

Altre organizzazioni

- Fra le molte altre organizzazioni possibili, alcune si incontrano con particolare frequenza:
 - bianco e nero
 - ogni pixel è un bit: 1=nero, 0=bianco (o viceversa)
 - usata per gestire testo come immagine (es., fax)
 - scala di grigi
 - ogni pixel è un valore da 0% (nero) a 100% (bianco)
 - tipicamente, si usano 8 bit, quindi 256 grigi
 - usata quando la destinazione finale è una stampa in bianco e nero (es., foto sui giornali)

Trasformazioni fra formati

- Esistono delle trasformazioni standard fra i diversi formati che abbiamo discusso
 - Il formato più “ricco” di informazioni (e più fedele) è il direct color
 - Può essere ridotto a indicizzato scegliendo, per esempio, i 256 colori che appaiono più frequentemente, oppure quelli “medi” fra gli altri simili (che minimizzano gli errori di approssimazione)

Trasformazioni fra formati

- Esistono delle trasformazioni standard fra i diversi formati che abbiamo discusso
 - I formati a colori possono essere ridotti a scala di grigi trasformando ogni colore in un valore di **luminanza**
 - Si moltiplica ogni componente per una costante fissa, che dipende dalla sensibilità (media) dell'occhio umano ai vari colori
 - grigio = $0.299 \times \text{rosso} + 0.587 \times \text{verde} + 0.114 \times \text{blu}$

Trasformazioni fra formati

- Infine, la scala di grigi può essere ridotta a b/n stabilendo un **valore soglia**
 - tutti i grigi più chiari della soglia diventano bianco
 - tutti i grigi più scuri della soglia diventano nero
- Ovviamente le trasformazioni inverse sono possibili, ma non si recupera l'informazione persa!

Trasformazioni fra formati

- Alcuni esempi di diversi valori soglia:



Originale
(in scala di grigi)



soglia 63 (25%)



soglia 127 (50%)



soglia 163 (64%)



soglia 192 (75%)

Esempi



- Colore diretto, **86.700** byte
 - 170x170, 24 bit colore diretto
- Colore indicizzato, **29.668** byte
 - 170x170, 24 bit colore, 256 colori
- Scala di grigi, **28.900** byte
 - 170x170, 8 bit colore, 256 grigi
- Bianco/nero, **3.613** byte
 - 170x170, 1 bit colore, b/n

Ancora sull'occupazione di memoria

- I calcoli sull'occupazione di memoria che abbiamo fatto riguardano la dimensione delle immagini **in memoria** – per esempio, mentre vengono visualizzate
- Per memorizzare le immagini su disco, o trasmetterle via rete, è conveniente usare **algoritmi di compressione** per ridurre la dimensione

Algoritmi di compressione

- In generale, possiamo dividere gli algoritmi di compressione in due grandi famiglie:
 - algoritmi **senza perdita** (*lossless*): consentono di ricostruire esattamente i dati di partenza
 - algoritmi **con perdita** (*lossy*): si perde una parte dell'informazione; non è più possibile ricostruire esattamente i dati di partenza

Algoritmi lossless

- Gli algoritmi di compressione senza perdita applicati alle immagini sono gli stessi usati nei normali programmi di compressione
 - per esempio, algoritmi di Run Length Encoding, Huffman, Lempel-Ziv-Welch, compressione aritmetica
 - usati in PKZip, gzip, compress, lha, RAR, ARJ, ecc.
 - I **formati di file** per le immagini specificano uno o più algoritmi da adottare

Algoritmi lossy

- Gli algoritmi lossy sono invece progettati esplicitamente per le applicazioni alle immagini
- I due principali sono:
 - **JPEG** (Joint Photographic Expert Group), famosissimo algoritmo basato sulla trasformata inversa del coseno e sull'eliminazione delle frequenze “alte” (spaziali e colorimetriche), comunque invisibili all'occhio umano
 - **Wavelet**, algoritmo molto efficiente ma “sperimentale” a causa dell'alto costo computazionale

Algoritmi lossy

- Quando si decide di applicare un algoritmo di compressione lossy a un'immagine, è in genere possibile specificare un **fattore di qualità**
- Ad elevati fattori di compressione (e quindi, forti riduzioni di dimensioni) corrispondono forti perdite di qualità
- In genere, **non si usano algoritmi lossy** se l'immagine deve ancora essere “lavorata”.

Esempio (file JPEG, lossy)

170x170x24
dimensione in
memoria:
86.7 Kb



originale
q=100
24.8 Kb



q=75
8.1 Kb

q=50
5.6 Kb

q=25
3.7 Kb

q=5
1.4 Kb

Esempio (file PNG, lossless)

170x170x24
dimensione in
memoria:
86.7 Kb



originale



c=0
85.1 Kb

c=2
56.9 Kb

c=5
53.9 Kb

c=9
53.3 Kb

Esempio (file PNG, lossless)

- Con gli algoritmi lossless, è a volte possibile specificare un **fattore di compressione**
- Visto che con gli algoritmo lossless non si perde **mai** in qualità, la scelta di una compressione maggiore **non** peggiora l'immagine
 - aumenta solo (in maniera impercettibile) il tempo per caricare e salvare l'immagine
- Conviene quindi usare sempre la compressione maggiore!

Principali formati grafici

- Con compressione lossless:
 - TIFF (Tagged Image File Format)
 - GIF (Graphics Interchange Format)
 - PNG (Portable Network Graphics)
 - BMP (Windows Bitmap)
 - TGA (schede grafiche Targa)
- Con compressione lossy:
 - JPEG (Joint Photographic Expert Group)

Prossimi argomenti

- Cenni sulle tecniche di compressione
 - JPEG
- Informazioni sui formati grafici più comuni
- Cenni sull'elaborazione digitale di immagini
 - Concetti e algoritmi
 - Tool di uso comune: Photoshop, GIMP

Riferimenti

- http://www.diodati.org/scritti/2001/algoritmi/algor_stam.asp contiene un conciso ma leggibile excursus delle principali tecniche di compressione di immagini
- <http://www.freeonline.org/art/a-128/> raccoglie un po' delle sigle che abbiamo incontrato, con relative (succinte) spiegazioni
- Un elenco di colori con la relativa codifica RGB è accessibile all'URL <http://riemann.unica.it/studenti/guida/colori.html> (ma se ne trovano centinaia di altri, più o meno estesi)
 - È possibile selezionare un qualunque colore RGB e vedere il suo codice a <http://www.allprofitallfree.com/color-wheel2.html>
- www.gimp.org è la home page di GIMP, il programma di grafica bitmap da cui abbiamo tratto le immagini sui selettori colore (e che useremo in laboratorio)

Algoritmi di compressione per le immagini

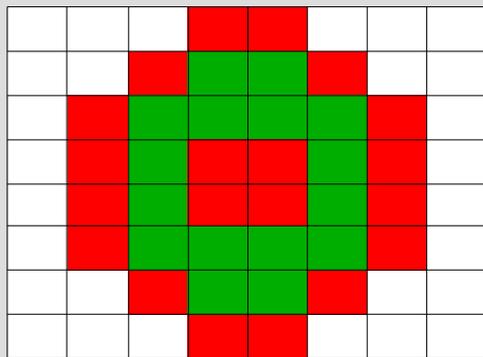
- Abbiamo visto nella lezione precedente che le immagini possono essere **compresse** in vari modi, per ridurre l'occupazione di memoria (su disco o rete)
 - compressione lossless – non si perdono dati
 - compressione lossy – si perdono i dati “meno importanti”

Algoritmo RLE

- L'**RLE** (*run length encoding*, codifica della lunghezza delle sequenze) è un algoritmo lossless particolarmente semplice
- Idea di base: quando trovo una serie di pixel dello stesso colore, codifico solo **la lunghezza** della serie e **il colore**
- Per immagini di tipo “fumetto” o geometriche, funziona molto bene!

Algoritmo RLE

- Esempio:



8x8x24 = 1536 bit

B	B	B	R	R	B	B	B	3B	2R	3B						
B	B	R	V	V	R	B	B	2B	1R	2V	1R	2B				
B	R	V	V	V	V	R	B	1B	1R	4V	1R	1B				
B	R	V	R	R	V	R	B	1B	1R	1V	2R	1V	1R	1B		
=	B	R	V	R	R	V	R	B	RLE →	1B	1R	1V	2R	1V	1R	1B
B	R	V	V	V	V	R	B	1B	1R	4V	1R	1B				
B	B	R	V	V	R	B	B	2B	1R	2V	1R	2B				
B	B	B	R	R	B	B	B	3B	2R	3B						

40x(4+24) = 1120 bit

- Se l'immagine ha ampie aree di colore uguale, si ottengono grossi risparmi
- Se viceversa ogni punto ha un colore diverso, si può anche peggiorare!

Un teorema importante sulla compressione

- **Teorema:**

- non esiste un algoritmo di compressione che “funzioni” **sempre** (cioè, che riduca la dimensione dell'input in tutti i casi)

- **Dimostrazione** (per assurdo, approssimata):

- supponiamo che un tale algoritmo esista (chiamiamolo A), e indichiamo con $\#$ la lunghezza di un dato
- dato un input i , $A(i)$ deve essere più piccolo di i , ovvero $\#A(i) < \#i$
- ma $A(i)$ è a sua volta un dato, quindi $\#A(A(i)) < \#A(i) < \#i \dots$
- se $\#i = k$, dopo al più k passi sono arrivato a 0: come posso da una dato lungo 0 ricostruire l'input originale i ?

Algoritmi Huffman e LZW

- Si tratta di due algoritmi lossless usati in molte applicazioni
 - **non** sono specializzati per le immagini
 - in entrambi i casi, il grado di compressione non è direttamente legato al contenuto dell'immagine
 - essendo lossless, non c'è mai perdita di qualità: quando il formato li supporta, è sempre* meglio usarli

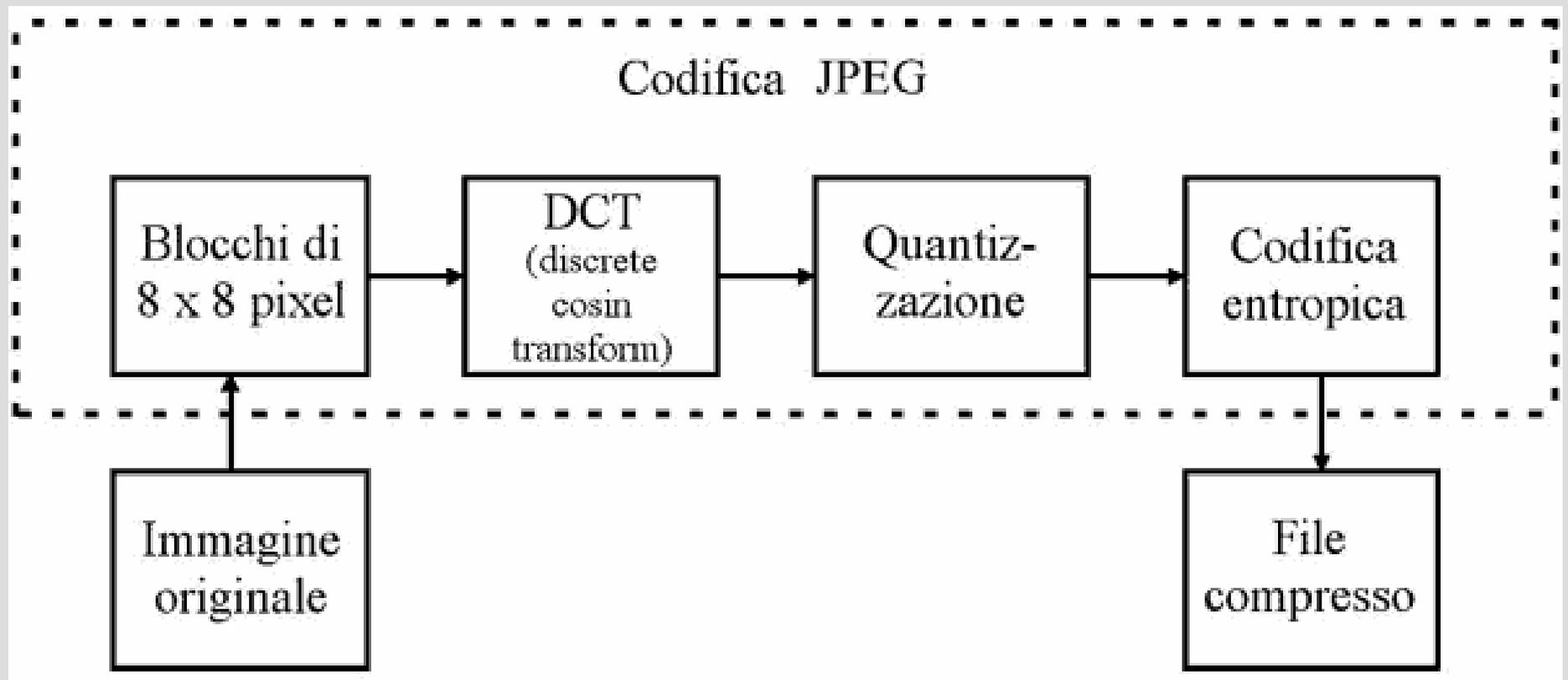
* ma si ricordi il teorema precedente...

Algoritmo JPEG

- JPEG è un algoritmo **lossy** specializzato per le immagini
- Funziona particolarmente su immagini con molti sfumati (fotografie, incarnati, paesaggi)
- Quando l'immagine ha un forte contrasto o passaggi bruschi di colore, si possono notare dei difetti (detti *artefatti*)

Algoritmo JPEG

- Il processo prevede 4 passi
 - lo scopo è togliere informazione che *comunque* l'occhio non noterebbe



Algoritmo JPEG

trasformazione colori

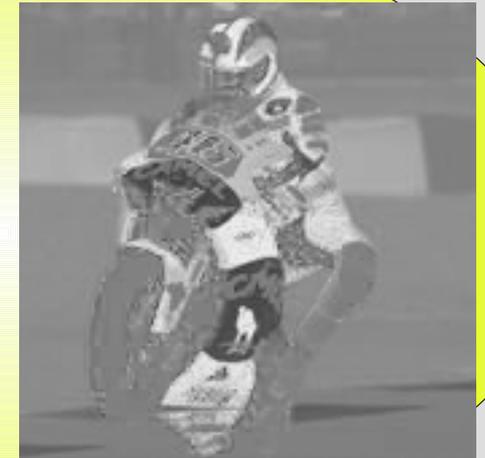
- L'immagine viene dapprima trasportata in un altro *spazio colore*
 - si cambia il modello colore da RGB (o altro) a **YUV**, che codifica **Crominanza** (colore, 2 valori) + **Luminanza** (luminosità, 1 valore)
 - l'occhio umano è infatti più sensibile alla luminosità che al particolare colore, quindi nei passi successivi si può “perdere” in crominanza senza danni, mentre le perdita di luminanza è sensibile

Algoritmo JPEG

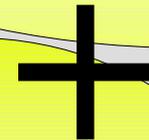
trasformazione colori



Immagine originale



Componenti crominanza



Componente luminanza

Algoritmo JPEG

riduzione componenti

- La componente luminanza viene lasciata invariata
- Le componenti di crominanza possono essere “tagliate”, sostituendo blocchi di 2x1 o 2x2 pixel (o più) con un solo valore, dato dalla media dei componenti eliminati
- Questa operazione, da sola, riduce già del 50%-60% la dimensione!
 - Lo specifico taglio è indicato con codici come 4-1-1, 4-2-1, 4-2-2, ecc.

Algoritmo JPEG

separazione in blocchi

- Il passo successivo consiste nel dividere l'immagine in blocchi di 8x8 pixel
- Un'immagine di tipo televisivo (640x512 pixel) richiede 80x64 blocchi
- A volte la “scalettatura” causata dai blocchi è ben visibile!
 - quasi sempre dovuto ad alti fattori di compressione



Algoritmo JPEG

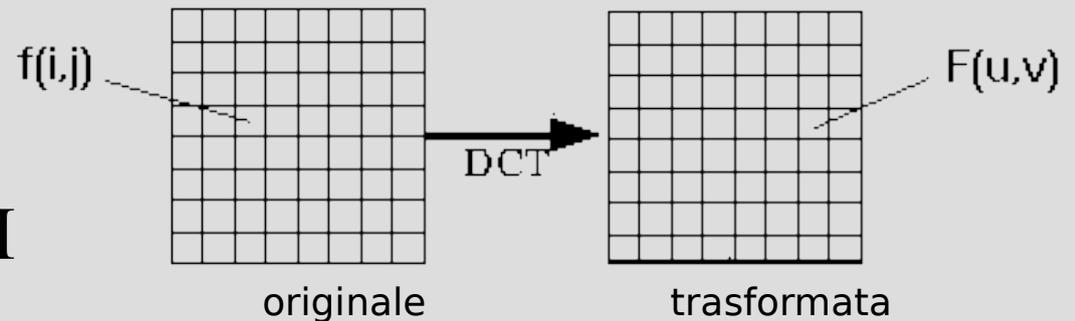
trasformata DCT

- Ciascun blocco viene **trasformato** usando la **DCT** (trasformata discreta del coseno)
 - è una variante della Trasformata di Fourier
 - trasforma l'immagine dallo spazio delle *ampiezze* a quello delle *frequenze*
 - le **frequenze alte** corrispondono a **dettagli fini**, che l'occhio non riuscirebbe comunque a scorgere – si possono tagliare!

Algoritmo JPEG

trasformata DCT

- Il valore della trasformata DCT di un'immagine di NxM pixel è data dall'equazione:



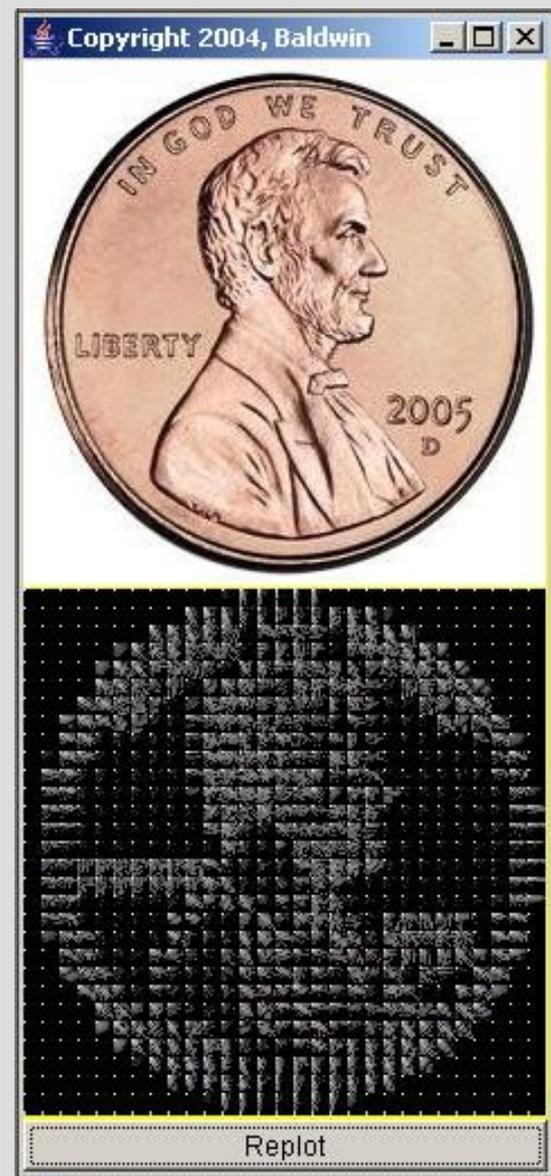
$$F(u, v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i) \cdot \Lambda(j) \cdot \cos \left[\frac{\pi \cdot u}{2 \cdot N} (2i + 1) \right] \cos \left[\frac{\pi \cdot v}{2 \cdot M} (2j + 1) \right] \cdot f(i, j)$$

$$\text{dove } \Lambda(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

- La funzione inversa trasforma $F(u,v)$ in $f(i,j)$ e restituisce l'immagine originale

Algoritmo JPEG

trasformata DCT

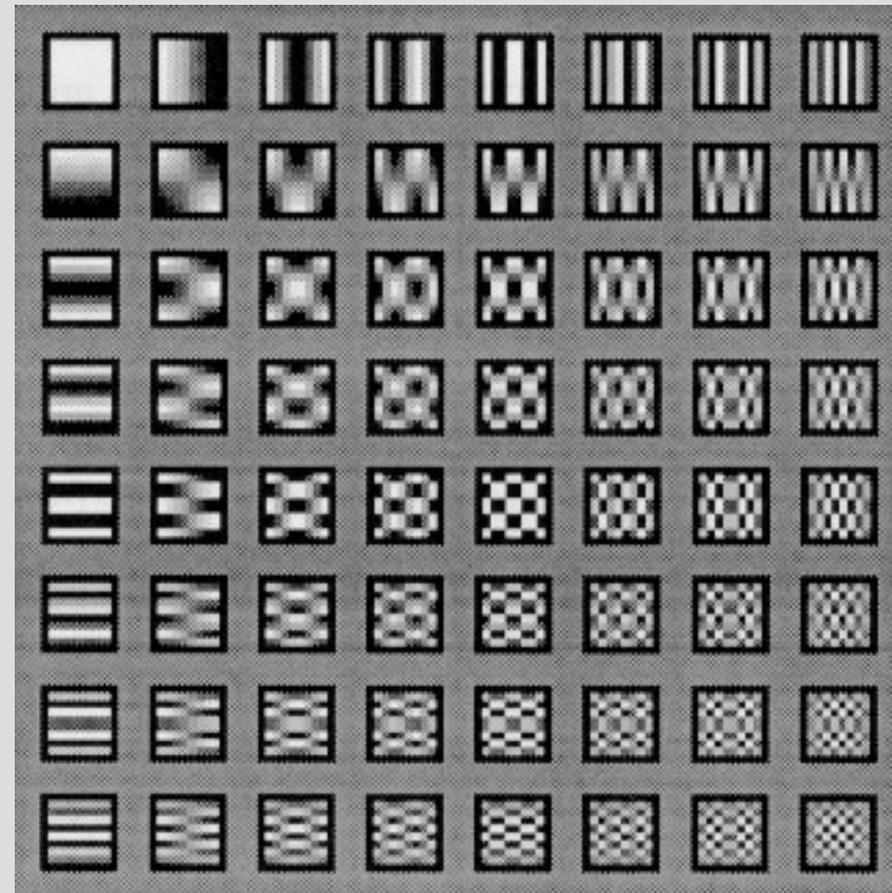


- L'applicazione della DCT ai blocchi 8x8 trasforma l'immagine
- Ogni blocchetto trasformato rappresenta uno *spettro di frequenza*
- L'energia è maggiore (= c'è più bianco) dove più alto è il contrasto

Algoritmo JPEG

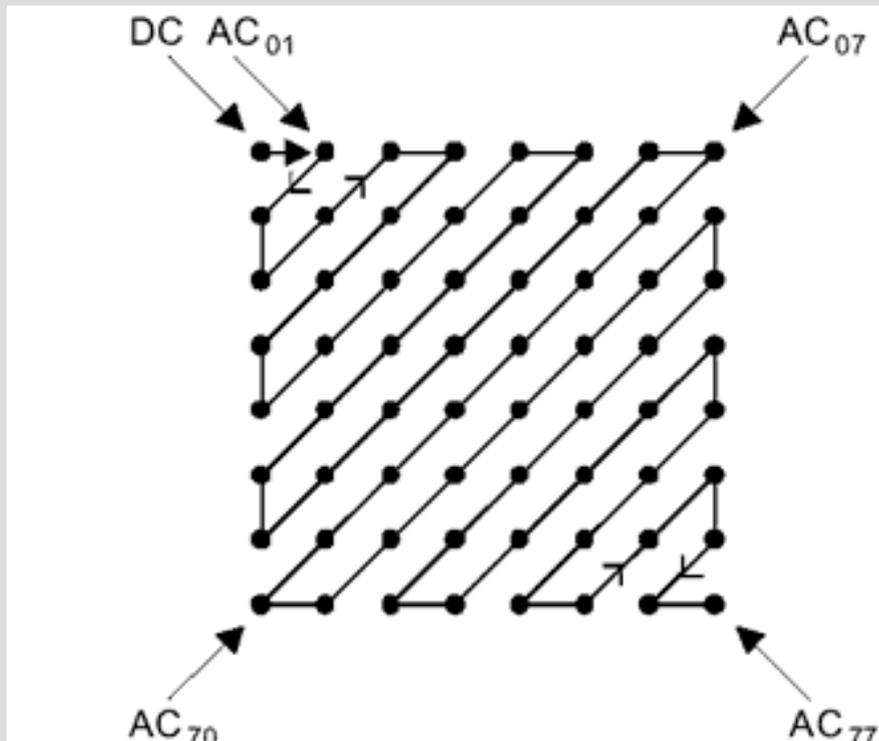
trasformata DCT

- I coefficienti che esprimono lo spettro di ciascun blocchetto vengono **approssimati**
- Questo equivale a rappresentare un blocchetto 8x8 come **combinazione** di un piccolo numero di blocchetti predefiniti



Algoritmo JPEG

riordinamento dei punti



- I singoli punti di un blocchetto vengono visitati a zig-zag
- Quando li si mette in sequenza in quest'ordine, è più facile che si trovino punti consecutivi simili
- La compressione funziona meglio!

Algoritmo JPEG

compressione lossless finale

- I passi visti fin qui **scartano** informazione:
 - si riduce la risoluzione della crominanza
 - si approssimano i colori
 - si eliminano i dettagli troppo fini per essere visti dall'occhio
- Sui dati rimanenti, si applica infine una compressione lossless (Huffman)
- Il risultato finale è la codifica JPEG dell'immagine originale
 - fattore di compressione: da 10 a 50 volte!

Algoritmo JPEG

altri usi notevoli

- L'algoritmo JPEG è usato (ovviamente) nei file .jpg o .jpeg – che costituiscono l'80%-90% delle immagini sul Web
- È un formato anche molto usato in ambito medico (raggi X, TAC, ecc.)
- Una variante più sofisticata è usata nei formati per i film (MPEG, AVI, WMV; anche DVD, satellite e digitale terrestre)
- Grande vantaggio: **qualità buona, compressione alta**

Algoritmo JPEG

JPEG2000 e il futuro

- JPEG2000 è (sarà) la nuova versione dello standard JPEG
- Lavoro iniziato nel 2001, tuttora in corso
- Usa la compressione **wavelet**
 - 20% di compressione in più
 - qualità significativamente più alta
 - la definizione matematica è *molto* complicata
- Pochissimo supporto nelle applicazioni

Algoritmo JPEG

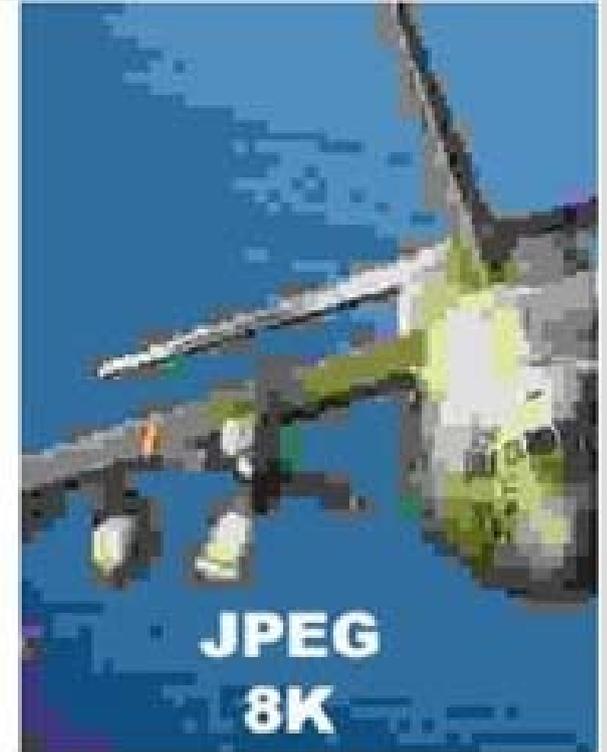
JPEG2000 e il futuro



Original
e



JPEG2000
0
compressione
wavelet



JPEG
compressione
DCT

- Notate comunque il rapporto **1:150!**