

Compressione delle immagini (codifica JPEG)



ALBERTO BELUSSI
ANNO ACCADEMICO 2009/2010

Motivazioni a favore della compressione

2

- Es. Si consideri di voler proiettare un'ora di un film mediante un dispositivo (CD-ROM, per esempio). Assumendo video frame con risoluzione di 620x560 pixel e 24 bpp, ogni frame richiederebbe circa 1 MB di memoria. Se il video animato necessita di 30 frame al secondo, occorrerà trasmettere 30MB/sec., o 108 GB in un'ora. Anche se fosse disponibile una quantità di memoria sufficiente, tale visione non sarebbe possibile in tempo reale a causa della bassa velocità di trasmissione che l'attuale tecnologia fornisce.
- Es. La larghezza di banda della rete di trasmissione classica (Ethernet, token ring) e' dell'ordine di **decine di Mb/sec, troppo** bassa anche per la trasmissione di un solo minuto video non compresso. L'ordine sale a centinaia di Mb/sec per reti ATM e FDDI, ma anche in questo caso solo poche sessioni multimediali simultanee sono possibili per trasmissioni di dati non compressi

Tipi di compressione

3

Senza perdita (*lossless*): *permette di ricostruire perfettamente la rappresentazione dell'oggetto originale, conserva infatti tutti i bit di dati (es. codifica di Huffman)*

Con perdita (*lossy*): *permette di ricostruire solo in parte la rappresentazione dell'oggetto originale (alcuni bit di dati vengono scartati). Le tecniche "lossy" sono classificate in:*

- Tecniche basate su **predizione**
- Tecniche orientate alla **frequenza (Discrete Cosine Transform - DCT)**
- Tecniche orientate all'**importanza**
- Tecniche di compressione "ibride" combinano alcuni approcci, come DCT e quantizzazione vettoriale

Rappresentazione delle immagini

4

- Dal punto di vista fisico un'immagine digitale è un **vettore 2-D** di elementi, detti **pixel**, in cui la prima coordinata rappresenta la larghezza dell'immagine mentre la seconda l'altezza. Un'immagine larga 500 e alta 300 (500x300) avrà dunque una dimensione totale di 150.000 pixel
- La **precisione** di un'immagine è espressa dal numero di bit usati per rappresentare ogni pixel (**bpp**) e determina il numero di colori che l'immagine stessa può contenere.

Rappresentazione delle immagini

5

In base alla precisione le immagini si possono classificare in:

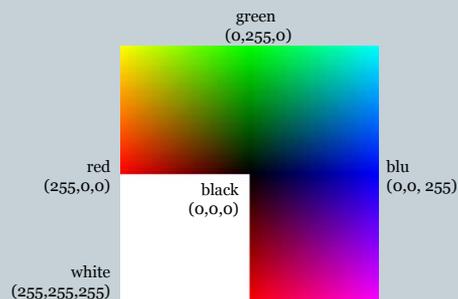
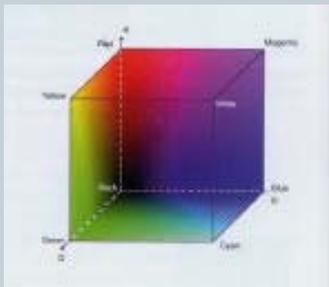
- Immagini **binarie**, rappresentate da 1 bpp (= 2 colori). Includono fotografie in bianco e nero
- Immagini **computerizzate**, rappresentate da 4 bpp (16 colori). Comprendono icone ma non immagini complesse
- Immagini **su scala di grigio**, rappresentate da 8 bpp (256 colori). Questa precisione non è ancora sufficiente per rappresentare fotografie o immagini scannerizzate
- Immagini **a colori**, rappresentate da 16, 24 o più bpp (da 32.768 a 16.777.216 colori). Includono fotografie e immagini scannerizzate.

Rappresentazione del colore

6

Sistemi di rappresentazione di colore (color space)

- **RGB**: Il colore si ottiene sommando tre colori primari: *red* (*R*), *green* (*G*) e *blue* (*B*). La linea del cubo in cui $R=G=B$ specifica i valori di grigio compresi tra il nero e il bianco.

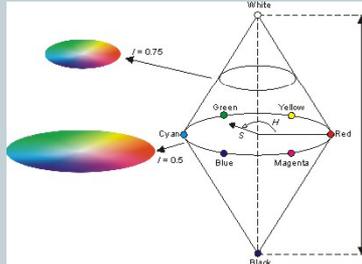


Rappresentazione del colore

7

Sistemi di rappresentazione di colore (color space)

- **HSI:** Il colore è scomposto in tre componenti: *hue* (H), rappresentante il colore, *saturation* (S), rappresentante la quantità di colore presente, e *intensity* (I), indicante la luminosità. Lo spazio in cui i tre valori sono resi graficamente è un **bi-cono in** cui l'asse centrale rappresenta i valori di grigio.



Rappresentazione del colore

8

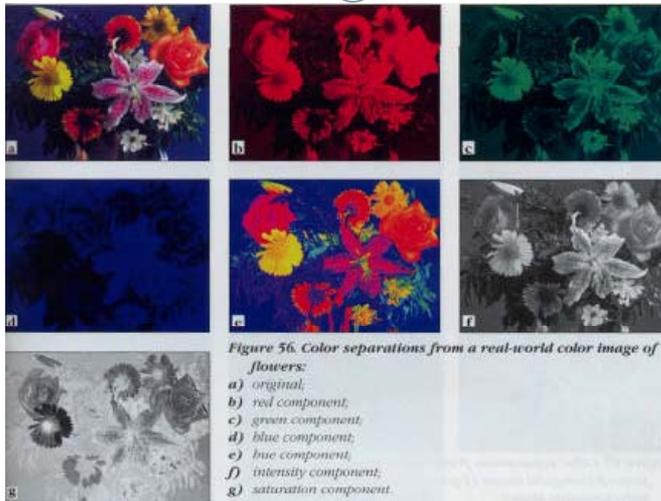


Figure 56. Color separations from a real-world color image of flowers:

- original;
- red component;
- green component;
- blue component;
- hue component;
- intensity component;
- saturation component.

Formati di un'immagine

9

GIF (*Graphics Interchange Format*)

- Creato nel **1987**, è uno dei formati standard più usati e supportati. Ogni browser grafico può rappresentarlo.
- La compressione del formato GIF è di tipo *lossless*. Supporta colori a **8 bit**: un'immagine GIF può avere **256** colori.
- Nel **1989** il formato viene aggiornato (**GIF 89a**) **includendo tre** varianti:
 - GIF **normale**
 - GIF **trasparente**
 - GIF **animata**

Formati di un'immagine

10

JPEG (*Joint Photographic Experts Group*)

- Standard creato nel **1990** con l'**intenzione di migliorare e** sostituire i formati di immagine già esistenti. Ogni browser grafico è in grado di rappresentarlo.
- La compressione JPEG è di tipo "ibrido", combina infatti DCT e quantizzazione
- Supporta colori a **8 e a 24 bit**: un'immagine JPEG può avere da 256 a oltre 16.7 milioni di colori.
- Un file JPEG occupa meno spazio di un file GIF della stessa immagine, JPEG usa infatti un rapporto di compressione più elevato. L'algoritmo di compressione JPEG e' più complesso rispetto a GIF e richiede un tempo di decompressione dell'immagine maggiore.

Codifica JPEG

11

JPEG standard fornisce quattro modi di operare:

- Codifica **sequenziale** basata su DCT: ogni componente dell'immagine è codificata in un singolo scan (**left-to-right, top-to-bottom**)
- Codifica **progressiva** basata su DCT: l'immagine è codificata in scan successivi al fine di produrre una decodifica della stessa più veloce in caso di tempi di trasmissione elevati
- Codifica **lossless**: l'immagine è codificata in modo da garantirne un'esatta riproduzione
- Codifica **gerarchica**: l'immagine è codificata in multipla risoluzione

Codifica JPEG sequenziale

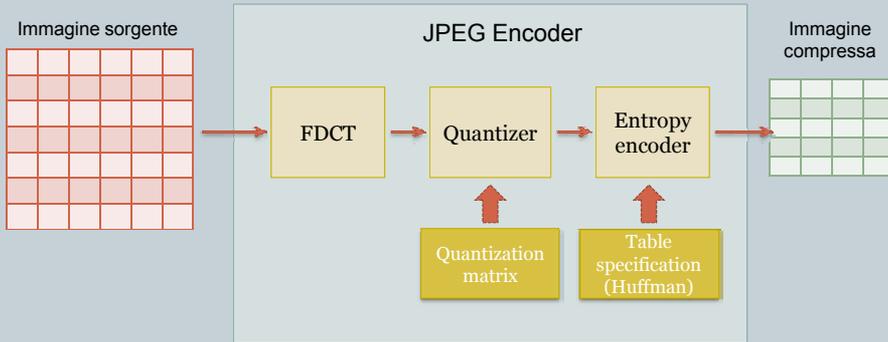
12

Codifica JPEG sequenziale su singola componente

- **JPEG encoder**
 - *Forward Discrete Cosine Transform (FDCT)*
 - *Quantizer*
 - *Entropy Encoder*
- **JPEG decoder**
 - *Entropy Decoder*
 - *Dequantizer*
 - *Inverse DCT (IDCT)*

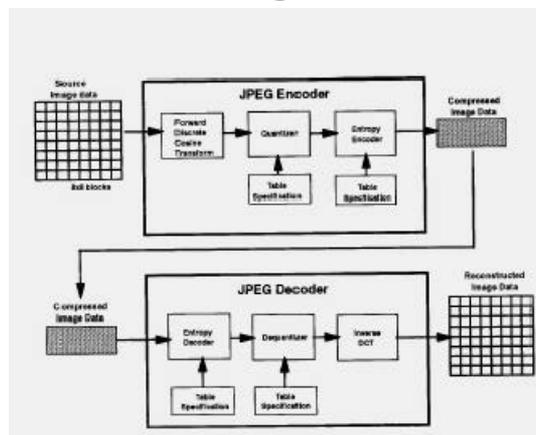
Schema codifica JPEG

13



Schema completo

14



JPEG - Encoder

15

Passi principali

Fase 1 – Partizionamento dell'immagine

- a. I pixel dell'immagine originale sono raggruppati in **blocchi di 8x8** elementi.
- b. Gli elementi di ciascuno blocco sono **shiftati** da interi senza segno, nel range $[0, 2^p - 1]$, a interi con segno, nell'intervallo $[-2^{p-1}, 2^{p-1} - 1]$ (con p , precisione dell'immagine).

JPEG – Encoder (FDCT)

16

Fase 2 – Trasformazione FDCT

- a. Ogni blocco 8x8 può essere visto come un segnale discreto a 64 punti. Tale segnale è dato in input alla trasformata FDCT, la cui definizione matematica è la seguente:

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

dove $C(u)$ e $C(v)$ sono così definiti:

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{se } u, v = 0 \\ 1 & \text{altrimenti} \end{cases}$$

JPEG Encoder (FDCT)

17

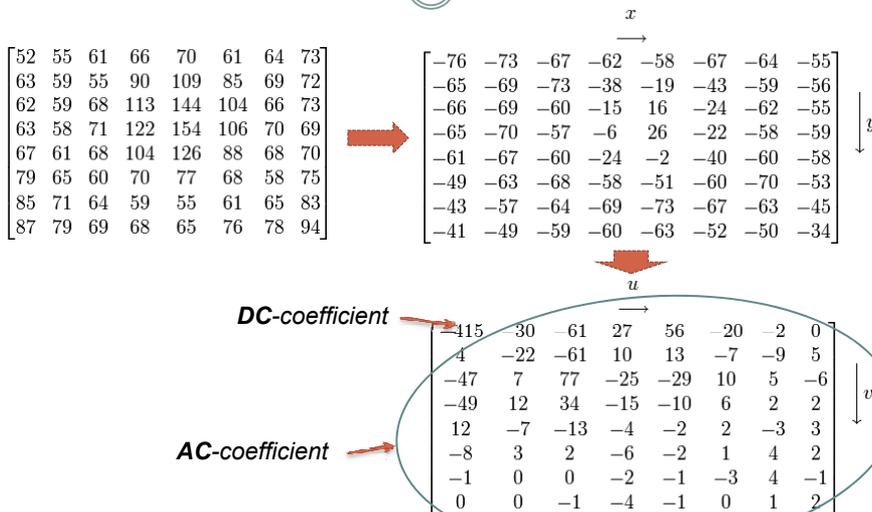
L'output della FDCT è un insieme di 64 coefficienti (*DCT coefficient*) che rappresentano le ampiezze dei segnali base in cui il segnale originale è stato scomposto.

Il coefficiente con valore di frequenza **nullo** in entrambe le dimensioni, $F(0, 0)$, è chiamato **DC-coefficient** (*Direct*) e i restanti 63 **AC-coefficient** (*Alternating*)

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

Esempio FDCT

18



JPEG Encoder (FDCT)

19

FDCT comprime l'immagine concentrando la maggior parte dell'energia contenuta nel segnale nelle componenti di bassa frequenza.

Si tratta di compressione *lossless*.

E' da notare che, per un tipico blocco, buona parte delle frequenze spaziali o sono nulle o sono prossime allo zero e quindi è possibile una ulteriore compressione che si realizza nella fase successiva.

JPEG Encoder (Quantizzazione)

20

Fase 3 – Quantizzazione

- I 64 coefficienti DCT sono quantizzati uniformemente mediante una Tabella di Quantizzazione a 64 elementi rappresentati da interi nell'intervallo [1-255].
- La quantizzazione riduce l'ampiezza dei coefficienti DCT il cui contributo è nullo o comunque basso per la qualità dell'immagine. La quantizzazione scarta dunque informazione che non è rilevante al fine della visualizzazione. Si tratta di compressione *lossy*.

JPEG Encoder (Quantizzazione)

21

- La quantizzazione è un **mapping multi-a-uno** (*lossy*) e si calcola mediante l'equazione:

$$F_q(u, v) = \text{Round} \left(\frac{F(u, v)}{Q(u, v)} \right)$$

- dove $Q(u, v)$ rappresenta il coefficiente di quantizzazione associato al coefficiente DCT.

JPEG Encoder (Quantizzazione)

22

Tabelle di quantizzazione: esempio.

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

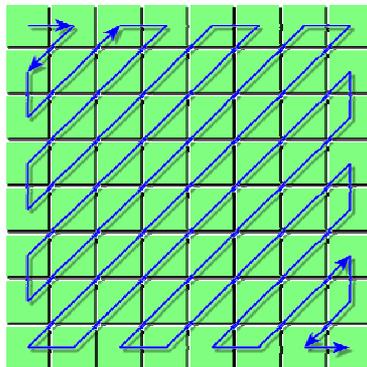
JPEG Encoder (Quantizzazione)

23

- A questo punto il coefficiente DC viene trattato separatamente dai 63 coefficienti AC:
 - esiste infatti una forte correlazione tra i valori DC dei blocchi (8x8) adiacenti
 - il coefficiente quantizzato viene quindi codificato come differenza rispetto al valore DC del blocco adiacente
- Infine, tutti i coefficienti quantizzati (DC+AC) sono ordinati in una sequenza a “**zig-zag**”. Questo ordine facilita la successiva fase di *Entropy Encoding* ponendo i coefficienti di bassa frequenza (più probabilmente non nulli) prima di quelli di alta frequenza.

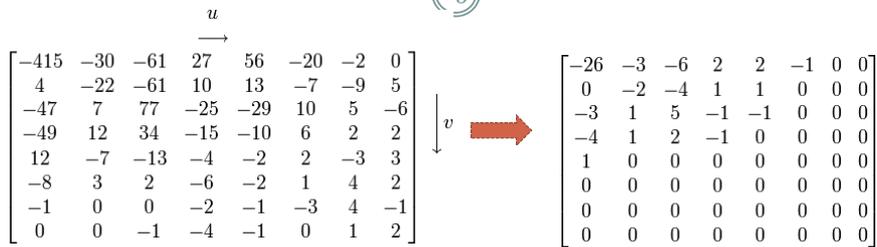
JPEG Encoder (Quantizzazione)

24



JPEG Encoder (Quantizzazione)

25



-26 -3 0 -3 -2 -6 2 -4 1 -4 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0 -1 -1 0 0

JPEG Encoder (Entropy Encoder)

26

Fase 3 – Entropy encoder

- *L'Entropy Encoder aggiunge ulteriore compressione (lossless) ai coefficienti DCT quantizzati codificandoli in modo più compatto.*
- Il processo si basa sulla **codifica di Huffman**:
 - La sequenza “zig-zag” di coefficienti DCT quantizzati è convertita in una **sequenza intermedia di simboli**
 - La sequenza intermedia è codificata in una **sequenza binaria** mediante l'uso di tabelle (**Tabelle di Huffman**)

JPEG Encoder (Entropy Encoder)

27

- Nella sequenza intermedia di simboli, ogni coefficiente AC è rappresentato da una coppia di simboli:
 - Simbolo-1 (**RUNLENGTH, SIZE**) e
 - Simbolo-2 (**AMPLITUDE**)
- **RUNLENGTH** rappresenta il numero di valori nulli che precedono un valore non nullo; **SIZE** indica il numero di bit necessari a codificare l'ampiezza (**AMPLITUDE**).

JPEG Encoder (Entropy Encoder)

28

- Esempio: Se la sequenza di coefficienti AC è:
0, 0, 0, 0, 0, 0, 476
allora il simbolo rappresentativo per il coefficiente AC 476 risulta essere ($2^9=512$):
(6,9)(476)

JPEG Encoder (Entropy Encoder)

29

- Nella sequenza intermedia di simboli il coefficiente DC è rappresentato invece dalla coppia:
 - Simbolo-1 (**SIZE**) e
 - Simbolo-2 (**AMPLITUDE**)

JPEG Encoder (Entropy Encoder)

30

- La sequenza di simboli intermedia viene poi convertita in sequenza binaria:
 - Ogni Simbolo-1 (sia DC che AC) è codificato con un Variable-Length Code (VLC)
 - I Simboli-2 sono codificati usando un Variable-Length Integer (VLI)

JPEG Encoder (Quantizzazione)

31

-26 -3 0 -3 -2 -6 2 -4 1 -4 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0 -1 -1 0 0



Sequenza simboli intermedia

(5)(-26),(0,2)(-3),(1,2)(-3),(0,2)(-2),(0,3)(-6),(0,2)(2),(0,3)(-4),(0,1)(1),(0,3)(-4),(0,1)(1),(0,1)(1),(0,3)(5),(0,1)(1),(0,2)(2),(0,1)(-1),(0,1)(1)(0,1)(-1),(0,2)(2),(5,1)(-1),(0,1)(-1),(0,0)



Sequenza di bit (dipende dalle tabelle di Huffman)

(01)....

JPEG Decoder

32

Il *JPEG Decoder* esegue i **processi inversi** rispetto quelli applicati nel *JPEG Encoder* invertendone l'ordine:

- *Entropy Decoder*: prende la sequenza binaria e la trasforma, prima in sequenza intermedia di simboli, poi nella sequenza a “zig-zag” rappresentante i 64 coefficienti DCT quantizzati.
- *Dequantizer*: prende i 64 coefficienti quantizzati DCT e li trasforma in 64 coefficienti DCT semplici seguendo l'espressione:

$$F_q^{INV}(u, v) = F_q(u, v)Q(u, v)$$

JPEG Decoder

33

- *Inverse DCT (IDCT)*: prende in input i 64 coefficienti DCT e ricostruisce in output il segnale discreto di 64 punti rappresentante un blocco 8x8 dell'immagine originale.

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

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{se } u, v = 0 \\ 1 & \text{altrimenti} \end{cases}$$

Esempi di compressione

34



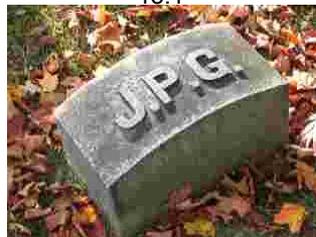
2.6:1



15:1



23:1



46:1