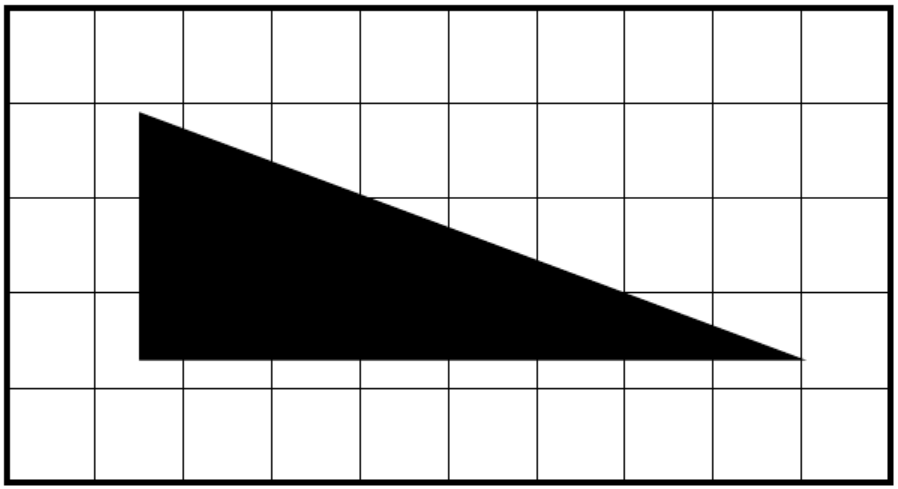
Data coding
Compression
Image formats

Andrés Méndez

April 2014

La codifica delle immagini

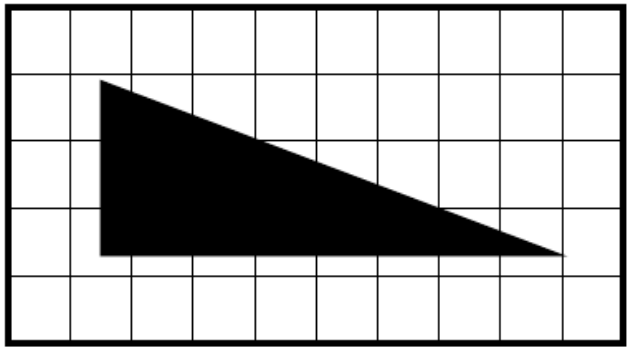
- Le immagini possono essere memorizzate in forma numerica (digitale) **suddividendole in milioni di punti**, per ognuno dei quali si definisce il colore in termini numerici.



- Ogni quadratino di questa griglia prende il nome di pixel (picture element). Ad ogni pixel può essere assegnato un valore binario ad es. 0 se nel quadratino prevale il bianco e 1 se nel quadratino prevale il nero.

La codifica delle immagini

- Partiamo a contare i quadratini dal più in basso a sx. Posso assegnare a questa figura geometrica la seguente serie di bit:

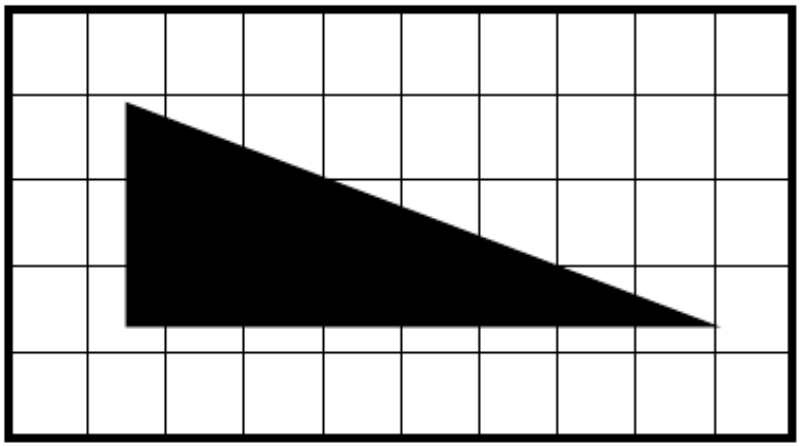


0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

- 0000000000 0111111110 0111110000 0110000000 0000000000
- Che posso memorizzare in un file: archivio, fila
- I file sono caratterizzati da un nome e una estensione nome.ext

La codifica delle immagini

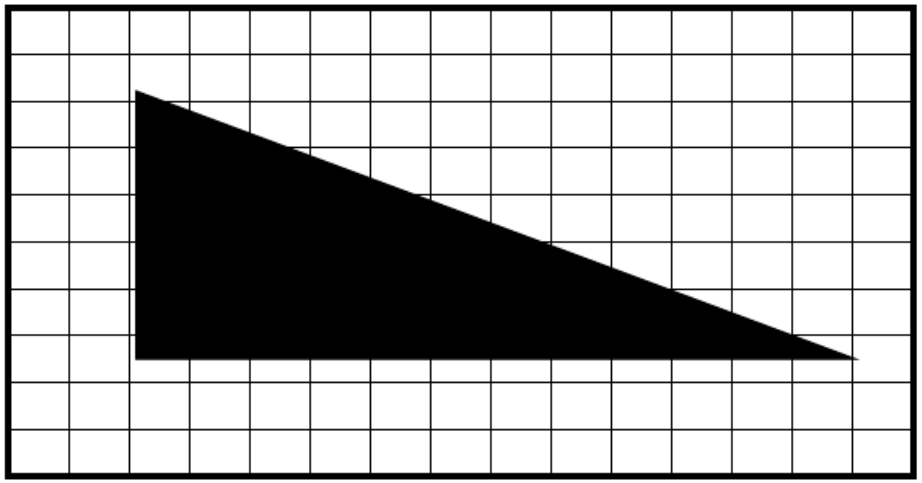
0000000000 0111111110 011110000 0110000000 0000000000



0	0	0	0	0	0	0	0	0	0
0	■	■	0	0	0	0	0	0	0
0	■	■	■	■	■	0	0	0	0
0	■	■	■	■	■	■	■	■	0
0	0	0	0	0	0	0	0	0	0

La codifica delle immagini

- Aumentando il numero dei quadratini, (pixel) in cui scompongo l'immagine la digitalizzazione sarebbe più precisa



Codifica dei dati

- In generale servono $\log_2 N$ bit (0/1) per distinguere un oggetto tra N altri.
- Esempi:
  - Servono 5 bit per individuare un carattere alfabetico tra 32 (o meno)
  - Servono 8 bit per individuare un simbolo alfanumerico (ASCII) tra 256
  - Servono 8 bit per individuare un tono di grigio tra 256
  - Servono 24 bit per individuare uno tra 2^{24} colori
- L'associazione tra l'oggetto e la sequenza di bit prende il nome di codifica

Rappresentazione di testi

- Ciascun carattere di un testo viene codificato con 8 bit tramite il codice ASCII
- Il codice ASCII è una tabella che elenca per ciascuno dei 256 caratteri alfanumerici la corrispondente sequenze di 8 bit
- La lettura sequenziale del testo nell'ordine consueto produce una sequenza di bit

Rappresentazione di suoni

- Un suono è una onda di pressione che può essere trasformata in un segnale elettrico continuo, che varia in funzione del tempo.
- Per rappresentare il suono nel computer sono necessarie due operazioni:
  - campionamento: viene registrato il valore del segnale sonoro ad intervalli di tempo fissati (es. 8000 Hz = ogni 0,125 ms)
  - quantizzazione: il valore del segnale sonoro viene rappresentato con un numero fisso di bit
- Al termine si ottiene una sequenza di bit.

Rappresentazione di immagini

- Una immagine digitale è una *matrice* (tabella a due dimensioni) di punti (detti pixel), ciascuno dei quali contiene la codifica di un colore.
- A seconda di quanti colori si vogliono poter distinguere serviranno più bit per pixel
- La matrice può essere stesa in una sequenza leggendola, per esempio, per righe

Rappresentazioni

- In ultima analisi tutti i dati multimediali si riducono ad una sequenza di bit (codifica)
- Un documento multimediale può richiedere una grossa mole di bit per essere codificato
- È necessario comprimere i dati, ovvero introdurre codifiche "parsimoniose" che richiedano (idealmente) il minimo numero di bit necessari a codificare l'informazione presente nei dati.

Enciclopedia multimediale (1)

- 500.000 pagine di testo (2 KB per pagine) – totale 1 GB
- 3.000 immagini a colori (640x480x24 bit = 1MB) – totale 3 GB
- 500 mappe (640x480x16 bit = 1MB) – totale 3 GB
- 60 minuti di suono stereo (176 KB/s) – totale 0.6 GB
- 60 minuti di animazioni (640x480x16 bit x 16 frame/s = 6.5 MB/s) – totale 23.4 GB
- 50 minuti di video (640x480x24 bit x 30 frame/s = 27.6 MB/s) – totale 82.8 GB

La codifica di un'enciclopedia richiederebbe 111,1 GB

  - 1 CD contiene 700 MB
  - una rete veloce trasmette a 100 Mb/s

Enciclopedia multimediale (2)

- Applicando opportune tecniche di compressione ai diversi tipi di media usati nell'enciclopedia possiamo ottenere mediamente i seguenti rapporti di compressione:
  - Testo 2:1
  - Immagini a colori 15:1
  - Mappe 10:1
  - Suono stereo 6:1
  - Animazioni 50:1
  - Video 50:1
- Dopo la compressione, l'enciclopedia richiede 2.96 GB

Tipi di compressione

- **Lossless** (senza perdita): permette al ricevente di ricostruire perfettamente il dato originale (reversibile).

  La compressione deriva dalla eliminazione della ridondanza, ma l'informazione è preservata integralmente.

  Es.: testi, immagini per uso forense o medico.

  Si applica ad una sequenza di bit, non dipende dal dato che è stato codificato.

- **Lossy** (con perdita): permette di ricostruire solo una approssimazione del dato originale (irreversibile).

  Viene eliminata l'informazione che si ritiene non pregiudichi il significato (la percezione) del dato.

  Es.: suoni, immagini, video.

  Tecniche specifiche per i dati a cui si riferiscono (essendo basate sulla percezione)

Tipi di compressione

- Lossless
  - Senza perdita di informazione
  - Si riduce la ridondanza
  - Si comprime mediamente fino al 50% delle dimensioni originali
  - Ottimale quando non si possono tollerare modifiche nei contenuti (ad es. testi, multimedialità professionale,etc.)
- Lossy
  - Con perdita di informazione
  - Si riduce l'irrilevanza (presunta)
  - Si comprime mediamente fino al 10% delle dimensioni originali
  - Ottimale quando si possono tollerare piccoli errori o modifiche (immagini e audio non professionali)

Tipi di compressione

- Gli schemi di compressione si possono dividere i due famiglie fondamentali:
  - Codifica Entropica: La codifica entropica manipola i flussi di bit senza preoccuparsi del loro significato. E' applicabile a qualsiasi tipo di dato. Appartengono a questa categoria le codifica Run Length, Huffman, le codifiche statistiche.
  - Codifica Sorgente: Questo tipo di codifica sfrutta le proprietà dei dati per comprimerli di più, di solito in modo infedele (*lossy*).

Tecniche di compressione senza perdita

- Le tecniche *lossless* possono essere suddivise in due categorie principali:
  - Basate su dizionario: generano un file compresso con codici a lunghezza fissa (e.g. 12-16 bits) che rappresentano una sequenza di valori nel file originale.
  - Metodi statistici: i dati più frequenti sono rappresentati con meno bit rispetto a quelli solitamente richiesti. Anche Samuel F. B. Morse ha utilizzato un sistema di questo tipo nello sviluppo del codice morse. La frequente lettera *e* è rappresentata da un singolo punto. La meno frequente *z* con - - . .

Codifiche *lossless* (entropiche)

- Codifica di Huffman
  - E' possibile risparmiare sulla lunghezza del messaggio impiegando una codifica a lunghezza variabile
  - I simboli più frequenti nel testo vengono codificati con meno bit (es. Morse)
  - In media i messaggi risulteranno più corti.

- Run Lenght Encoding (RLE)
  - Ogni sequenza di simboli uguali (*run*) viene sostituita dal simbolo stesso seguito dal numero di occorrenze

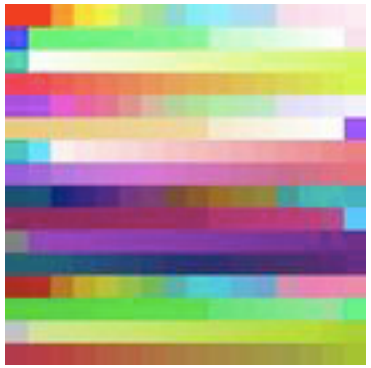
Esempio di codifica RLE

- Messaggio: abbccccccddddddddd
- Codifica: abb&c6&d8

- Note
  - Serve un carattere speciale “&” per segnalare l'inizio del *run*
  - Non è conveniente codificare *run* più corti di 4.

Esempio

- Caso limite di un'immagine creata artificialmente, 16 x 16 pixel, costituita da 256 colori unici differenti (pixel tutti differenti l'uno dall'altro nei valori cromatici!). In alcuni casi, l'uso della compressione RLE si dimostra addirittura controproducente.



Tecniche di compressione senza perdita

- Codifica di Huffman
  - Sviluppata nel 1950. E' una tecnica di compressione con perdita che determina un codice a lunghezza variabile con ridondanza minima.
  - Utilizza un albero di codifica binario: per rappresentare le occorrenze più comuni utilizza pochi bit, quelle più rare sono codificate con più bit.
  - La versione statica di Huffman utilizza un albero precedentemente costruito attraverso una tabella di probabilità di occorrenze dei possibili valore dei dati.
  - La versione dinamica invece costruisce tale tabella *on the fly,* durante la compressione.

Esempio di codifica di Huffman

- Esempio: alfabeto di 4 simboli {a,b,c,d} che appaiono nel testo con frequenza a: 70%, b: 10%, c: 8%, d: 12%
  - Codifica a lunghezza fissa: 2 bit per simbolo (ignorando le differenti frequenze)
  - Codifica a lunghezza variabile (Huffman)
    - a → 0
    - d → 10
    - b → 110
    - c → 111
  - Nota: Come distinguere le diverse lettere in una sequenza di 1 e 0? Nessuna sequenza appare come prefisso in una più lunga.

Tecniche basate su dizionario

- Run Length Encoding (RLE): Le img soprattutto quelle a ldg contengono ampie regioni di pixel dello stesso ldg. Un insieme di pixel con lo stesso ldg è detto *run*. Si può memorizzare un codice che specifica il ldg seguito dalla lunghezza del run. Nel caso peggiore si raddoppia la lunghezza del file.
- Codifica LZW: Come nel caso RLE codifica stringhe di caratteri. All'inizio del processo di codifica viene costruito un codebook che contiene i simboli sorgente da codificare. Per img 8-bit a ldg, si assegnano le prime 256 parole del dizionario ai ldg 0..255. Durante il processo di codifica le seq di ldg non nel dizionario sono opportunamente memorizzate nel dizionario.
  - MAMA&MA&MA&M

Esempio

- Se due pixel dell'img sono bianchi allora assegno la sequenze 255-255 alla locazione 256. La prossima volta che si incontra una seq 255-255 si assegna il valore del codeword 256. Chiaramente risulta di fondamentale importante la lunghezza scelta del dizionario.
- La codifica LZW è utilizzata nei formati: GIF, TIFF, PDF.

Codifica sorgente

- Codifica differenziale: la seq di valori (e.g. video) viene codificata rappresentando ciascun valore per differenza rispetto al precedente, ipotizzando che grossi salti tra due dati siano improbabili. (es., formato video MPEG).

Codifica sorgente

- Il secondo esempio di codifica sorgente sono le *trasformazioni*. La compressione può diventare molto semplice trasformando i segnali da un dominio all'altro.
- Passando alla TdF la trasmissione di ampiezze può richiedere meno bit che trasmettere tutta la forma d'onda.
- Molto usata è la DCT (Discrete Cosine Transform).
  - Ha la proprietà che nel caso di immagini con forti discontinuità la maggior parte del potere di spettro risiede nei primi termini, il che permette di ignorare gli altri senza gravi perdite di informazione (e.g. standard JPEG)

Codifiche lossy: JPEG

- L'immagine viene suddivisa in blocchetti di 8x8 pixels, per ciascun blocchetto solo le *componenti più importanti* vengono preservate (quelle che si vedono di più).

- L'informazione cromatica viene trattata diversamente da quella di intensità, sfruttando il fatto che il sistema visivo umano è molto più sensibile a variazioni di luminosità piuttosto che di colore.

Esempio

- Originale (532 KB)

Esempio

- JPEG (9 KB)

Formati audio/video

- MP3 (MPEG layer 3):
  - è semplicemente il nome della parte di codifica audio contenuta in MPEG
- AAC: Audio Advanced Coding
- MPEG (Moving Pictures Experts Group)
  - Formato lossy per audio e video digitale. Raggiunge rapporti di compressione di 200:1 per il video e 10:1 per l'audio
- Quicktime
  - Formato sviluppato da Apple, permette codifica ed integrazione di documenti multimediali complessi, comprendenti audio, video, dati 3D, etc…

Formati di immagini

- BMP, *lossless*
- TIFF, *lossless/lossy*
- GIF (Graphics Interchange Format)
  - Formato *lossless* per immagini a 256 colori, utile per icone e disegni (*cartoon*)
- JPEG (Joint Photographic Expert Group)
  - Formato lossy per immagini a 16 ML di colori (*True Color*), adatto per fotografie, opere artistiche etc.
- PNG (Portable Network Graphics)
  - Intende sostituire GIF, il cui algoritmo di compressione è coperto da copyright. Gestisce immagini *True Color*.
- In genere i vari formati, molti proprietari, si differenziano dall'intestazione (*header*) che contiene info aggiuntiva, e dalla modalità di memorizzazione dei dati (compressione).

Formati di immagini

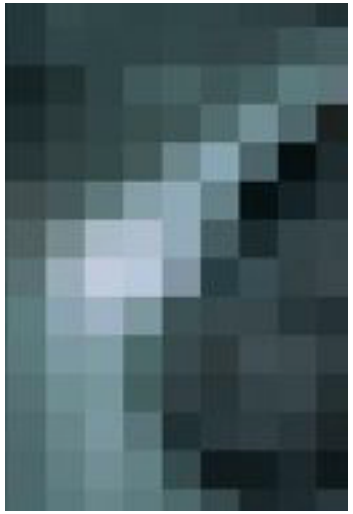
- Esistono 44 diversi formati di immagini
- Un'immagine è formata da un insieme ordinato (matrice) di punti (pixel).
- Ogni punto è caratterizzato da un “colore”: livello di grigio o colore (vero, o falso LUT).
- La risoluzione del colore si misura in bit, bit per pixel (bpp)
- La dimensione dell'immagine è NxMxbpp/8 [byte]

Vettoriale vs. Raster

- I formati di immagini si dividono in due grandi famiglie: formati vettoriali e formati raster
- Formati vettoriali: l'immagine viene descritta attraverso formule matematiche, utilizzando curve e poligoni geometrici (eg., xfig) la qualità viene mantenuta anche ingrandendo infinite volte l'immagine
- Formati raster (o bitmap): l'immagine è costituita da un insieme di pixels generalmente più grosse di quelle vettoriali, se ingrandite non mantengono la qualità originale ma diventano sempre più sgranate;
- Esiste anche il formato META che può contenere sia dati in forma raster e vettoriale.

Zoom nei due tipi di formato

Raster



Vettoriale

Il formato Bitmap, BMP

- È stato definito dalla Microsoft
- È un formato di tipo *raster*
- Ad ogni pixel si associa l'indice di una colormap (vettore di valori RGB). La profondità del colore può essere di 16 colori (4 bit), 256 colori (8 bit), 2^24 colori (24 bit). Supporta anche immagini binarie (1 bit) o a livelli di grigio (8 bit).

  Esempio: icone, contatori nei siti Internet, dove i numeri vengono pre-composti con matrici 0/1 e richiamati da script.
- Formato oneroso in termini di memoria, non tiene conto della ridondanza dei dati.

Il formato Bitmap, BMP

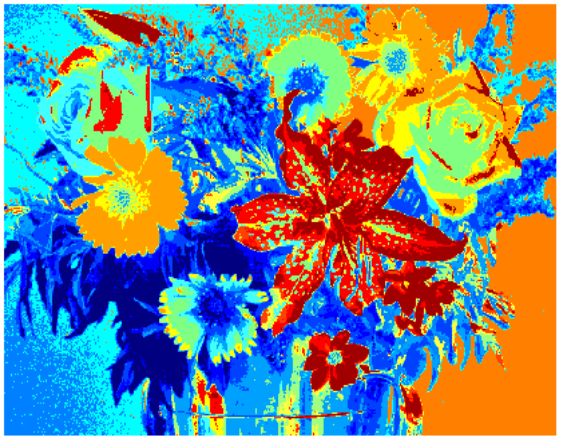
- Il formato BMP prevede anche una semplice compressione: Run Length Encoding, RLE
- Tale compressione tiene conto della struttura raster dell'img come righe di 0/1 e, per ogni riga, codifica (ad esempio) le stringhe di 1:
  - 1110011111000101111
  - codificata come (1,3) (6,4) (13,1) (15,4)
- Esistono diverse variazioni sul tema (eg., RLE con codice Huffman)
- Le versioni compresse RLE permettono una profondità massima di solo 8 bit per pixel.

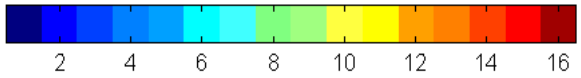
Esempi di immagini BMP

Immagine a 8 bit x colore

Immagine a 4 bit

Immagine a 4 bit con diversa colormap





Il formato Bitmap, BMP

- Ogni file bitmap contiene:
  - BITMAPFILEHEADER Questo è l'header della bitmap che contiene informazioni sulla grandezza in byte del file e l'offset dall'inizio del file del primo byte nella mappa dei pixel.
  - BITMAPINFOHEADER Qui sono indicate le dimensioni in pixel dell'immagine e il numero di colori utilizzati. Sempre in questa struttura sono indicate inoltre la risoluzione orizzontale e verticale del dispositivo di output: questi valori, uniti a quelli della larghezza e dell'altezza in pixel,determinano le dimensioni di stampa dell'immagine in grandezza reale.
    - modello di colore
    - tavolozza: questa struttura è un array che fa corrispondere un colore ad ogni indice che può essere assegnato ad un pixel.
    - mappa dei pixel: questa struttura di dati costituisce il corpo vero e proprio della bitmap, dove ad ogni pixel si fa corrispondere un colore sotto forma di indice nella tavolozza, oppure nelle sue componenti cromatiche.

Il formato Bitmap, BMP

- Il file ha la seguente forma:
  - BITMAPFILEHEADER bmfh;
  - BITMAPINFOHEADER bmih;
  - RGBQUAD aColors[];
  - BYTE aBitmapBits[];
- L'header è definito come una struttura BITMAPFILEHEADER.
- Il bitmap-file header contiene informazioni sul tipo, dimensione e struttura del file DIB.
- Il *bitmap-information header* è definito come una struttura BITMAPINFOHEADER, e specifica la dimensione, il tipo di compressione, e il formato di colore della bitmap.

Il formato Bitmap, BMP

- La tabella di colore, definita come un array di strutture RGBQUAD, contiene tanti elementi quanti i colori della bitmap.
- La tabella non è presente per file bitmap con 24 bit per colore perché ogni pixel è rappresentato 24-bit (valori RGB) nell'area dati.
- I colori appaiono nella tabella in ordine di importanza in modo che il dispositivo di visualizzazione possa gestire anche situazioni in cui tutti i colori non possono essere resi.
- La struttura BITMAPINFO può essere usata per rappresentare una combinazione dell'intestazione *bitmap-information header* e tabella di colore.

Il formato Bitmap, BMP

- L'area dati, la bitmap che segue immediatamente la tabella di colore, consiste di un array di BYTE che rappresentano linee consecutive o "*scan line*" della bitmap.
- Ogni *scan line* consiste di byte consecutivi che rappresentano i pixel nella linea ordinati da sx a dx, a partire dal basso verso l'alto.
- Il numero di byte di una *scan line* dipende dal colore, formato e larghezza in pixel della bitmap.
- Se necessario, una *scan line* deve essere *zero-padded* per finire con un limite di 32-bit (tuttavia, i limiti dei segmenti possono essere ovunque nella bitmap).

Il formato Bitmap, BMP

- Un campo della struttura BITMAPINFOHEADER determina il # di bit di ogni pixel e il max # di colori:
  - 1: la bitmap è monocromatica e la tabella di colore contiene 2 campi. Ogni bit della bitmap rappresenta un pixel che viene reso con il colore del 1° o 2° campo.
  - 4: la bitmap ha 16 colori al max. Ogni pixel della bitmap è rappresentanto da un indice a 4-bit nella tabella di colore.
    - Es.: se il 1° byte della bitmap è 0x1F, il byte rappresenta 2 pixels, il primo contiene il colore del 2° campo della tabella, e il secondo pixel contiene il colore del 16° campo della tabella.
  - 8: la bitmap ha 256 colori al max. Ogni pixel della bitmap è rappresentato da un indice di 1-byte nella tabella.
    - Es.: se il primo byte è 0x1F, il primo pixel ha il colore del 32° campo della tabella.
  - 24: la bitmap ha un massimo di 2^{24} colori. Ogni sequenza di 3-byte nell'array bitmap rappresenta l'intensità relativa di rosso, verde e blu del pixel.

Il formato GIF

- GIF è un acronimo per *Graphic Interchange Format*, e fu standardizzato nel 1987 da CompuServe, sebbene il brevetto della formula originale appartiene a Unisys.
- Il primo formato GIF permetteva img con 8 bit-per-pixel, ie., 256 colori: GIF87a
- CompuServe aggiornò il formato nel 1989 per prevedere animazioni, trasparenza, e interalacciamento: GIF89a

GIF87a

GIF89a

Il formato GIF

- Il formato GIF non può essere usato per rappresentare img con formato colori particolarmente elevati poiché utilizza solo 8 bit per la codifica del colore.
- Il fatto che vi siano solo 8 bits/pixel non significa che non sia supportato il colore ma soltanto che non possono essere più di 256. Questo si implementa tramite una *lookup table* (LUT) dove i 256 colori sono memorizzati in una tabella e 1 byte viene usato come indice per identificare il colore del pixel.
- Ammette compressione senza perdita (LZW, Lempel-Ziv-Welch)

Animazione nel formato GIF

- Le animazioni sono simili a quelle dei *cartoons* di qualche anno fa.
- L'esempio è formato da un certo numero di img in sequenza.
- Esiste la possibilità di fissare il movimento del filmato fino a 1/100 secondo per ogni “animation frame”.
- Le animazioni si sono evolute nel tempo permettendo compressioni sempre più spinte, legate al fatto che un'img non differisce molto dalla successiva.

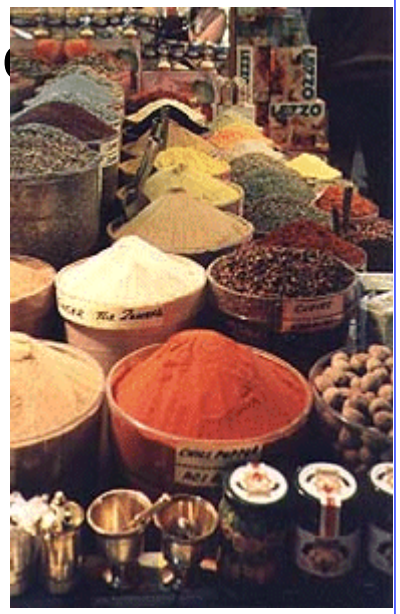
Trasparenza nel formato GIF

- La trasparenza è limitata ad un colore solo (ie., chroma-key) che viene praticamente rimosso dalla tabella (non visualizzato)

Interlacciamento nel formato GIF

- L'img non interallacciata viene visualizzata dall'alto verso il basso.
- Nel caso di visualizzazione remota (WWW) il processo può essere lento → *interlaced GIF89a*
- Nel formato GIF interallacciato le scan line s visualizzate in modo alternato.

  L'effetto risultante è che l'img appare tutta in tempi più brevi, ma appare subito sfuocata fino a raggiungere il dettaglio massimo



Il formato TIFF

- TIFF - Tag Image File Format, mantenuto dall'Adobe;
- Può supportare immagini in vari spazi di colori:
- Scala di grigi;
- Pseudocolori (di ogni dimensione)
- RGB;
- YCbCr;
- CMYK
- CIELab;
- È un sistema portabile, non predilige un particolare sistema operativo.

Il formato TIFF

- Permette l'utilizzo di vari schemi di compressione:
  - non compresso;
  - PackBits;
  - Lempel-Ziv-Welch (come Gif)
  - Jpeg;
- Progettato per essere estensibile.
- Permette l'inclusione di una quantità illimitata di informazione (di tipo privato o meno).

Il formato TIFF

- Il formato di immagine TIFF considera un file come una sequenza di byte di 8 bit numerati da 1 a N.
- I campi di un file TIFF sono univocamente identificati così da poter essere presenti o assenti come richiesto dall'applicazione.
- La dimensione massima è 2^{32} byte.
- Un file TIFF è composto da:
  - *image file header* (IFH) di 8 byte, che punta ad
  - una o più *image file directory* (IFD)
- IFD contiene info sull'img e puntatori all'area dati img vera e propria.

Il formato TIFF

- IFH è così composto:
  - Byte 0-1 - La prima parola specifica l'ordine dei byte usato nel file. Valori legali sono:
    - II (hex 4949), dal byte meno significativo al più significativo (*little-endian*);
    - MM (hex 4D4D), viceversa (*big-endian*). Entrambi validi per interi a 16 e 32 bit.
  - Byte 2-3 - La seconda parola del file è il numero della versione.
    - In realtà, non esiste un numero di formato; in questo campo c'è il numero 42 (2A in hex) che non è mai stato cambiato e mai lo sarà per ragioni storiche.
    - Può servire anche come conferma del formato TIFF. I campi hanno un significato ben definito e permanente, così che SW precedente può leggere anche versioni più recenti.

Il formato TIFF

  - Byte 4-7 - Contiene l'*offset* (# byte rispetto all'inizio del file) della prima IFD. La directory può essere in ogni posizione del file dopo l'header, ma deve cominciare all'inizio della parola.
  - In particolare, un IFD può seguire i dati img che descrive.
- Una IFD è composta da:
  - un contatore di 2 byte contenente il numero di campi presenti
  - un campo composto da una sequenza di 12-byte
  - un campo di 4-byte contenente l'offset della prossima IFD (o 0 se assente).
- Ogni campo di 12-byte è così composto:
  - Byte 0-1: Contiene il TAG (identificatore) del campo: ad es., tipo di img, a colori, ldg, etc.
  - Byte 2-3: Contiene il campo Tipo (*Type*): ad es., SHORT

Il formato TIFF

- Byte 4-7: Contiene il numero di valori, *Count*, del Tipo indicato.
- Byte 8-11: Contiene il valore di offset del campo *Value,* o il valore stesso se meno di 4 byte. Ad es., tipo di compressione, unità di misura di riferimento, etc.

Esempi di formato TIFF

Tiff a colori RGB

Tiff a colori con diversa palette

Esempi di formato TIFF

Tiff a 256 ldg (8 bit)

Tiff B/W (1 bit)

Il formato JPEG

- Algoritmo di compressione sviluppato dal Joint Photographic Experts Group
- E' importante perché lo std multimediale per le immagini in movimento MPEG è grosso modo la codifica JPEG di ciascuna img separatamente, più alcune funzioni extra per la compressione di serie di img e rilevamento del movimento.
- Supporta il True-color (16 milioni di colori)
- La compressione è di tipo lossy (con perdita): dall'immagine compressa non è possibile ricostruire l'immagine originale
- La qualità dell'immagine dipende dal grado di compressione: più si comprime peggiore è la qualità

Rapporto compressione - qualità

JPEG Image 1%
Compression 44.6KB

JPEG Image 50%
Compression 14.2KB

JPEG Image 99%
Compression 1.64KB

Il formato JPEG

- L'algoritmo comprime mantenendo le informazioni sulla luminanza dei pixel, mediando invece le tinte (che l'occhio umano non percepisce accuratamente).
- La caratteristica compressione == smoothing rendono il formato adatto per immagini di tipo fotografico (che non hanno brusche variazioni di colore)

Il formato JPEG

- Proviamo a capire come si implementa ad esempio su un' immagine raster con profondità di colore pari a 24 bit.
- Inizialmente lo spazio dei colori RGB (3 matrici di elementi di 8 bit) viene trasformato in YCbCr, con Y, Cb e Cr matrici a 8 bit con le stesse dimensioni dell'immagine RGB. Y contiene coefficienti riguardanti la Luminanza, Cb e Cr invece quelli riguardanti la Crominanza rispettivamente blu e rossa, nel seguente modo:
  - Y= 0,299R + 0,587G + 0,114B
  - Cb= - 0,1687R - 0,331G + 0,5B (crominanza blu)
  - Cr= 0,5R - 0,419G - 0,081B (crominanza rossa)
- Questo passaggio ed il seguente sono stati pensati perchè l'occhio umano avverte meno le differenze tra colori rispetto a quelle di luminosità.

Il formato JPEG

- In questo passaggio, sulle matrici Cb e Cr vengono calcolate le medie dei colori su blocchi di 2x2 pixel. Supponendo che l'immagine originale RGB abbia dimensione 640x480 pixel, le due matrici di crominanza vengono così portate alla dimensione 320x240.
- A ciascun coefficiente delle 3 matrici viene sottratto 128, portando l'intervallo [0..255] a [-128..127] : vengono così portati a 0 molti dei valori medi.

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Blocco 8X8 estratto da
una delle matrici originali

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

Shift di -128

Il formato JPEG

- Si dividono le tre matrici in blocchi di 8x8 pixel. Se la dimensione della matrice non è divisibile per 8 si aggiungono delle copie dell'ultima riga o colonna in altezza o larghezza.
- Viene applicato per ogni elemento dei blocchi la **DCT** (Trasformata discreta del coseno) che produce ancora 8x8 coefficienti.
  - N rappresenta la dimensione del blocco, 8 pixel. f(m,n) sono i valori in posizione m,n del blocco da trasformare.
  - t(i,j) sono i valori DCT di posizione (i,j), con i,j=0..7.
  - C(i,j) vale 1/N per C(0,j) e C(i,0), 2/N altrimenti. Quindi 1/8 nel primo caso e 1/4 nel secondo.

$$t(i,j) = c(i,j)\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f(m,n)\cos\frac{\pi(2m+1)i}{2N}\cos\frac{\pi(2n+1)j}{2N}$$

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

Il formato JPEG

- Si nota che nella matrice ottenuta, i coefficienti in alto a sinistra rappresentano le basse frequenze mentre quelli via via in basso a destra rappresentano le alte frequenze spaziali ossia i dettagli dell'immagine.
- In particolare il primo coefficiente del blocco trasformato rappresenta la media dei valori del blocco 8x8 originario (detto anche componente continua).

Il formato JPEG

- Un ulteriore passo è la **quantizzazione**.
  - In questa fase ogni elemento di ciascun blocco viene diviso per un coefficiente presente nella ***matrice di quantizzazione*** sempre di dimensione 8x8.
  - Il risultato della divisione viene arrotondato all'intero più vicino. L'eliminazione dei decimali è la principale operazione di compressione distruttiva dello standard **JPEG**.
  - Il tutto è studiato in modo che le frequenze più importanti per l'occhio umano, cioè le più basse, memorizzate nell'angolo superiore sinistro del blocco di 8 x 8, siano preservate, mentre le più alte, la cui perdita è relativamente ininfluente, vengano eliminate.
  - Si utilizza una tabella di quantizzazione per la luminanza e una opportuna per le due crominanze
- Le tabelle di quantizzazione non sono standard ma differiscono dal produttore di software.
- Il fattore di compressione finale dipende molto da quanto si è stati aggressivi nelle divisioni fatte con le matrici di quantizzazione.

Il formato JPEG

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

DCT

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Matrice di quantizzazione

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

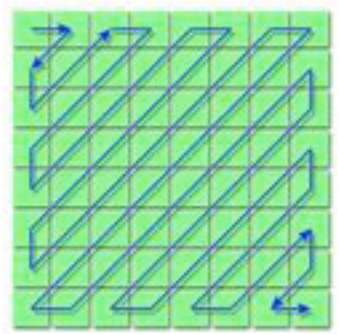
Matrice quantizzata

Il risultato è la concentrazione di
pochi coefficienti diversi da 0 in alto a sinistra e 0 tutti gli altri.

Il formato JPEG

- Il passo finale consiste nella codifica. Gli elementi dell’immagine vengono letti in ordine "zigzag" (frequenze simili così dovrebbero essere più vicine), gli zeri consecutivi vengono codificati con simboli, e poi su quello che resta viene applicata una codifica lossless come ad esempio quella di Huffman (Entropy coding).

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



CODIFICA:

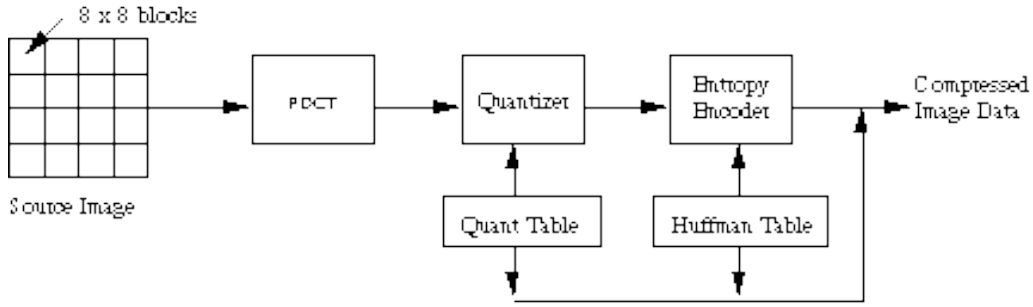
-26, -3, 0, -3, -3, -6, 2, -4, 1 -4, 1, 1, 5, 1, 2, -1, 1, -1, 2, 0, 0, 0, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

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

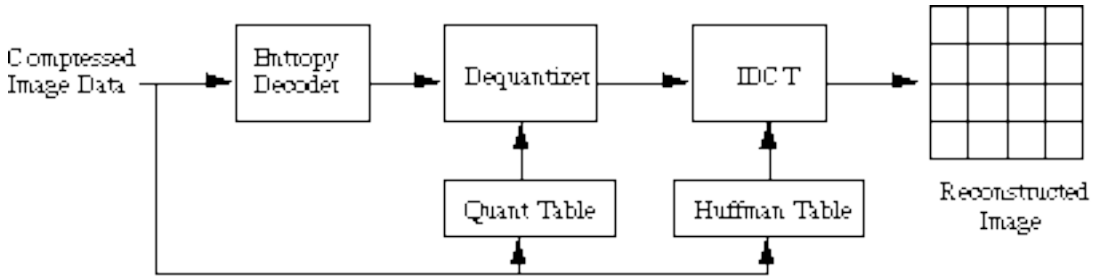
Su questi ultimi dati si può applicare una codifica lossless come ad esempio quella di **Huffman**

Il formato JPEG

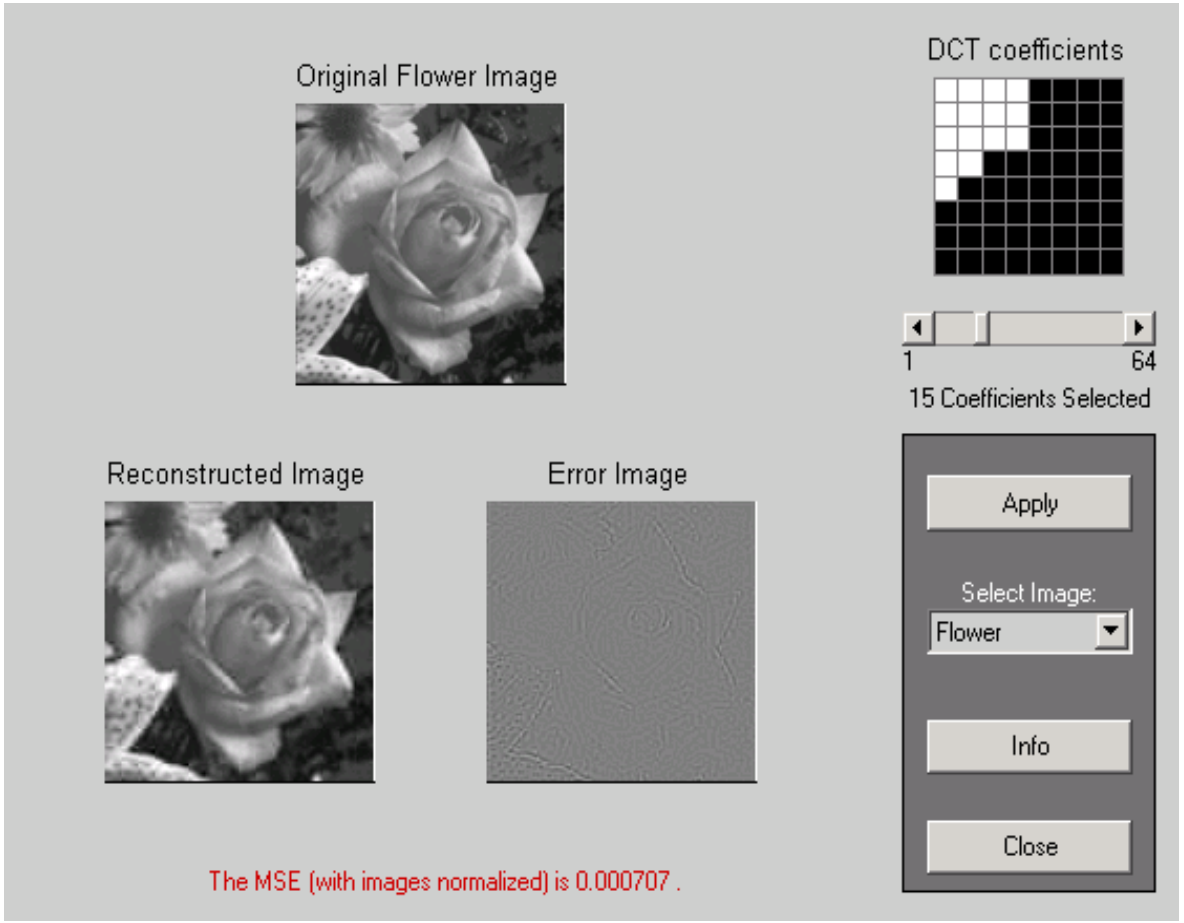
Codifica



Decodifica



MATLAB demo


Original Flower Image
DCT coefficients
1
64
15 Coefficients Selected
Reconstructed Image
Error Image
Apply
Select Image:
Flower
Info
Close
The MSE (with images normalized) is 0.000707 .

JPEG Encoded Images – Gray level Lena

0.9 bpp

0.56 bpp → 14:1

0.37 bpp

0.25 bpp → 32:1

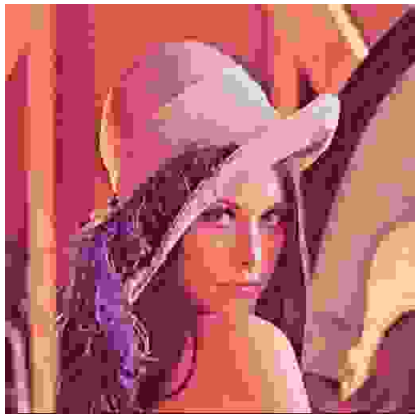
0.13 bpp

JPEG Encoded Images – Color Lena

0.95 bpp

0.53 bpp

0.36 bpp



0.18 bpp

JPEG progressiva

- L'algoritmo di compressione è sempre lo stesso, cambia il formato di salvataggio dei dati nel file.
- Nel file, all'inizio ci sono dati che permettono di ricostruire l'immagine in grossi blocchi di pixels (prima grossa approssimazione).
- Successivamente, in modo incrementale, vengono descritte le variazioni da apportare al primo pezzo per ottenere la versione definitiva dell'immagine.
- In tal modo si riesce ad avere in breve tempo una rappresentazione di massima dell'immagine (fondamentale per navigazione in Internet).

JPEG progressiva

- il formato JPEG progressivo funziona in modo simile al GIF89a interallacciato.
- Nell'esempio si vedono 3 img compresse a 1, 50 e 99% in formato JPEG progressivo.

1%
23 KB

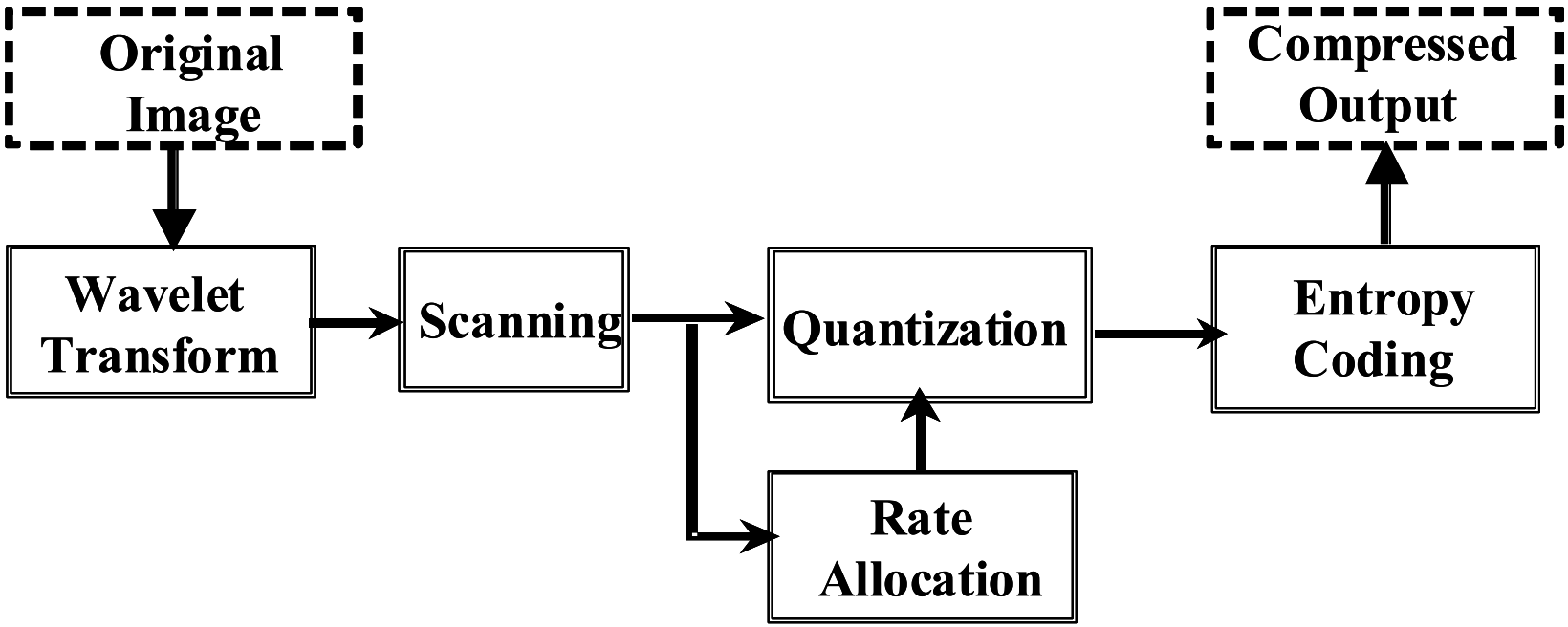
50%
8.33 KB

99%
1.22 KB

Formato JPEG 2000

- Non ancora formalmente adottato, tuttavia costituisce un'estensione dello standard JPEG al fine di aumentarne la flessibilità sia in termini di compressione che di accesso ai dati compressi
- Per esempio, si possono estrarre parti di img compresse al fine della ritrasmissione, visualizzazione, memorizzazione.
- Lo standard si basa sulla codifica wavelet.
  I coefficienti di quantizzazione sono adattati in base alla scala wavelet ed alle sottobande.

Schematic of JPEG2000 Algorithm



Subjective Quality of JPEG2000 Images – Gray level

0.90 bpp

0.56 bpp

0.37 bpp

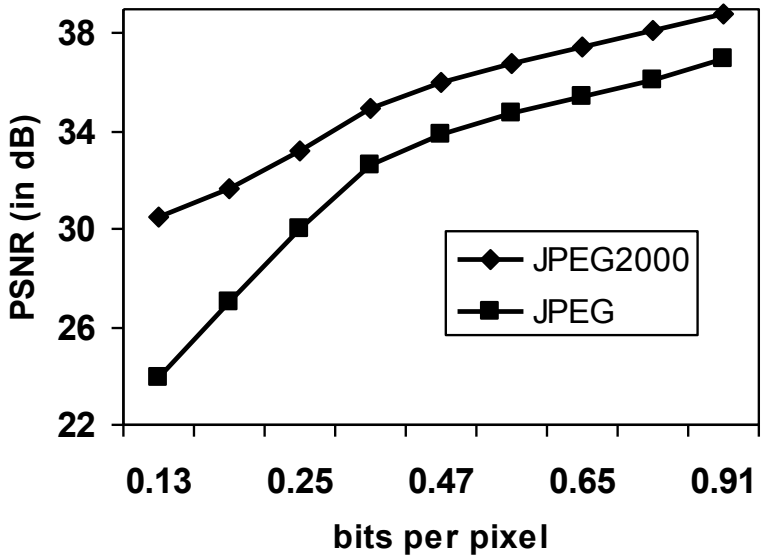
0.25 bpp

0.13 bpp

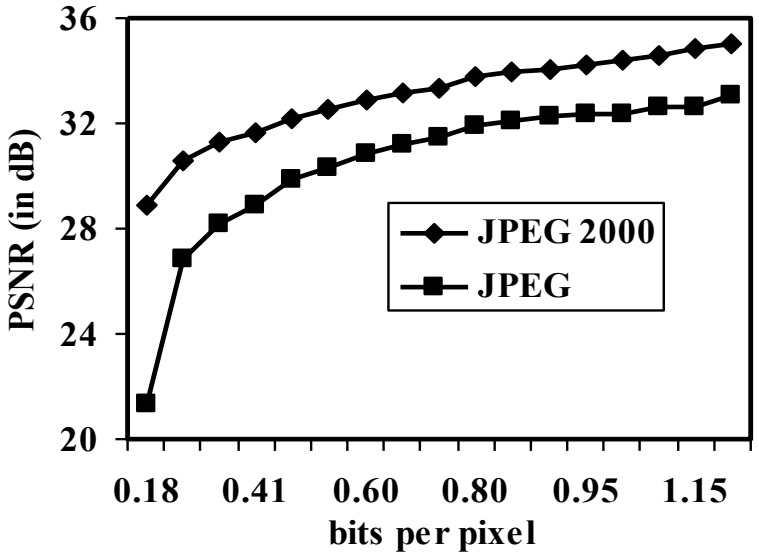
JPEG 0.13 bpp

Objective Quality of JPEG2000 Images

Gray-level image



Color Lena Image



Formato PBM

- PBM - Portable Bit-Map
- Tratta solo immagini monocromatiche (1 bit)
- È stato sviluppato per trasmettere le immagini in rete, tra macchine di diversa piattaforma
- Il file è composto da:
  - l'identificatore di formato (magic number)
  - dimensioni dell'immagine
  - sequenza di bit (1 = nero, 0 = bianco)

Esempio di immagine PBM

Contenuto del file:

```
P1
# flower.pbm
442   395
0 0 0 1 0 0 1
0 0 0 1 1 1 0
1 1 0 …
```

Formato PGM

- PGM - Portable Gray-Map
- Tratta immagini a 256 livelli di grigio (1 byte)
- E' molto usato in visione e in genere nelle applicazioni in cui non servono le informazioni sul colore
- Il file è composto da:
  - l'identificatore di formato (magic number)
  - dimensioni dell'immagine
  - sequenza di byte o di numeri decimali in formato ASCII (0 = nero, 255 = bianco)

Esempio di immagine PGM

Contenuto del file:

```
P2
# flower.pgm
442 395
3 3 4 5 6 10
11 30 30 1 1 30
15 45 45 …
```

Formato PPM

- PMM - Portable Pix-Map
- Tratta immagini a colori (True color)
- Il file è composto da:
  - l'identificatore di formato (magic number)
  - dimensioni dell'immagine
  - valore massimo della componente di colore (in genere 256)
  - sequenza di byte o di numeri decimali in formato ASCII. Ad ogni terna di numeri si ricava il rispettivo valore RGB

Esempio di immagine PPM

Contenuto del file:

```
P3
# flower.ppm
442   395
255
3   3   4
5   6   10
11  30  30
15  45  45 …
```

Il formato PNG

- PNG - Portable Network Graphics
- Formato di tipo bitmap con compressione senza perdita
- Nato per rimpiazzare il formato GIF
- Caratteristiche (comuni con GIF):
  - formati di memorizzazione e visualizzazione progressiva;
  - trasparenza: un colore può essere marcato come trasparente;
  - commenti testuali o altri dati possono essere memorizzati con l'immagine;
  - Indipendenza dalla piattaforma e dall'hardware;
  - Compressione senza perdita basata su LZW (Lempel-Ziv-Welch);

Il formato PNG

- Caratteristiche avanzate:
  - supporta immagini a 256 colori, truecolor (fino a 48 bpp) e a livelli di grigio (fino a 16 bpp);
  - Canale *alpha;*
  - Gamma correction;
  - Error detection nei file;
  - La forma progressiva permette una presentazione più veloce
- Non supporta l'animazione (a tal proposito, esiste il formato MNG)

Il canale *alph*

- Il formato Gif supporta una trasparenza di tipo binario: ogni pixel è classificato come trasparente o come non trasparente;
- il formato PNG supporta fino a 254 livelli di trasparenza (65.534 per alcuni formati);
- RGB RGBA (Red Green Blu Alpha)

Gamma correction

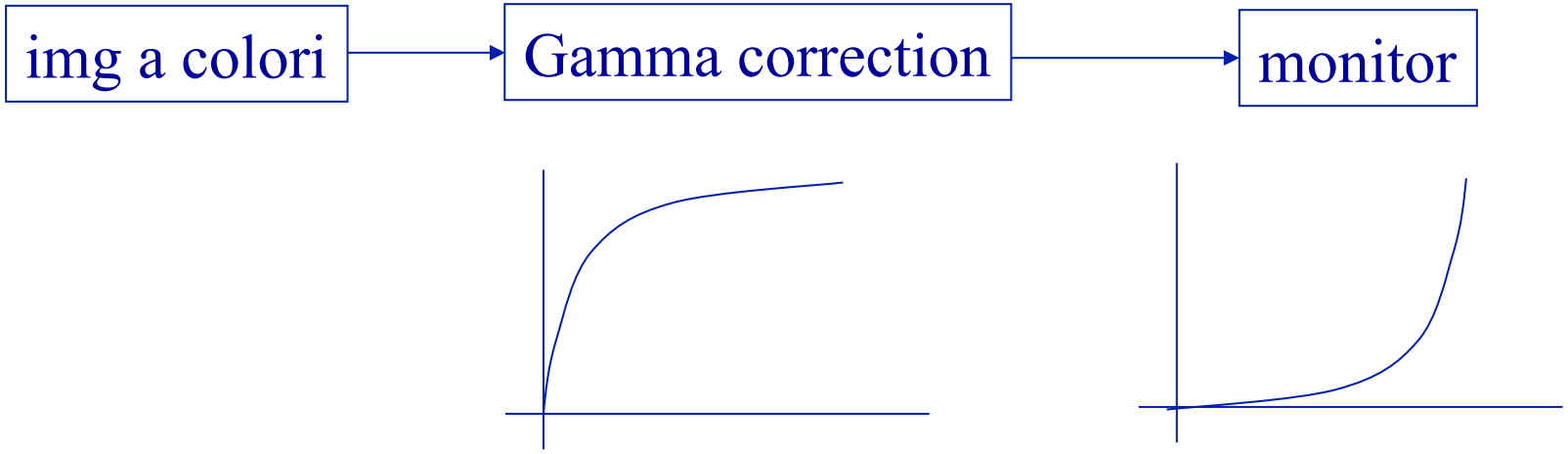
- I procesi di produzione, manipolazione e resa di img a colori assumono implicitamente che esiste una relazione lineare tra i valori RGB e l'intensità del monitor. In realtà non è vero.
- La correzione *gamma* rappresenta un sistema che consente di correggere (almeno parzialmente) le differenze con cui i computer (in particolare i monitor) interpretano i valori dei colori;
  - Es.: un'immagine generata con un Mac sembra molto più scura se visualizzata con un PC, mentre un'immagine creata con un PC risulta essere troppo chiara in un Mac

Gamma correction

- La caratteristica di resa del monitor non è lineare e cambia da monitor a monitor
- Ogni colore viene reso sul monitor mediante una funzione esponenziale (quasi quadratica)

$$C_m = K\, C_i^{\gamma}$$

γ è compreso tra 2.3 e 2.8, C_i è il colore in ingresso al sistema e C_m il colore sul monitor, K una costante



Error detection

- *Magic Signature*: posta all'inizio dei file, controlla che un file binario non venga trasferito in modalità testo (ASCII);
  - gestisce inoltre le differenze CR, CR-LF, LF dei vari sistemi operativi;
- *CRC-32 (cyclic redoundacy check) a livello di blocco*: le immagini PNG (compresse) sono divise in blocchi logici, di ogni blocco viene calcolato un CRC32;
- *Adler 32 checksum*: viene applicato ai dati non compressi, serve per trovare eventuali errori in fase di codifica o di decodifica;

Confronto tra formati

Dimensione file (in bytes) per la stessa immagine codificata in differenti formati:

Image File Format	No. Bytes "Hi"	No. Bytes "Cars"
PGM	595	509,123
GIF	192	138,267
TIF	918	171,430
PS	1,591	345,387
HIPS	700	160,783
JPG (lossless)	684	49,160
JPG (lossy)	619	29,500

"Cars": Img a colori da 347x489

Img originale "Hi": 8x16

"Hi": Compressione con perdita

DICOM standard

- DICOM (*Digital Imaging and COmmunication in Medicine)*
  - Rappresenta lo standard sviluppato nel 1993 per la gestione di immagini digitali ed informazioni ad esse correlate in medicina.
  - Nasce dall'esigenza di distribuire e visualizzare immagini mediche.
  - Lo standard DICOM include in un unico file un'immagine, a chi e cosa questa immagine fa riferimento ed in che modo è stata ottenuta (le informazioni contenute in un file sono divise in gruppi: paziente, studio, serie, immagine).
  - Lo scambio di informazioni si basa su un protocollo di comunicazione in rete.