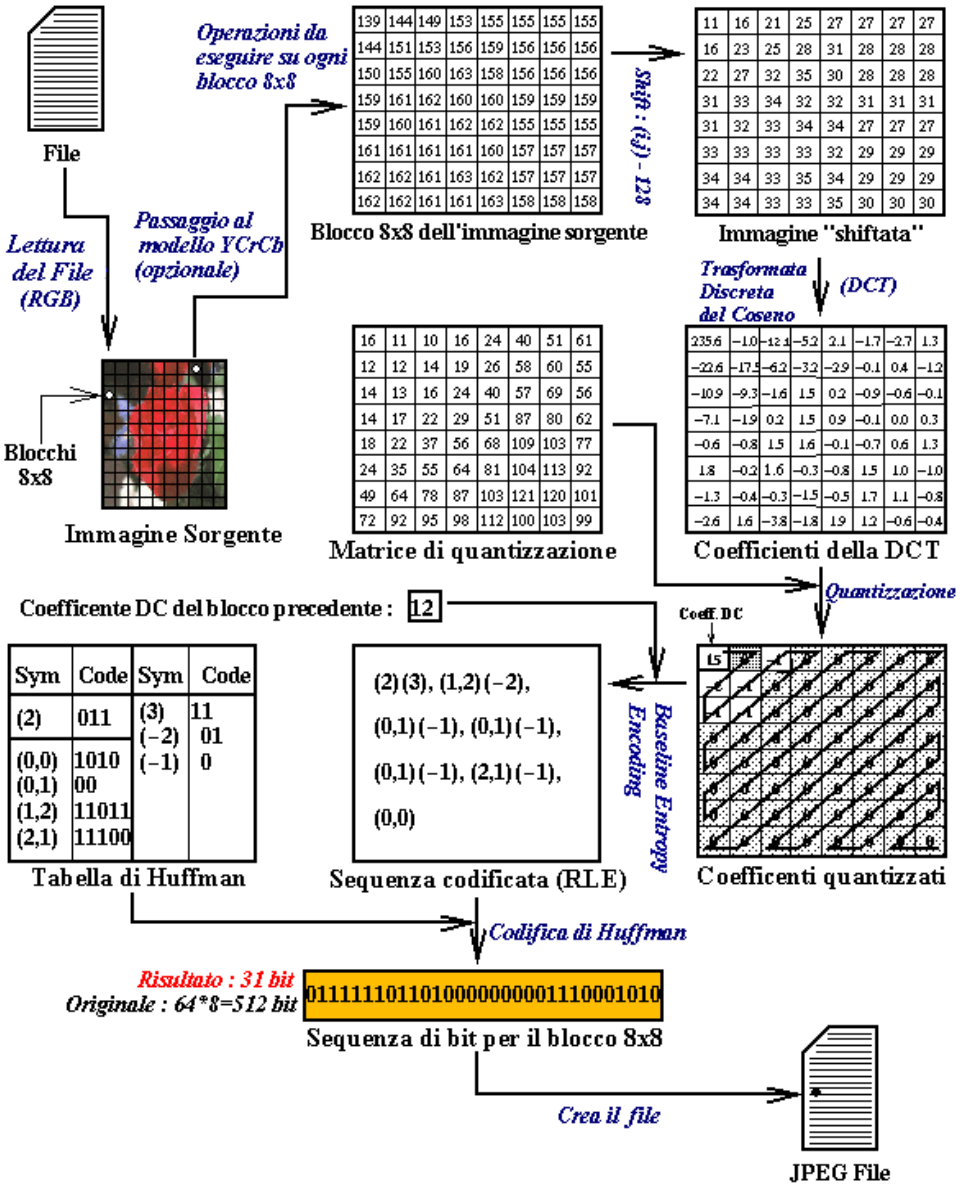
della stessa immagine. In questo modo si possono codificare le parti più importanti dell'immagine con una qualità migliore e quelle meno con una qualità inferiore. Ad ogni codifica di un blocco della DCT verra' introdotto un nuovo parametro che specifica il fattore di scalatura della matrice di quantizzazione per quel blocco.

Lossless JPEG

Questo tipo di codifica non fa uso della DCT, in quanto gli errori di arrotondamento introdotti da questa operazione non le permettono di essere "lossless". Per lo stesso motivo non possono essere usate le operazioni di conversione dello spazio cromatico e la riduzione delle componenti cromatiche dell'immagine. Il "lossless" JPEG codifica la differenza tra ogni pixel ed il valore "predetto" per quel pixel. Il valore "predetto" è dato da una funzione che usa i valori già modificati dei pixel che si trovano sopra e alla sinistra del pixel corrente (posso usare, per esempio, come funzione la media dei pixel). La sequenza cosi ottenuta viene poi codificata con il metodo Huffman. L'uso principale di questa tecnica di codifica è l'abbinamento all'estensione gerarchica del JPEG: il frame finale in una sequenza gerarchica dell'immagine può essere codificato in modo "lossless" per avere una minore perdita di informazione.

L'Algoritmo "baseline"

In questa immagine vediamo i passi fondamentali per la realizzazione dell'algoritmo della codifica JPEG che verrano spiegati in questa sezione.

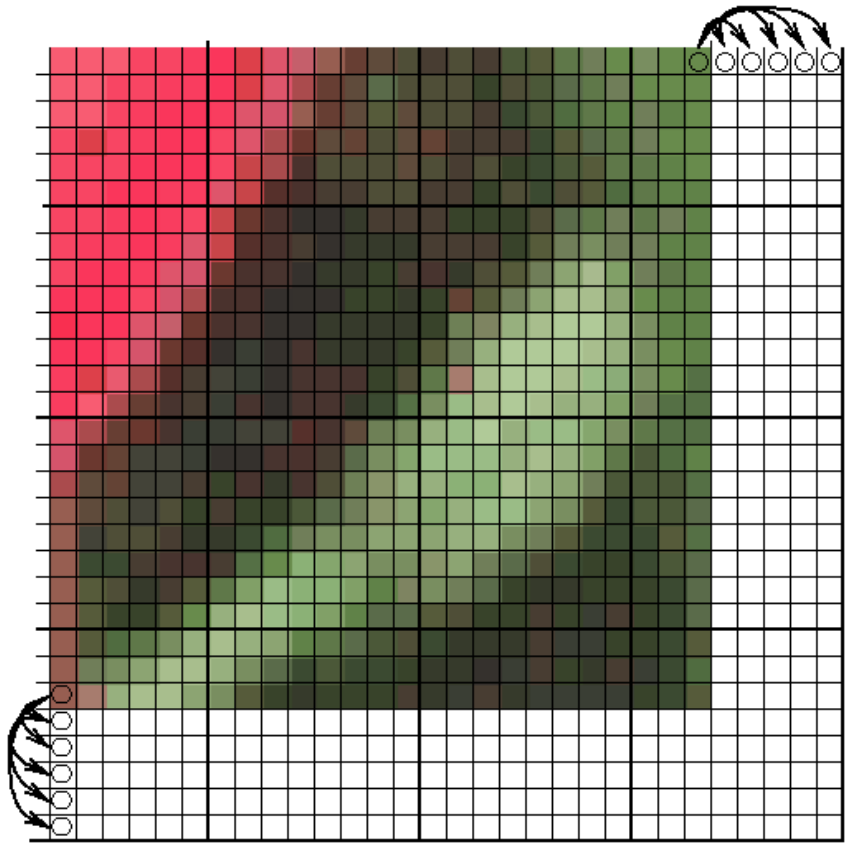

File
Lettura del File (RGB)
Passaggio al modello YCrCb (opzionale)
Operazioni da eseguire su ogni blocco 8x8
Blocchi 8x8
Immagine Sorgente
139 144 149 153 155 155 155 155
144 151 153 156 159 156 156 156
150 155 160 163 158 156 156 156
159 161 162 160 160 159 159 159
159 160 161 162 162 155 155 155
161 161 161 161 160 157 157 157
162 162 161 163 162 157 157 157
162 162 161 161 163 158 158 158
Blocco 8x8 dell'immagine sorgente
Shift : (ij) - 128
11 16 21 25 27 27 27 27
16 23 25 28 31 28 28 28
22 27 32 35 30 28 28 28
31 33 34 32 32 31 31 31
31 32 33 34 34 27 27 27
33 33 33 33 32 29 29 29
34 34 33 35 34 29 29 29
34 34 33 33 35 30 30 30
Immagine "shiftata"
Trasformata Discreta del Coseno
(DCT)
235.6 −1.0 −12.1 −5.2 2.1 −1.7 −2.7 1.3
−22.6 −17.5 −6.2 −3.2 −2.9 −0.1 0.4 −1.2
−10.9 −9.3 −1.6 1.5 0.2 −0.9 −0.6 −0.1
−7.1 −1.9 0.2 1.5 0.9 −0.1 0.0 0.3
−0.6 −0.8 1.5 1.6 −0.1 −0.7 0.6 1.3
1.8 −0.2 1.6 −0.3 −0.8 1.5 1.0 −1.0
−1.3 −0.4 −0.3 −1.5 −0.5 1.7 1.1 −0.8
−2.6 1.6 −3.8 −1.8 1.9 1.2 −0.6 −0.4
Coefficienti della DCT
16 11 10 16 24 40 51 61
12 12 14 19 26 58 60 55
14 13 16 24 40 57 69 56
14 17 22 29 51 87 80 62
18 22 37 56 68 109 103 77
24 35 55 64 81 104 113 92
49 64 78 87 103 121 120 101
72 92 95 98 112 100 103 99
Matrice di quantizzazione
Quantizzazione
Coeff. DC
Coefficente DC del blocco precedente : 12
Coefficenti quantizzati
Baseline Entropy Encoding
(2)(3), (1,2)(−2),
(0,1)(−1), (0,1)(−1),
(0,1)(−1), (2,1)(−1),
(0,0)
Sequenza codificata (RLE)
Sym Code Sym Code
(2) 011 (3) 11
(−2) 01
(−1) 0
(0,0) 1010
(0,1) 00
(1,2) 11011
(2,1) 11100
Tabella di Huffman
Codifica di Huffman
Risultato : 31 bit
Originale : 64*8=512 bit
01111110110100000000001110001010
Sequenza di bit per il blocco 8x8
Crea il file
JPEG File

Lettura del file sorgente

I dati letti dal file sorgente vengono rappresentati, se M e N sono la dimensione orrizzontale e verticale in pixel dell'immagine, come matrici di dimensioni (M,N) per immagini a livello di grigio o di dimensioni (M,N,3) oppure 3 matrici di dimensione (M,N) per immagini a colori. L'algoritmo non supporta la rappresentazione di tipo "colormapped" per le immagini a colori, che quindi vanno trasformate nella rappresentazione RGB.

La DCT verrà applicata a blocchi 8x8 completi quindi, se M o N, cioè la dimensione dell'immagine, non è multiplo di 8 devo aggiungere delle copie dell'ultima riga o dell'ultima colonna alla matrice dell'immagine, sino a che la dimensione della nuova matrice non diventa un multiplo di 8. Questa operazione fa in modo che la tonalità o la luminosità base del blocco 8x8 non vari in modo sostanziale dopo l'applicazione della DCT, fatto che invece può accadere se gli elementi mancanti per completare un blocco 8x8 vengono inizializzati con zero, cioè l'equivalente di un riempimento di nero.

Nella decodifica devo tener conto di questi pixel di riempimento aggiunti ed eliminarli tenedo conto della dimensione originale dell'immagine.



Trasformazione da RGB a YCrCb e successiva riduzione di Cr e Cb

Questa trasformazione è facoltativa. Se voglio cambiare lo spazio cromatico di rappresentazione dell'immagine da RGB a YCrCb devo usare le seguenti formule che vengono applicate alle matrici distinte R,G e B :

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \qquad (2.1)$$

Dopo che ho calcolato le nuove matrici posso ridurre (anche questo è facoltativo) l'informazione nelle matrici cromatiche Cr e Cb nei seguenti modi: 2h2v : riduco le dimensioni di Cr e Cb di 2:1 sia orizzontalmente che verticalmente facendo la media dei 4 pixel confinati. 2h1v : riduco le dimensioni di Cr e Cb solo orizzontalmente di 2:1, la dimensione verticale resta invariata, facendo la media

dei 2 pixel confinanti in orizzontale. La matrice della luminosità Y non deve venire ridotta altrimenti avrei una perdita molto elevata di informazione a livello di qualità percettiva.

"Shift" dell'immagine

Su tutte le matrici R,G,B oppure Y,Cr,Cb oppure Y (solo toni di grigio) bisogna applicare uno shift, cioè passo dalla rappresentazione [0 .. 255] a quella [-128 .. 127] nel seguente modo : X=X-128.

DCT

Divido tutte le matrici in blocchi 8x8 a cui applico la formula per calcolare la trasformata discreta del coseno bidimensionale (DCT):

$$F(u,v) = (1/4)C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)\cos((2i+1)u\pi/16)\cos((2j+1)v\pi/16) \qquad (0.1)$$

dove F(u,v) sarà il coefficente della DCT del blocco 8x8 nella posizione (u,v) della matrice 8x8 che memorizza i coefficenti trasformati, f(x,y) il valore presente nel blocco 8x8 in posizione (x,y) e con:

$$C(x) = \begin{cases} 1/\sqrt{2} & x = 0 \\ 1 & \text{otherwise} \end{cases} \qquad (0.1)$$

Nella decodifica dell'algoritmo del JPEG si applica l'antitrasformata discreta del coseno, descritta dalla seguente formula:

$$f(i,j) = (1/4)\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos((2i+1)u\pi/16)\cos((2j+1)v\pi/16) \qquad (0.1)$$

con:

$$C(x) = \begin{cases} 1/\sqrt{2} & x = 0 \\ 1 & \text{otherwise} \end{cases} \qquad (0.1)$$

Quantizzazione

Ogni blocco 8x8, a cui è stata applicata la DCT, viene quantizzato nel seguente modo :

FQ(u,v)=round int(F(u,v)/Q(u,v))

dove Q(u,v) è la matrice di quantizzazione e F(u,v) è la matrice dei coefficenti della DCT. Posso usare delle matrici di quantizzazione, che non vengono predefinite dallo standard JPEG, ma che sono state calcolate per ottenere dei buoni risultati. Queste matrici variano a seconda della componente cromatica (Y,Cr,Cb,R,G,B...) a cui vengono applicate. Effettuando una scalatura dei valori della matrice di quantizzazione si può ottenere una variazione sulla quantità di informazione eliminata nell' immagine. Il parametro che si potrà specificare dovrà essere compreso tra 1 (molta informazione scartata, immagine di pessima qualità) e 100 (pochissima informazione scartata, immagine simile all'originale) ed agirà nel seguente modo sulla matrice di quantizzazione Q(u,v): prima viene calcolato un valore di scalatura nel seguente modo:

if parametro < 50 then scalatura := {5000/parametro} else scalatura := 200-(parametro)*2

Il valore di scalatura così ottenuto si applicherà ad ogni valore di Q(u,v) nel seguente modo:

Qs(u,v)={(Q(u,v)*scalatura+50)100}

Si ottiene così Qs(u,v) come nuova matrice di quantizzazione 8x8 scalata. Nella fase di decodifica per la dequantizzazione, che ricalcolerà i coefficenti della DCT (non tutti in quanto con la quantizzazione vengono eliminati una parte di questi dati), devo usare la seguente formula:

FQ(u,v)=FQ(u,v)Q(u,v)

Baseline Entropy Encoding

Attraverso la quantizzazione dei coefficienti della DCT delle matrici 8x8 si ottiene una matrice 8x8 con molti elementi nulli che bisognerà rappresentare in modo piu efficiente. I coefficienti non nulli si trovano in alto nell'angolo sinistro della matrice per questo motivo si codifica la matrice 8x8 in una sequenza lineare di 64 elementi ordinati secondo il seguente metodo, detto a zig-zag:

F(1,1); F(1,2); F(2,1); F(3,1); F(2,2); F(1,3); F(1,4); F(2,3); F(3,2); F(4,1); F(5,1); F(4,2);

F(3,3); F(2,4); F(1,5); F(1,6); F(2,5) ;F(3,4); F(4,3); F(5,2); F(6,1); F(7,1) ;F(6,2); F(5,3) ;

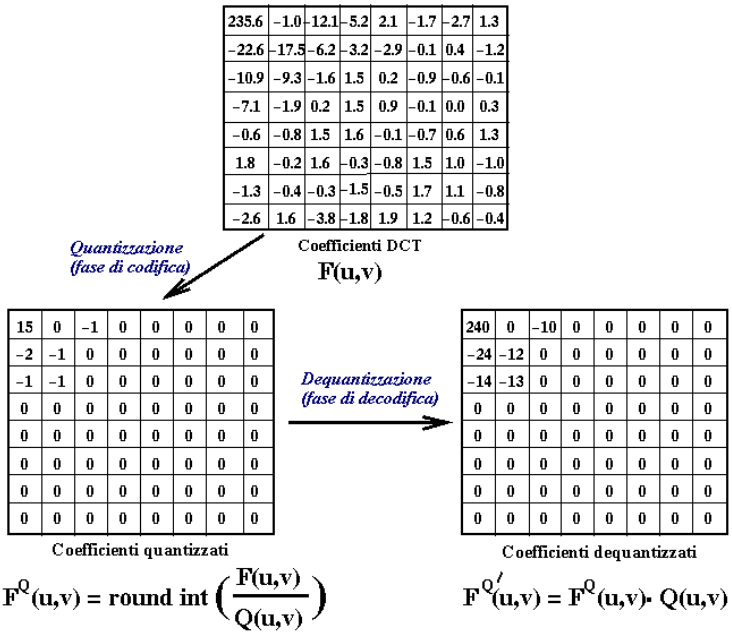
F(4,4); F(3,5); F(2,6) ;F(1,7); F(1,8) ;F(2,7); F(3,6); F(4,5) ;F(5,4) ;F(6,3) ;F(7,2); F(8,1);

F(8,2) ;F(7,3) ;F(6,4) ;F(5,5) ; F(4,6); F(3,7) ;F(2,8) ;F(3,8) ;F(4,7) ; F(5,6); F(6,5) ;F(7,4) ;

F(8,3) ;F(8,4) ; F(7,5); F(6,6) ;F(5,7) ;F(4,8) ;F(5,8) ; F(6,7); F(7,6) ;F(8,5) ;F(8,6) ;F(7,7) ;

F(6,8); F(7,8) ;F(8,7) ;F(8,8) .

Poi si potrà procedere a codificare questi 64 elementi con un metodo di tipo "run length encoding" (RLE), in cui si codificherà in modo efficente le sequenze di zeri. Il primo elemento dei 64 viene chiamato coefficente DC e viene codificato nel seguente modo: per ridurre il coefficente DC, che di solito è un numero abbastanza elevato, si prende il DC del blocco precedente (DCprec) e lo si sottrae a quello del blocco corrente (DCcorr), si otterrà il seguente numero (DCdiff):



if "blocco precedente non esiste, cioe' sono al primo blocco" then (DCprec):=0

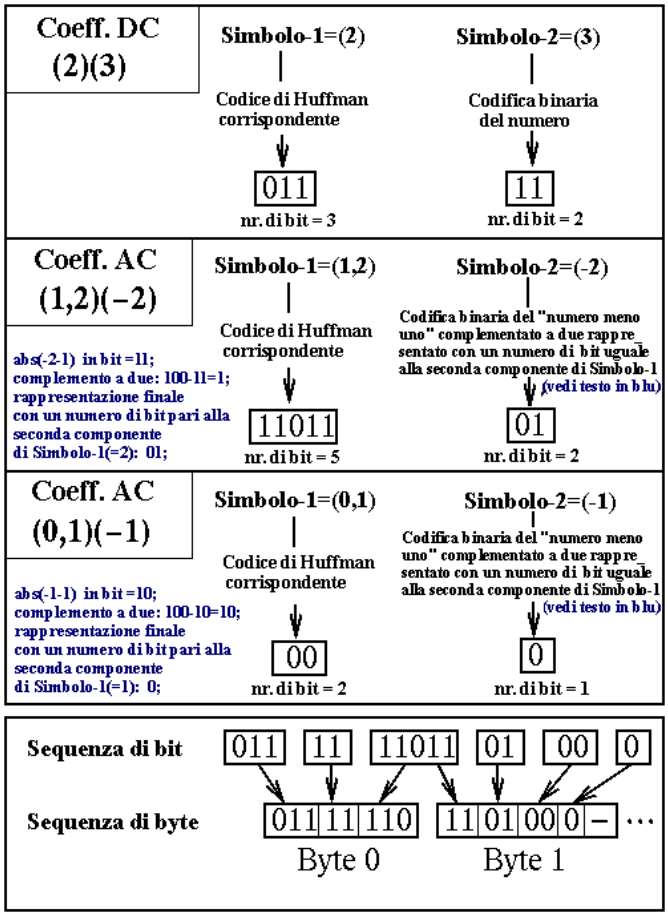
(DCdiff):=(DCcorr)-(DCprec)

che si dovrà codificare ulteriormente nel seguente modo: si costruisce una coppia (Size,Ampiezza) dove Size è il numero di bit necessari a rappresentare il coefficente DCdiff e Ampiezza è il DCdiff stesso. Size può essere calcolato attraverso la tabella rappresentata nella figura sotto. I restanti 63 coefficenti, detti coefficenti AC, vengono rappresentati nel seguente modo: (Runlength,Size)(Ampiezza) dove Runlength è il numero di zeri consecutivi e può rappresentare sequenze di zeri da 0 a 15. Il simbolo speciale (15,0) rappresenta 16 zeri consecutivi e si possono usarne sino ad un massimo di tre in sequenza e poi bisogna comunque inserire il simbolo (Ampiezza) del valore non nullo dopo la sequenza di zeri. La sequenza di zeri che termina senza un numero diverso da zero non viene codificata in quanto il numero di elementi finali, nella decodifica, dovrà comunque essere uguale a 63 , quindi una volta decodificati gli elementi sino al primo non nullo, si aggiunge degli zeri per completare la sequenza sino a 63 elementi. Size ha lo stesso significato che in precedenza. Ampiezza è il primo numero diverso da zero che viene trovato dopo la sequenza di zeri. Dopo questa codifica per tenere traccia di dove finisce un blocco si usa il simbolo (0,0), questo simbolo è quello che codifica anche gli zeri restanti.

Codifica di Huffman

A questo punto, dopo la codifica di ogni blocco 8x8, si ottiene una sequenza di numeri che andranno ulteriormente codificati, per ridurre la quantità di dati, attraverso l'algoritmo di Huffman. L'algoritmo di Huffman utilizza la frequenza statistica con cui un simbolo compare in una sequenza per ottenere una rappresentazione più compatta dei dati. I simboli che compaiono più frequentemente vengono rappresentati con sequenze di bit corte, quelli meno frequenti con sequenze più lunghe. Le tabelle di codifica di Huffman, in cui viene associato ad ogni simbolo della sequenza un codice di Huffman, possono essere calcolate, prima della fase di compressione, attraverso una analisi statistica dei dati da rappresentare oppure si possono usare delle tabelle di Default già stabilite a priori. Lo standard JPEG non stabilisce a priori queste tabelle però mette a disposizione delle tabelle che sono state stabilite attraverso l'analisi di immagini con caratteristiche frequenti. La codifica avviene nel seguente modo: bisogna creare una sequenza di bit che deriva dalla codifica dei diversi simboli che ho ottenuto tramite la "Baseline Entropy Encoding". Definendo (Runlength,Size) con Simbolo-1 e (Ampiezza) con Simbolo-2, a tutte le occorrenze di Simbolo-1 viene associato un codice di Huffman che è distinto a seconda se Simbolo-1 codifica un elemento AC o uno DC, infatti si usano due tabelle di Huffman separate per le due classi di elementi. Questo codice è formato da una sequenza di bit di lunghezza variabile a seconda della frequenza dell'elemento. A questa sequenza si deve aggiungere la codifica del Simbolo-2, che avviene come segue:
Simbolo-2 rappresenta un numero che verrà convertito in un numero binario di un numero di cifre pari a Simbolo-1 se il coefficente è di tipo DC se invece il coefficente è di tipo AC si usa un numero pari alla seconda componente della coppia di Simbolo-1, cioè size.
Se Simbolo-2 è un numero positivo uso la rappresentazione binaria cosı ottenuta e la aggiungo alla sequenza di bit del Simbolo-1, nel caso in cui sia un numero negativo per ottenere la sequenza di bit che lo rappresenta devo procedere nel seguente modo:

al numero negativo si sottrae 1,la rappresentazione positiva del numero così ottenuto viene trasformata in binario, si applica a questo numero binario il complemento a due, di questo numero complementato si prende a partire da destra un numero di cifre pari a Simbolo-1, si ottiene in questo modo una rappresentazione in bit del numero negativo Simbolo-2. Sulla sequenza di bit così ottenuta, che rappresenta la codifica di Huffman dei dati dell'immagine, bisogna applicare un'ultima operazione detta di "Byte stuffing": si suddivide la sequenza di bit in byte; scandendo tutta la stringa byte per byte ogni volta che si trova un byte che rappresenta il numero 0xFF in esadecimale cioè 11111111 in binario, si inserisce nella sequenza il numero 0x00 cioè



00000000 subito dopo; se la stringa di bit non è divisibile per otto, si avrà un byte incompleto che dovrà essere completato con una sequenza di "1".

Scrittura del file

Il file che rappresenta l'immagine JPEG è costituito da una serie di segmenti che iniziano con un "marker" che identifica il tipo di segmento memorizzato. Un marker è costituito da un Byte 0xFF seguito da un altro Byte diverso da 0xFF e da 0x00. Siccome 0xFF viene usato come marker di un segmento non potrà essere presente nei dati codificati con Huffman. Per ovviare a questo inconveniente si utilizza la seguente convenzione : se in tali dati trovo 0xFF inserisco subito dopo il Byte 0x00. Nella fase di lettura dei dati dal file JPEG, cioè la decodifica bisogna eliminare dai dati codificati con Huffman il Byte 0x00 che si trova dopo il Byte 0xFF. Un file JPEG incomincia sempre con un marker SOI e finisce con un marker EOI :

JPEG File	Spazio	Parametri	Descrizione
JPEG File	0xFF 0xD8	SOI	Start of Image Marker *"Reject file" se non esatto*
	Segmenti		
	0xFF 0xD9	EOI	End of Image Marker *"Reject file" se non esatto*

In un file JPEG se si segue il formato JFIF si usa il seguente "application segment":

Segmento	Spazio	Parametri	Descrizione
Application Segment	0xFF 0xE0	APP0	Application marker
	2 Byte INT (MSB LSB)	Lp	Lunghezza del application segment (escluso marker)
	0x4A 0x46 0x49 0x46 0x00	identificatore	Stringa "JFIF" terminata con zero (JPEG File Interchange Format)
	2 Byte	versione	Numero della versione : 1.02 *Ignora / Usa 0x01 0x02*
	1 Byte	unita'	Unita' per densita' X e Y *Ignora / Usa 0*
	2 Byte	Xdensita'	Densita' orizzontale pixel *Ignora / Usa 0x00 0x01*
	2 Byte INT	Ydensita'	Densita' verticale pixel *Ignora / Usa 0x00 0x01*
	1 Byte	Xthumbnail	Quantita' pixel orizzontali Thumbnail *Ignora / Usa 0x00*
	1 Byte	Ythumbnail	Quantita' pixel verticali Thumbnail *Ignora / Usa 0x00*
	? Byte	RR GG BB	valori 24-bit RGB per thumbnail *Ignora / niente*

Ogni matrice di quantizzazione ha un segmento proprio e le viene assegnato un identificatore (0-3). Si possono utilizzare al massimo 4 matrici distinte di quantizzazione nello stesso file. Di solito alla matrice di quantizzazione per le componenti di luminosità viene assegnato l'identificatore Tq=0 (vedi tabella sotto) e quella per le componenti cromatiche Tq=1. I dati di queste matrici verranno poi memorizzati come un lista di componenti con un ordinamento a zig-zag, come si usa per i blocchi della DCT.

Segmento	Spazio	Parametri	Descrizione
Matrice di quantizzazione	0xFF 0xDB	DQT	Define Quantization Table Marker
	2 Byte (MSB LSB)	Lq	Lunghezza segmento(escluso marker = 67 Byte) *"Reject file" se <> 67*
	4 Bit	Pq	Precisione elementi matrice quantizzazione 0 = 8 bit, 1 = 16 bit *"Reject file" se <> 0*
	4 Bit	Tq	Identificatore uso matrice quantizzazione (0-3) (cf. **SOF0:Tqi**)
	64 Byte	Qk	Elementi matrice quantizzazione - 1 Byte per elemento (ordinamento a zigzag !)

Un file JPEG può essere costituito da più frame. Il "Frame Header" mi da informazioni sulle caratteristiche dell'immagine. Il parametro Tqi (vedi tabella sotto) tiene traccia di quale matrice di quantizzazione viene usata dalla componente dell'immagine (Y,Cr o Cb).

Segmento	Spazio	Parametri	Descrizione
Frame Header	0xFF 0xC0	SOF0	Start of Frame, Baseline DCT
	2 Byte INT (MSB LSB)	Lf	Lunghezza frame (escluso marker) **= 8 + 3 * Nf**
	1 Byte	P	Precisione in bit *"Reject file" se <> 8 / Usa 8*
	2 Byte (MSB LSB)	Y	Numero di linee nell'immagine
	2 Byte (MSB LSB)	X	Numero di campioni per linea
	1 Byte	Nf	Numero di componenti dell'immagine nel frame (es. Y, Cr, Cb) Campi **Ci**, **Hi**, **Vi** e **Tqi** ripetuti **Nf** volte!
	1 Byte	Ci	Identificatore del componente (e.g. 1=Y, 2=Cr, 3=Cb)
	4 Bit	Hi	Fattore orizzontale di compressione
	4 Bit	Vi	Fattore verticale di compressione
	1 Byte INT	Tqi	Identificatore della matrice di quantizzazione usata (0-3) (vedi **DQT:Tq**)

Ogni tabella di codici di Huffman utilizzata viene memorizzata in un segmento proprio. Il tipo di codici usati, se per coefficenti DC o AC, viene codificato in Pq (vedi figura sotto) e l'utilizzo in Tq (di solito 0 se componenti di luminosita e 1 per quelli cromatici).

Segmento	Spazio	Parametri	Descrizione
Codici di Huffman	0xFF 0xC4	DHT	Define Huffman Table Marker
	2 Byte (MSB LSB)	Lh	Lunghezza del segmento (escluso marker)
	4 Bit	Pq	Destinazione Codici: 0=Matrice DC , 1=Matrice AC *"Reject file" se <> 0 e <> 1*
	4 Bit	Tq	Identificatore dell'uso dei codici di Huffman (0-1)
	16 Byte	Li	Numero di codici di Huffman di lunghezza i gli Li sono gli elementi della lista BITS
	(Lh-19) Byte	Vij	Valori associati ai codici di Huffmann i Vij sono gli elementi della lista VAL

Lo "Scan Header" si trova prima dei dati codificati con Huffman dell'immagine. Quali codici di Huffman corrispondano a quali componenti (1-3) (Y,Cr o Cb) viene specificato da Tdj e da Taj.

Segmento	Spazio	Parametri	Descrizione
Scan Header	0xFF 0xDA	SOS	Start of Scan
	2 Byte (MSB LSB)	Ls	Lunghezza dello Scan header (escluso marker) = 6 + 2 * Ns
	1 Byte	Ns	Numero di componenti dell'immagine nello scan. Campi **Csj**, **Tdj** e **Taj** ripetuti Ns volte.
	1 Byte	Csj	Identificatore della componente dello scan (1-3)
	4 Bit	Tdj	Codici di Huffman relativi a DC (0-1) (vedi **DHT:Pq** & **Tq**)
	4 Bit	Taj	Codici di Huffman relativi a AC (0-1) (vedi **DHT:Pq** & **Tq**)
	1 Byte	Ss	Inizio selezione spettrale : 0 *"Reject file" se <> 0*
	1 Byte	Se	Fine selezione spettrale: 63 (=0x3F) *"Reject file" se <> 63*
	4 Bit	Ah	Approssimazioni successive posizione del "bit high" *"Reject file" se <> 0*
	4 Bit	Al	Approssimazioni successive posizione del "bit low" *"Reject file" se <> 0*

Dopo lo "Scan Header" vengono scritti i dati veri e propri dell'immagine codificati con Huffman. I segmenti del file JPEG vanno inseriti nel file nel seguente ordine:

Segmento	Spazio
SOI Marker	vedi sopra
Segmento **APP0**	vedi sopra
Segmento **DQT**	Matrice di quantizzazione per la componente di luminosita' (ordinata a zigzag)
Segmento **DQT**	Matrice di quantizzazione per la componente cromatica (ordinata a zigzag)
Segmento **SOF0**	vedi sopra
Segmento **DHT**	Codici di Huffman - DC luminosita'
Segmento **DHT**	Codici di Huffman - DC cromatici
Segmento **DHT**	Codici di Huffman - AC luminosita'
Segmento **DHT**	Codici di Huffman - AC cromatici
Segmento **SOS**	vedi sopra
Scan Data	I dati dell'immagine codificati con Huffamn
EOI Marker	vedi sopra

Nella figura che segue si può vedere una rapprese ntazione in esadecimale del file binario JPEG: