Algoritmi di compressione per le immagini

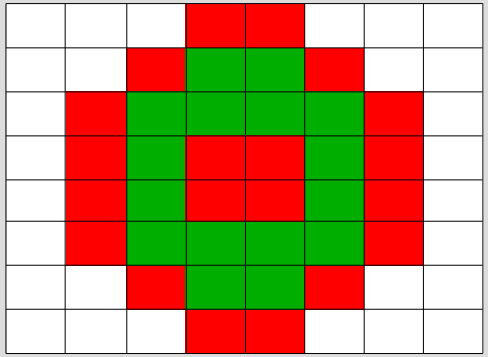
- Abbiamo visto nella lezione precedente che le immagini possono essere **compresse** in vari modi, per ridurne l'occupazione di memoria (su disco o rete)
  - compressione lossless – non si perdono dati
  - compressione lossy – si perdono i dati “meno importanti”

Algoritmo RLE

- L'**RLE** (*run lenght encoding*, codifica della lunghezza delle sequenza) 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:



=

B	B	B	R	R	B	B	B
B	B	R	V	V	R	B	B
B	R	V	V	V	V	R	B
B	R	V	R	R	V	R	B
B	R	V	R	R	V	R	B
B	R	V	V	V	V	R	B
B	B	R	V	V	R	B	B
B	B	B	R	R	B	B	B

RLE

3B	2R	3B				
2B	1R	2V	1R	2B		
1B	1R	4V	1R	1B		
1B	1R	1V	2R	1V	1R	1B
1B	1R	1V	2R	1V	1R	1B
1B	1R	4V	1R	1B		
2B	1R	2V	1R	2B		
3B	2R	3B				

8x8x24 = 1536 bit

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 ...
  - 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 Huffmann 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

---

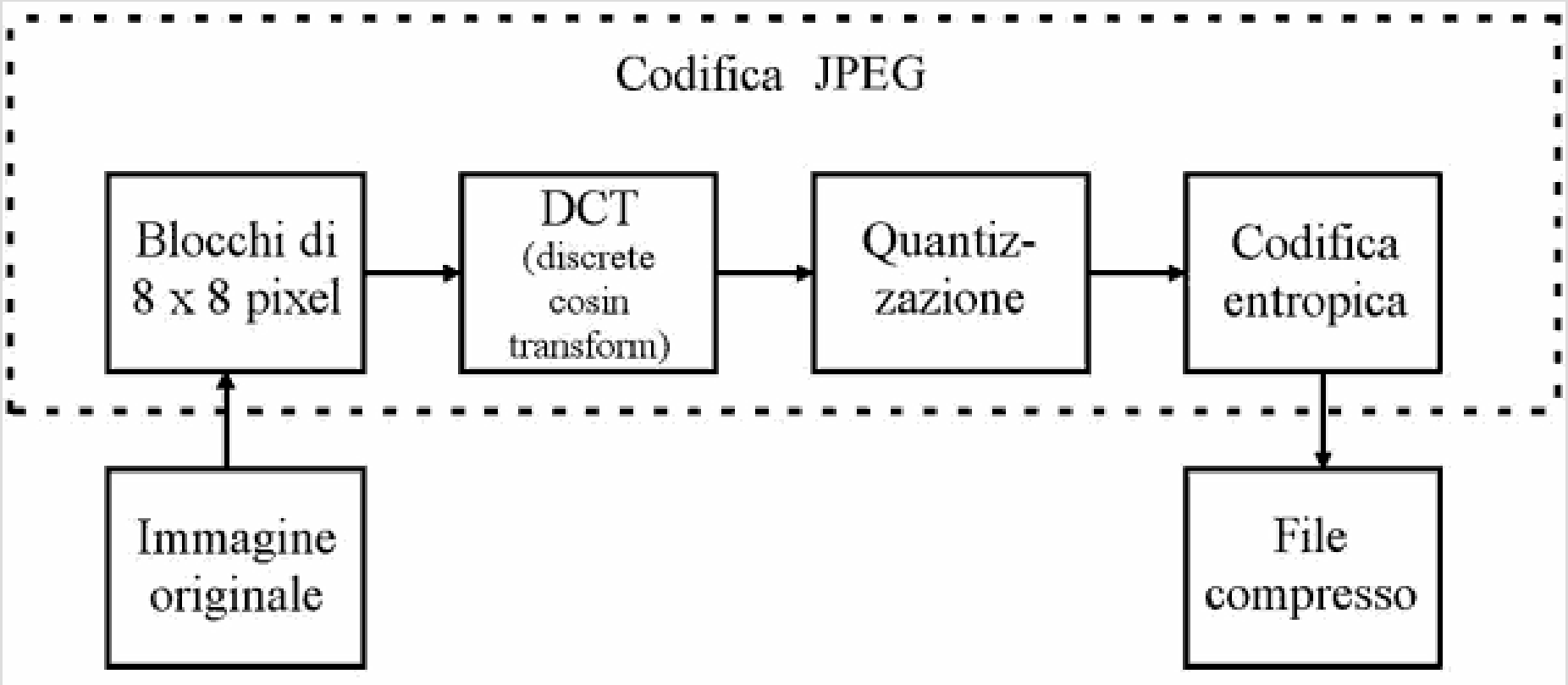
* ma si ricordi il teorema precedente...

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

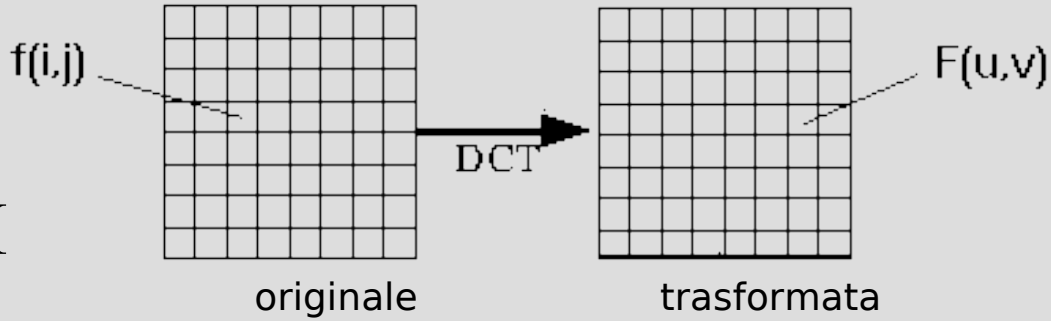
- Il passo successivo consiste nel dividere l'immagine in blocchi di 8x8 pixel
- Un'immagine di tipo televisivo (640x512 pixel) richiede 80x64 blocchi
- A volte la “scalettatura” causata dai blocchi è ben visibile!
  - quasi sempre dovuto ad alti fattori di compressione

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:

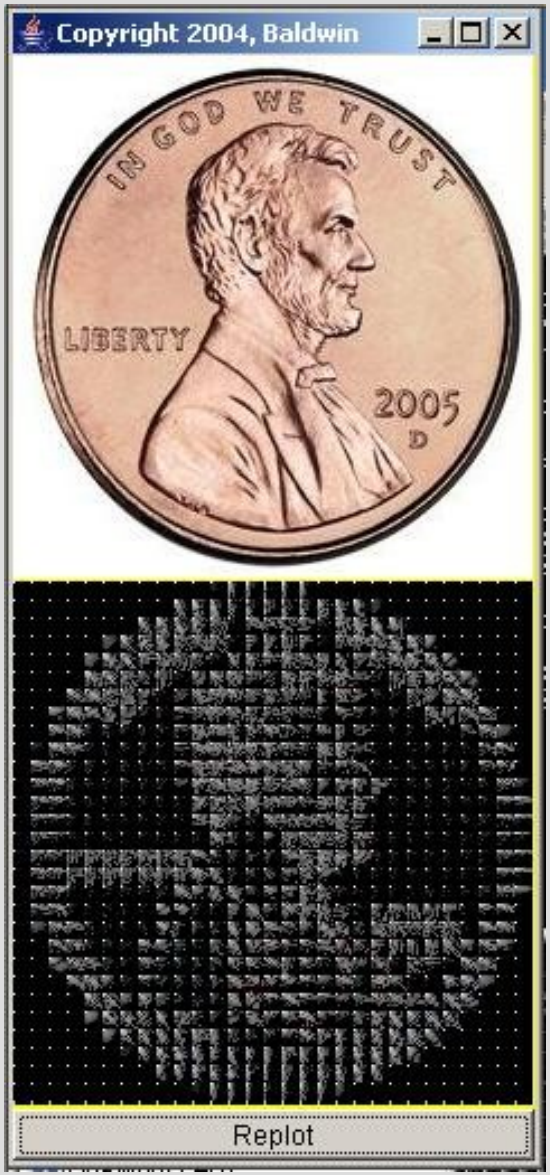


$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right].f(i,j)$$

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

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

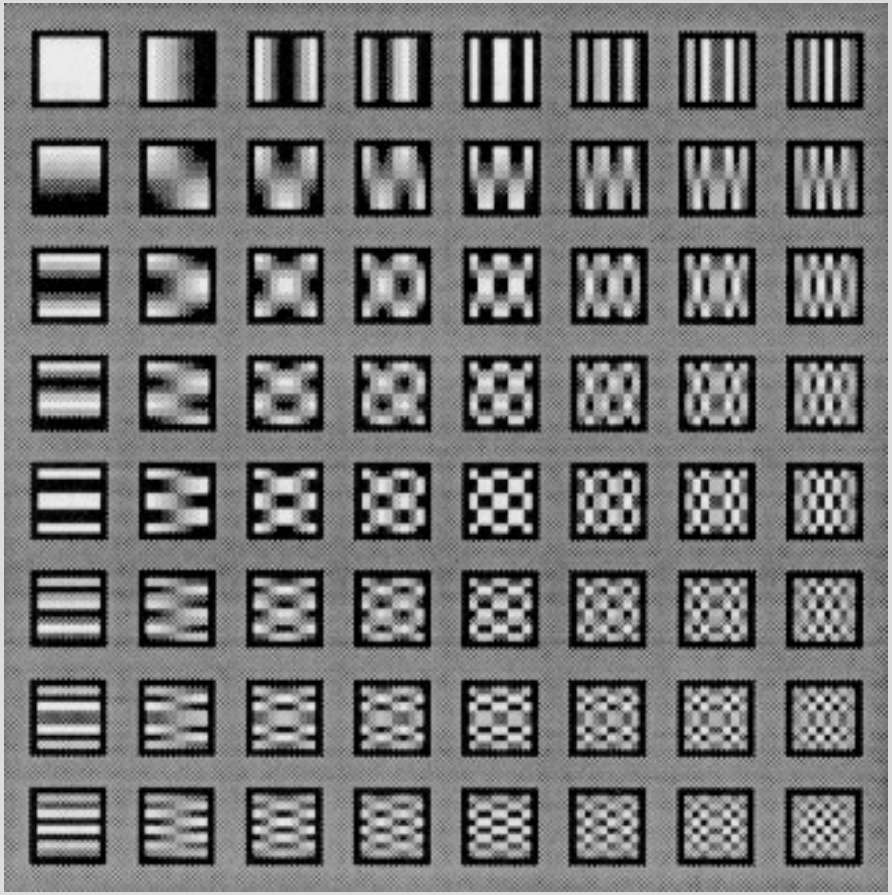
Algoritmo JPEG
trasformata DCT



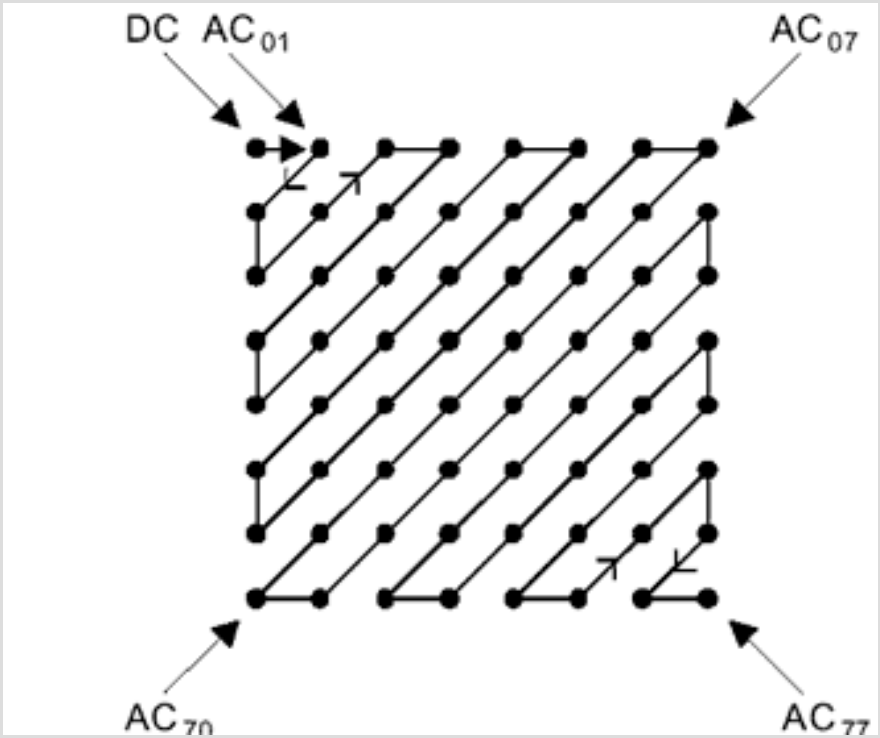
- L'applicazione della DCT ai blocchi 8x8 trasforma l'immagine
- Ogni blocchetto trasformato rappresenta uno *spettro di frequenza*
- L'energia è maggiore (= c'è più bianco) dove più alto è il contrasto

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

JPEG200
0
compressione
wavelet

JPEG
compressione
DCT

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