

10 Compressione

Con il termine compressione dati si indica la tecnica di elaborazione dati che, attuata a mezzo di opportuni algoritmi, permette la riduzione della quantità di bit necessari alla rappresentazione in forma digitale di una informazione

La compressione riduce la dimensione del file e anche la quantità di banda necessaria per una generica trasmissione dati

Un dato è ridondante quando all'interno di un file esso è irrilevante o ripetuto

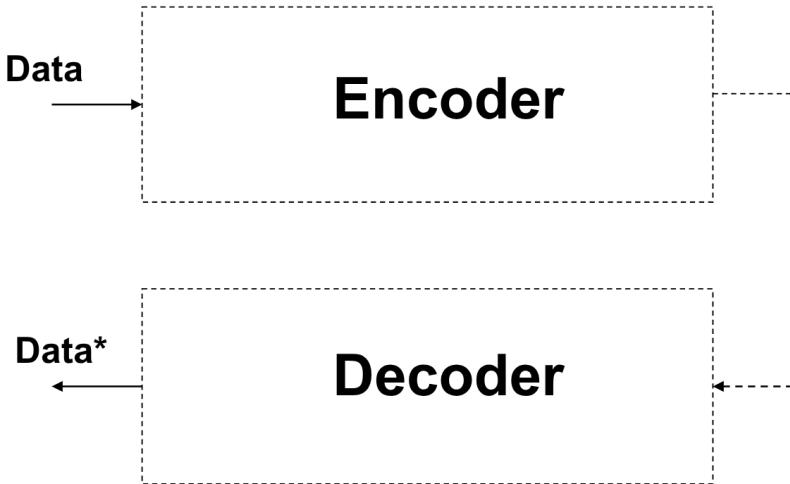
Dal momento che la maggior parte degli array di intensità 2-D sono relazionati spazialmente (per es. ciascun pixel è simile ai pixel del suo intorno, o dipende da esso), l'informazione è replicata inutilmente nei pixel correlati

In una sequenza video, i pixel correlati temporalmente (per es. simili o dipendenti dai pixel dei frame vicini) rappresentano anch'essi un'informazione duplicata

Spesso i dati contengono informazioni ignorate dal sistema sensoriale umano. E' ridondante nel senso che non viene utilizzata

Algoritmo di compressione

Un algoritmo di compressione è una tecnica che elimina la ridondanza di informazione dai dati e consente un risparmio di memoria



Lo possiamo riassumere così

La compressione è basata sul tipo di dati (audio,immagini,video ecc...) e sul *tipo di compressione* ovvero:

- **lossless** (reversibile), cioè senza perdita di informazione
- **lossy** (irreversibile), cioè con una perdita di informazione

Criterio per una buona compressione di tipo lossless

Cercare di raggiungere il limite teorico per la compressione senza perdita che viene fornito dal primo teorema di Shannon, ma prima vediamo altre nozioni importanti:

Frequenza

Sia data una sequenza S di N caratteri tratti da un alfabeto di M possibili caratteri:

a_1, \dots, a_M . Sia f_i la frequenza del carattere a_i cioè:

$$f_i = (\# \text{occorrenze } a_i) / N$$

Entropia

Definiamo entropia E della sequenza di dati S la quantità media di informazione associata ad un singolo simbolo nella sequenza S:

- Se un simbolo è molto prevedibile, porta poca informazione.
- Se un simbolo è totalmente inaspettato, porta molta informazione

Questa è la formula:

$$E = - \sum f_i \log_2(f_i)$$

- f_i : rappresenta la **frequenza** (o probabilità) con cui appare il simbolo i -esimo nella sequenza.
- \log_2 : si usa il logaritmo in base 2 perché l'informazione si misura solitamente in **bit**.
- Il segno meno ($-$) serve a rendere il risultato positivo (poiché il logaritmo di una frazione è negativo)

Un concetto fondamentale è:

- **Più incertezza = Più entropia:** Se non hai idea di quale simbolo uscirà dopo, l'entropia è alta. Se la sequenza è monotona (es. "AAAAAA"), l'entropia è quasi zero perché non c'è sorpresa.
- **Il caso massimo (Equiprobabilità):** L'entropia è massima quando tutti i simboli hanno la stessa probabilità di apparire

Teorema di Shannon

«per una sorgente discreta e a memoria zero, il bit rate minimo è pari all'entropia della sorgente»

I dati possono essere rappresentati senza perdere informazione (lossless) usando almeno un numero di bit pari a: $N * E$

Dove N è il numero di caratteri mentre E è l'entropia

Shannon ci dice il numero minimo di bit ma non ci dice come trovarli, ci serve quindi un algoritmo che permetta di codificare i nostri caratteri usando esattamente il numero di bit ricavati con il teorema di Shannon

Un algoritmo che fa ciò è stato proposto da Huffman

David Huffman ha proposto un semplice algoritmo greedy che permette di ottenere un "dizionario" (cioè una tabella carattere- codifica_binaria) per una compressione quasi ottimale dei dati cioè pari al limite di Shannon con un eccesso di al più qualche bit

Proprietà

- Si tratta di codifica a lunghezza variabile che associa a simboli meno frequenti i codici più lunghi e a simboli più frequenti i codici più corti.
- Si tratta di una codifica in cui nessun codice è prefisso di altri codici.
- È una codifica ottimale perché tende al limite imposto dal teorema di Shannon

Come funziona in breve

- È un algoritmo iterativo. Ad ogni iterazione si scelgono i due nodi con frequenza più bassa.
- Questi due nodi vengono collegati per formare un sottoalbero la cui frequenza del nodo padre è la somma delle frequenze dei due nodi.
- Se ci fossero più nodi con peso minimo se ne scelgono solo due.
- Se c'è un solo nodo con frequenza più bassa si seleziona tale nodo e poi si prende da seconda frequenza più bassa e si seleziona un nodo con quella frequenza
- Si deve creare un albero binario bilanciato.
- Al termine delle iterazioni la radice avrà peso 1.
- Si etichetteranno i rami a sinistra con codice 1 e quelli a destra con codice 0.
- Il codice che si forma procedendo dalla radice alla foglia è il codice abbinato al carattere presente nella foglia stessa

C'è un esempio ben fatto nel power point [11 compressione](#) dalla slide numero 22 alla 26

Dopo l'algoritmo avremo ottenuto la codifica che ci interessa, servirà un altro po di memoria per memorizzare la tabella che contiene la codifica

L'algoritmo di Huffman è usato nei seguenti standard di compressione: CCITT, JBIG2, JPEG, MPEG-1,2,4

Altri algoritmi LOSS LESS

Run Length Encoding

Si voglia comprimere la sequenza formata da 29 bit:

0000011100101110111010111111

Si potrebbe ricordare in alternativa:

5 volte 0, 3 volte 1, 2 volte 0 etc.

O meglio basterebbe accordarsi sul fatto che si inizia con il simbolo 0 e ricordarsi solo la lunghezza dei segmenti (run) di simboli eguali che compongono la sequenza:

5,3,2,1,1,3,1,3,1,1,1,7

Tali valori vanno adesso scritti in binario. Sono sufficienti 3 bit per numero. La sequenza finale (senza spazi) sarà la seguente:

101 011 010 001 001 011 001 011 001 001 001 111

Occorrono quindi 36 bit.

- Non sempre tale codifica porta un risparmio rispetto a quella di input. Infatti nell'esempio della slide precedente, alla fine della codifica abbiamo usato 36 bit contro i 29 della sequenza iniziale.
- La compressione RLE funziona solo se la lunghezza della run è molto grande e prevede un numero di bit superiore a quelli necessari per scrivere il numero che rappresenta la run.
- Se ci sono molte "run" (sequenze di simboli eguali) piuttosto lunghe ricordare la sequenza delle loro lunghezze potrebbe portare un risparmio.

Codifica Differenziale

- Si tratta di una tecnica assai usata.
- Se la sequenza dei valori varia lentamente, invece di registrare i valori è sufficiente ricordarsi del valore iniziale e delle differenze successive.

Esempio:

134, 137, 135, 128, 130, 134, 112, ...

ricorderò il valore iniziale 134 e poi la sequenza delle differenze successive:

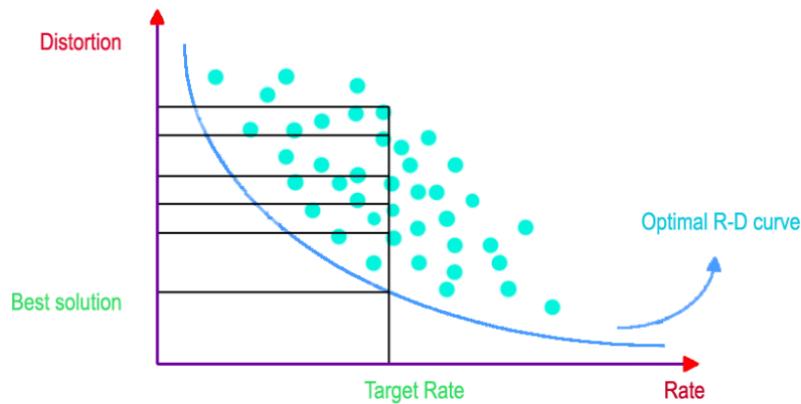
+3,+1,+6,-4,0,-22 ...

Si dimostra sperimentalmente che per le immagini la sequenza delle differenze ha una entropia minore di quella dei valori originali e quindi richiede meno bit per essere memorizzata

Compressione Lossy

Si parla di compressione LOSSY quando i dati possono essere trasformati in modo da essere memorizzati con risparmio di memoria ma con perdita di informazione. *Tale tipo di compressione produce un maggiore risparmio di memoria!*

- Fissata la massima distorsione accettabile l'algoritmo di compressione deve trovare la rappresentazione con il più basso numero di bit.
- Viceversa, fissato il massimo numero di bit accettabile occorre trovare il miglior algoritmo di compressione che a parità di numero di bit mi dia la minima distorsione



L'idea della compressione lossy è: *Se "percettivamente" non è importante: buttalo via!*

- MP3 : applicano questa idea al caso del suono e della musica;
- JPEG : applicano questa idea alle immagini fisse (still images)
- MPEG, AVI, DVX etc: applicano questa idea alle sequenze di immagini (filmati)

Ovviamente una volta buttata via l'informazione non può essere ricostruita esattamente: si tratta di compressione IRREVERSIBILE

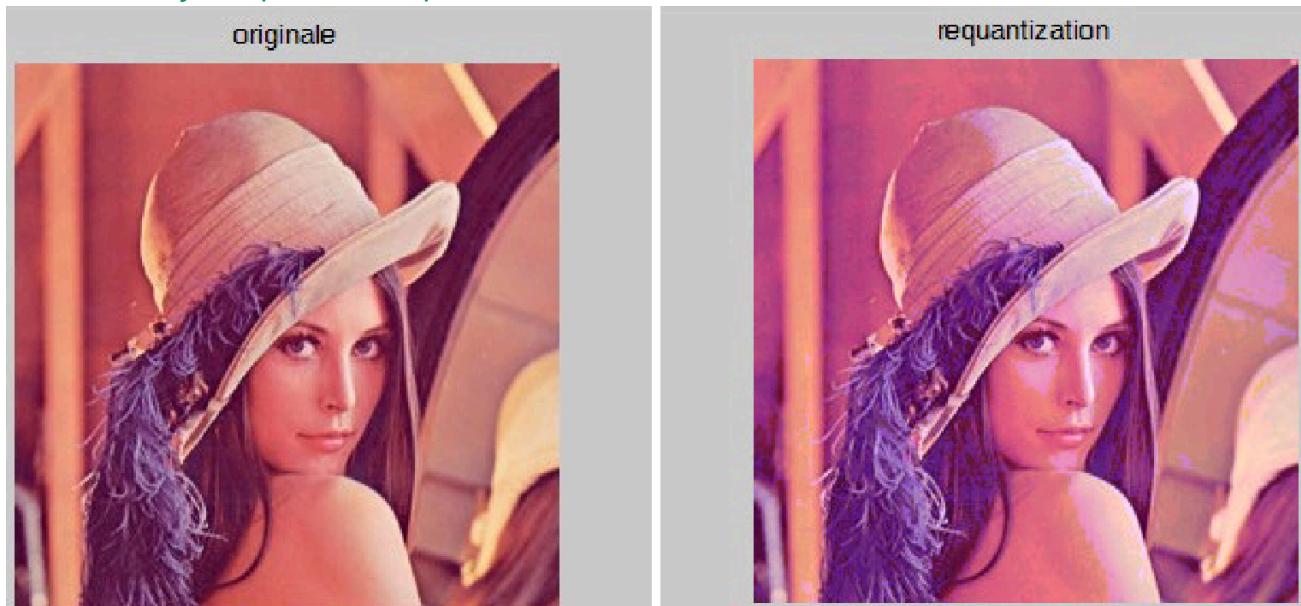
Noi vedremo soltanto:

1. Un algoritmo di requantization;
2. Il JPEG

Requantization

- Si tratta molto semplicemente di una riduzione del numero di livelli disponibili in modo da risparmiare bit per pixel.
- La si realizza "dimenticando" n bit meno significativi per canale.
- Per esempio: RED: da 8 bit si conservano solo i 4 più significativi;
- GREEN: da 8 bit si conservano solo i 6 più significativi;
- BLUE: da 8 bit si conservano solo i 2 più significativi.

- Si risparmia così il 50% dei bit inizialmente necessario! In più: se ci sono meno simboli ... la compressione LZW o Huffman è più efficiente!
- *Ma c'è una forte perdita di qualità*



Lo standard JPEG

Passi fondamentali della codifica JPEG:

1. Pre-processing:
 - i. Color Transform ($RGB \rightarrow YC_bC_r$);
 - ii. Sottocampionamento della crominanza
 - iii. Suddivisione della immagine in sottoimmagini.
2. Trasformazione:
 - i. Discrete Cosine Transform;
 - ii. Quantization;
3. Codifica:
 - i. DC Coefficient Encoding;
 - ii. Zig-zag ordering of AC Coefficients;
 - iii. Entropy Coding (Huffman).

Pre-processing

Iniziamo con il *color transform (i)* appunto dallo spazio RGB a YC_bC_r

Applichiamo la formula:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

sottocampionamento della crominanza (ii)

In questo modo abbiamo un canale separato Y per la luminanza e possiamo sfruttarlo a nostro vantaggio, infatti l'occhio umano è più sensibile alla luminanza che alla crominanza quindi JPEG conserva tutta la luminanza ma campiona i 2 valori di crominanza, scegliendo 1 valore per ogni 4 per C_b e C_r

Questa operazione è lossy ed è irreversibile

partizione dell'immagine

JPEG suddivide l'immagine in quadretti da 8x8 cioè 64 pixel NON sovrapposti, quadretti diversi subiranno una elaborazione differente ed è questa l'origine del problema della quadrattatura che si verifica quando le immagini compresse con JPEG vengono zoommate o stampate

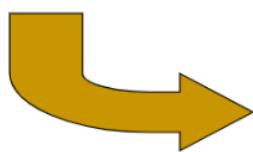
La «quadrettatura» è molto evidente se la compressione è elevata.



Shift dei livelli di grigio

- Prima della applicazione della DCT ai 64 pixel di ciascun blocco viene sottratta una quantità pari a 2^{n-1} , dove 2^n rappresenta il numero massimo di livelli di grigio dell'immagine.
- Se il blocco considerato presenta $256 = 2^8$ possibili livelli di grigio, a ciascun pixel di tale blocco verrà sottratto un offset pari a $128 = 2^7$.
- Con questo processo, noto come shift dei livelli di grigio, il grigio medio (128) diventa 0

Esempio



Trasformazione

Trasformata discreta del coseno DCT(i)

Il JPEG trasforma i blocchi 8 x 8 di pixel secondo un algoritmo detto DCT, è un algoritmo della famiglia delle trasformate di Fourier,

E' stato dimostrato che, statisticamente, tale trasformazione "*decorrela*" al massimo i dati permettendo maggiori rapporti di compressione nella fase successiva di codifica.

Decorrela in questo contesto indica che questa DCT separa i dati non rendendoli più interdipendenti, detto in breve se ho 64x64 pixel di cielo azzurro con variazioni minime ne prendo un valore medio anziche il valore di ogni singolo pixel, risparmiando spazio

Quantizzazione(ii)

Un vantaggio in termini di simboli da usare (e quindi in termini di lunghezza dei codici Huffman) si ottiene se si riduce il numero di "livelli" su cui i coefficienti della DCT possono variare. Tale operazione permette di rappresentare i diversi coefficienti incrementando il fattore di compressione, più precisamente avviene un processo di riduzione del numero di bit necessari per memorizzare un valore intero riducendone la precisione

Il valore F quantizzato si ottiene come :

$$F_{uantizzato} = round(F/)$$

dove Q è un fattore di quantizzazione, F è un numero da quantizzare.

Il valore ricostruito si ottiene moltiplicando $F_{quantizzato}$ per Q

La quantizzazione è un processo irreversibile (lossy)

Si è dimostrato che non è conveniente usare un unico fattore di quantizzazione per tutti i 64 coefficienti della DCT della luminanza, o per quantizzare i valori provenienti dalla DCT delle crominanze, quindi si utilizzano 2 fattori diversi forniti dallo standard o

eventualmente uno fornito dall'utente (a cui però poi va aggiunta la tabella che deve essere trasmessa non essendo uno standard)

Tabella di quantizzazione luminanza

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

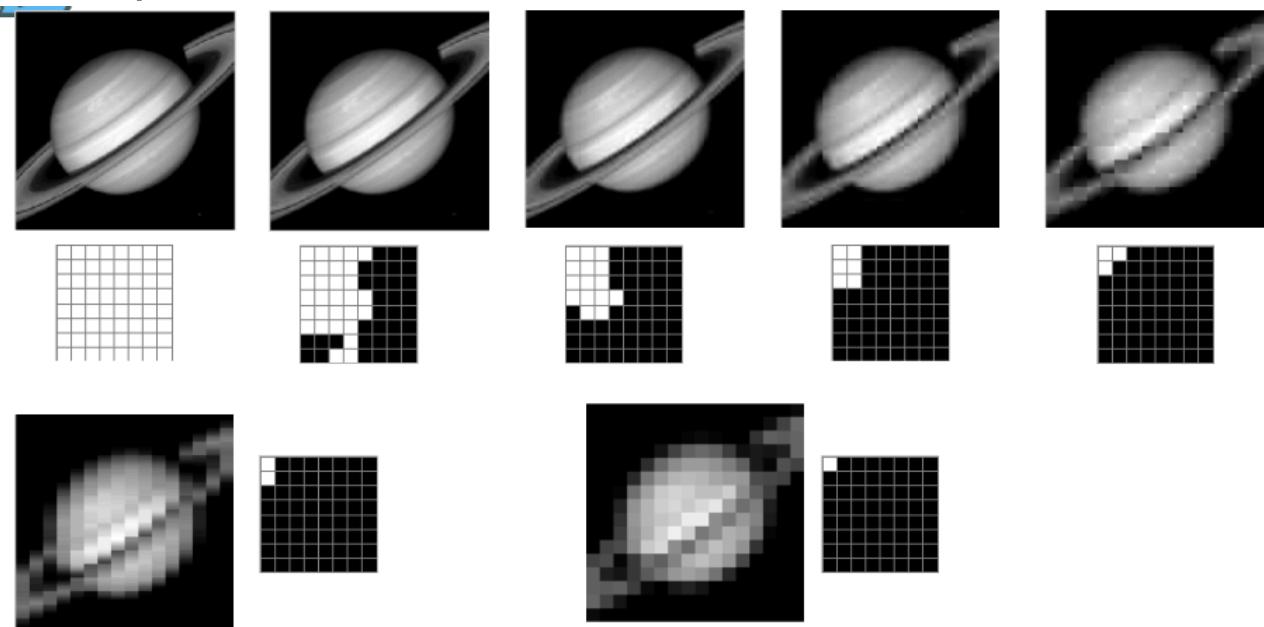
Tabella di quantizzazione crominanza

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Si osservi che un fattore di compressione maggiore comporta una maggiore perdita di informazione e quindi una qualità visiva minore

L'utente del JPEG può scegliere il "grado" di quantizzazione da adottare fornendo un "quality factor" QF che va da 1 a 100. Maggiore è il QF e minore sarà la perdita di informazioni. Quindi fattore di qualità e fattore di compressione sono l'uno l'inverso dell'altro

Effetto della quantizzazione

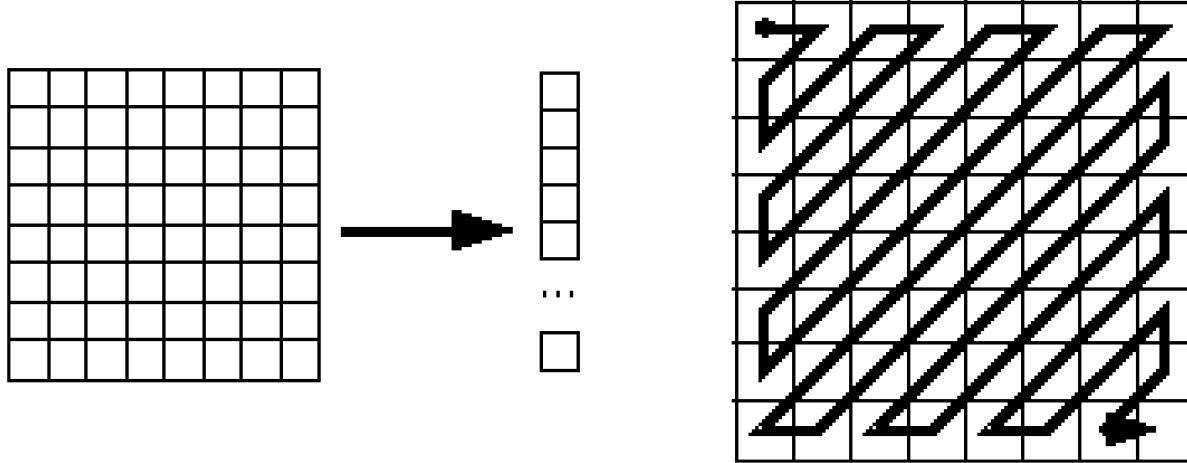


I quadretti neri rappresentano i coefficienti DCT che la quantizzazione ha portato a zero per ogni blocco 8x8 della immagine di Saturno.

Codifica

Tutti i coefficienti vengono riordinati in un vettore 64 x 1 seguendo l'ordinamento "a

serpentina" (per creare lunghe run di zeri) e codificati in un altro stream



A questo punto si hanno 2 differenti codifiche

- I coefficienti DC, cioè quelli che stanno nella posizione (1,1) del blocco 8x8, sono codificati usando una codifica differenziale;
- I coefficienti AC, cioè tutti gli altri del blocco, sono codificati usando una codifica run-length.
- Le tabelle usate di seguito forniscono i codici di Huffman ottenuti sulla base di calcoli statistici preventivi e sono fornite dallo standard

(vedi codifiche → [11 Compressione da slide 75 a 85](#))

■ La sequenza finale sarà

-26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 0 -1 -1 EOB
1010110 0100 001 0100 0101 10001 0110 100011 001 100011 001 001 100101 11100110 110110 0110 11110100 000 1010

- Dove gli spazi sono inseriti solo per migliorare la leggibilità.
- Senza spazi la sequenza diventa (72 bit):

10101100100001010001011000010110100011001100011001001100
101111001101101100110111101000001010

Usando 72 bit contro i 512 bit del blocco di 8 x 8 pixel da 8 bit ciascuno.

Nella ricostruzione

Si deve tornare indietro "ricostruendo" i dati originali (o le loro approssimazioni per i passi

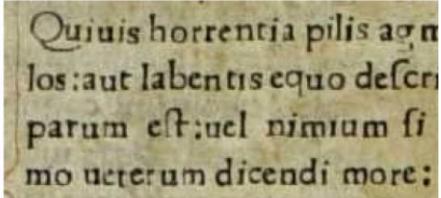
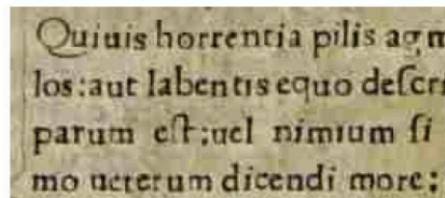
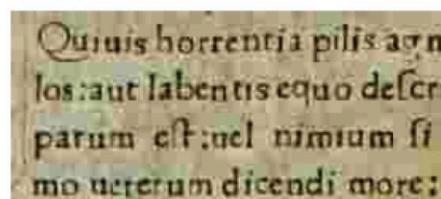
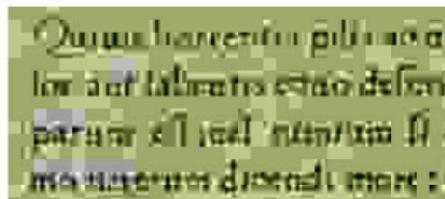
irreversibili)

A destra vediamo il blocco 8x8 di input A sinistra invece quello ricostruito

52	55	61	66	70	61	64	73	58	64	67	64	59	62	70	78
63	59	66	90	109	85	69	72	56	55	67	89	98	88	74	69
62	59	68	113	144	104	66	73	60	50	70	119	141	116	80	64
63	58	71	122	154	106	70	69	69	51	71	128	149	115	77	68
67	61	68	104	126	88	68	70	74	53	64	105	115	84	65	72
79	65	60	70	77	68	58	75	76	57	56	74	75	57	57	74
85	71	64	59	55	61	65	83	83	69	59	60	61	61	67	78
87	79	69	68	65	76	78	94	93	81	67	62	69	80	84	84

Sono diversi tra loro perché JPEG è lossy

■ Fattore di qualità 1, 16, 50, 100



Originale



CR. 75:1 QF=10



CR. 110:1 QF=5



Original Uncompressed (3.2MB)

JPEG High level (179 KB)

JPEG Low level (15 KB)