

Applicare un kernel  $h$  di dimensioni  $s \times t$  ad un'immagine se gli indici del kernel sono disposti in modo da avere gli indici (0,0) al centro:

$$g_{m,n} = \sum_{i=-[s/2]}^{[s/2]-1} \sum_{j=-[t/2]}^{[t/2]-1} (h_{i,j} * f_{m+i,n+j})$$

-1	0	1	
-1	a	b	c
0	d	e	f
1	g	h	i

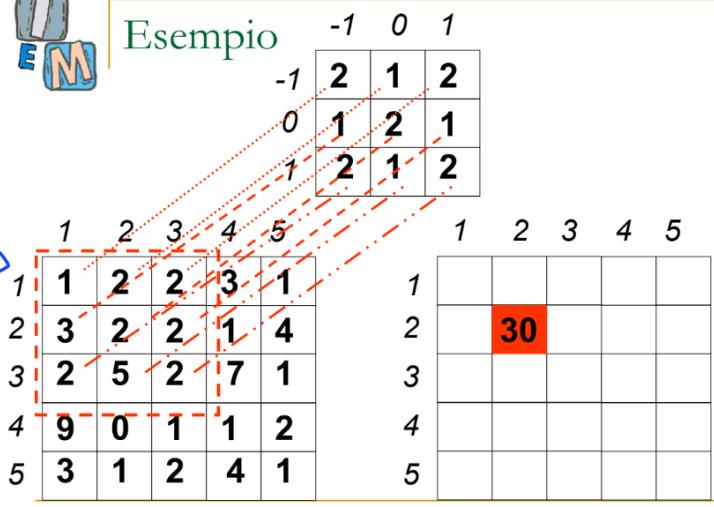
Applicare un kernel  $h$  di dimensioni  $s \times t$  ad un'immagine se gli indici del kernel sono disposti partendo da 1 fino ad arrivare a s e t:

$$g_{m,n} = \sum_{i=1,j=1}^{s,t} h_{i,j} * f_{m+(i-s+[s/2]),n+(j-t+[t/2])}$$

1	2	3	
1	a	b	c
2	d	e	f
3	g	h	i



Esempio



Interazione & Multimedia

21

Fine lezione posta

## Operatori lineari e invarianti per traslazione

- **Operatore Lineare:** un operatore  $F: V \rightarrow W$  si dice lineare se per ogni coppia di vettori  $v_1$  e  $v_2$  in  $V$  e per ogni coppia di scalari  $a$  e  $b$  si ha che:
 
$$F(av_1 + bv_2) = aF(v_1) + bF(v_2)$$
  - Se conosco la base di  $V$  e il comportamento dell'operatore  $F$  su ogni elemento di tale base, posso calcolare il comportamento di  $F$  su ogni elemento di  $V$ .
  - Per descriverlo basta conoscere il suo comportamento su tutte le immagini impulsive.
- **Operatore invariante per traslazione:** un operatore si dice invariante per traslazione quando il suo comportamento sulle immagini impulsive è sempre il medesimo indipendentemente dalla posizione in cui si trova il pixel.
  - Tutti gli operatori puntuali sono invarianti per traslazione (anche se non sono lineari).
- Se un operatore è sia lineare che invariante per traslazione per descriverlo basta conoscere come si comporta in un solo impulso, la point spread function (PSF) diventa la carta d'identità dell'operatore.
- Spesso un operatore su un'immagine prende il nome di filtro.

## Il Kernel

Ad ogni operatore lineare e invariante per traslazione corrisponde una maschera ma vale anche il viceversa: ad ogni maschera corrisponde un operatore. Tale maschera si chiama spesso *kernel* o *maschera di convoluzione*.

La grandezza del kernel può variare fino ad essere infinito, ma per ragioni pratiche vengono usati solo kernel di dimensioni finite. Le dimensioni del kernel influenzano la complessità dell'operatore di filtraggio.

È importante sapere che la complessità dipende anche dal numero di pixel dell'immagine.

## La convoluzione e il problema dei bordi

I filtri lineari e invarianti per traslazione vengono chiamati anche filtri convolutivi.

Per indicare l'operazione di convoluzione si usa la notazione:  $h = f \otimes g$  e gode delle seguenti proprietà:

- *Proprietà commutativa*:  $f \otimes g = g \otimes f$ ;
- *Proprietà associativa*:  $(f \otimes g) \otimes h = f \otimes (g \otimes h)$

Applicare un filtro lineare e invariante per traslazione ad un'immagine è equivalente a calcolare la convoluzione del kernel del filtro con l'immagine.

Un problema ricorrente è come fare la convoluzione e il filtraggio ai bordi, seguono alcune soluzioni:

- Filtrare solo le zone centrali e lasciare i bordi non calcolati;
- Supporre che tutto intorno all'immagine sia 0;
- Riempire le righe e colonne aggiunte in maniera toroidale;
- Riempire le righe e colonne aggiunte con i valori più vicini;

## Operatori locali basici

Si definisce operatore locale, un operatore che preso in input il valore di un pixel ne restituisce uno cambiato che dipende da un intorno limitato centrato sul pixel in ingresso. Gli usi tipici di un operatore locale sono i seguenti:

- Migliorare la qualità delle immagini, come aumentare il contrasto o la nitidezza;
- Estrarre informazioni dalle immagini, come per esempio i bordi;

Tali operazioni si possono pensare come filtraggi dell'immagine, ottenuto facendo la convoluzione tra l'immagine ed il kernel dell'operatore.

Si sono studiati i seguenti operatori locali:

- *Mediano*: è un filtro non lineare che fornisce in uscita il valore mediano dell'intorno del pixel. Consiste in un effetto di smoothing dell'immagine;
- *Massimo*: è un filtro che fornisce in uscita il valore massimo dell'intorno del pixel. Consiste in uno schiarimento dell'immagine;
- *Minimo*: è un filtro che fornisce in uscita il valore minimo dell'intorno del pixel. Consiste in uno oscuramento dell'immagine;
- *N-box (di media)*: sono filtri definiti da kernel di dimensione  $N \times N$  con ogni elemento pari a  $\frac{1}{N^2}$  con  $N$  generalmente dispari. Consiste in un effetto di sfocatura dell'immagine, una sfocatura molto forte in orizzontale e verticale ma meno in diagonale;
- *N-binomiale*: sono filtri definiti da kernel derivati dalla distribuzione binomiale e, poiché tale distribuzione è una approssimazione discreta della distribuzione gaussiana, sono detti filtri gaussiani. Consiste in un effetto di sfocatura sull'immagine uguale in tutte le direzioni ma meno forte rispetto all'n-box;

Tali filtri vengono detti *energy preserving*, poiché la somma dei loro pesi fa 1.

I filtri appena visti servono anche a ridurre il rumore in un'immagine, più è grande è il kernel e migliore sarà il risultato rischiando di aumentare la sfocatura.

## Il rumore

Ci sono due tipo principali di rumore:

- Rumore impulsivo, detto "sale e pepe", caratterizzato dalla frazione dell'immagine modificata (in %):

$$p(z) = \begin{cases} P_a \text{ per } z = a \\ P_b \text{ per } z = b \\ 0 \text{ altrimenti} \end{cases}$$

Il filtro ottimale per la rimozione di questo rumore è il filtro mediano;



- Rumore gaussiano bianco, caratterizzato dalla media e dalla varianza:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}}$$

Il filtro ottimale per la rimozione di questo rumore è il filtro di media.



dove  $z$  rappresenta l'intensità,  $\bar{z}$  è il valore medio<sup>1</sup> (media) di  $z$  e  $\sigma$  è la sua deviazione standard. La deviazione standard al quadrato  $\sigma^2$  è detta *varianza* di  $z$ . Il grafico di questa funzione è mostrato nella Figura 5.2a. Quando  $z$  viene descritta dall'Equazione (5.2-1), circa il 70% dei suoi valori ricade nell'intervallo  $(\bar{z} - \sigma, (\bar{z} + \sigma)]$  e circa il 95% ricade nell'intervallo  $(\bar{z} - 2\sigma, (\bar{z} + 2\sigma)]$ .

## L'estrazione dei bordi

Un altro uso degli operatori locali è estrarre informazioni dalle immagini, una di queste informazioni sono i contorni, o bordi, che sono definiti come delle discontinuità locali della luminanza.

I filtri *edge detector* forniscono immagini in cui sono preservate le variazioni di luminanza ed eliminano tutte le altre informazioni.

Alcuni *edge detector* sono basati sulla derivata prima. In un segnale monodimensionale se calcolo da derivata prima, scopro che i lati sono in corrispondenza dei massimi della derivata: è necessario calcolare la derivata della direzione x e poi della direzione y e poi combinarle.

- Sobel<sub>x</sub>**: [ (-1, -2, -1), (0,0,0), (1,2,1)], fornisce una matrice con i lati orizzontali (e le componenti orizzontali dei lati obliqui) che hanno valori non nulli;
- Prewitt<sub>x</sub>**: [ (-1, -1, -1), (0,0,0), (1,1,1)] ], fornisce una matrice con i lati orizzontali (e le componenti orizzontali dei lati obliqui) che hanno valori non nulli;
- Sobel<sub>y</sub>**: [ (-1,0,1), (-2,0,2), (-1,0,1)], fornisce una matrice dei lati verticali (e le componenti verticali dei lati obliqui) che hanno valori non nulli;
- Prewitt<sub>y</sub>**: [ (-1,0,1), (-1,0,1), (-1,0,1)], fornisce una matrice dei lati verticali (e le componenti verticali dei lati obliqui) che hanno valori non nulli;

Questi filtri possono essere combinati mediante la formula:  $\sqrt{filtro_x^2 + filtro_y^2} = magnitudo$

Si ottengono risultati migliori con algoritmi più sofisticati (e non lineari) o con strategie più intelligenti.

Alcuni *edge detector* sono basati sulla derivata seconda. In un segnale monodimensionale se calcolo da derivata seconda, scopro che i lati passano per lo zero.

Il filtro più diffuso di questa tipologia è il *Laplaciano* avente come maschera  $[ (-1,0,-1), (0,4,0), (-1,0,-1) ]$ , ma dopo aver applicato tale filtro è necessario che si verifichi la condizione *Zero-crossing*, ovvero rispetto al punto in questione vi deve essere nel suo intorno un valore positivo e un valore negativo.

## Filtri di sharpening

Un filtro di sharpening incrementa la nitidezza di un'immagine aumentando il contrasto locale. Possiamo definirla come l'operazione opposta allo sfocamento.

Per ottenere tale effetto si può adottare una maschera derivante dal Laplaciano, che rinforza i lati dell'immagine, come effetto collaterale vi è il rinforzamento del rumore:

$$[ (-1,0,-1), (0,5,0), (-1,0,-1) ]$$

# Capitolo 20

## Dominio delle Frequenze - Serie di Fourier

### 20.1 Il teorema di Fourier

Il teorema di Fourier afferma che qualsiasi segnale periodico può essere rappresentato come somma di funzioni seno e coseno (serie di Fourier), con differenti frequenze e ampiezze. Con la trasformata di Fourier, è possibile esprimere in tal modo anche segnali non periodici.

Convertire un segnale nel cosiddetto "dominio di Fourier" è un processo sempre reversibile grazie all'anttrasformata di Fourier. Ci permetterà quindi di operare sul segnale all'interno del dominio delle frequenze, per poi ricostruire un segnale nel dominio originale.

#### 20.1.1 La trasformata di Fourier applicata alle immagini

L'idea della trasformata di Fourier, è quella di rappresentare l'immagine in un altro spazio di funzioni, passando da un dominio spaziale, a un dominio delle frequenze. Un'immagine può essere vista come una funzione **discreta** in due dimensioni i cui valori rappresentano il livello di grigio di un determinato pixel. La funzione immagine potrebbe essere vista come una funzione variabile in un dominio con una propria frequenza. Sappiamo inoltre che è possibile rappresentare un'immagine come combinazione lineare di basi canoniche. Con la trasformata discreta di Fourier, otteniamo una combinazione lineare di funzioni seno e coseno, con coefficienti pari a  $c_1, c_2, \dots, c_n$ . Il contributo della  $n$ -esima frequenza è direttamente proporzionale al valore di  $c_n$ .

#### 20.1.2 Formule della trasformata discreta di Fourier

Lavorando con segnali digitalizzati, e quindi discretizzati, andremo a lavorare in termini di sommatorie e non di integrali, avvalendoci così della trasformata **discreta** di Fourier. Di seguito, le formule relative alla trasformata.

##### Trasformata della funzione $f(x,y)$

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi(\frac{u}{M}x + \frac{v}{N}y)}$$

per  $u = 0, 1, \dots, M-1$  e  $v = 0, 1, \dots, N-1$

##### Antitrasformata della funzione $F(u,v)$

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{i2\pi(\frac{u}{M}x + \frac{v}{N}y)}$$

per  $x = 0, 1, \dots, M-1$  e  $y = 0, 1, \dots, N-1$

##### Formula di Eulero

Apparentemente questa sommatoria non contiene né il seno, né il coseno.

Rimembriamo quindi la formula di Eulero, per cui, per ogni numero reale  $x$  si ha:

$$e^{ix} = \cos x + i \sin x$$

E quindi

$$e^{-ix} = \cos x - i \sin x$$

### Spettro della trasformata

$$|F(u, v)| = \sqrt{R^2(u, v) + I^2(u, v)}$$

### Angolo di Fase

$$\phi(u, v) = \tan^{-1} \left[ \frac{I(u, v)}{R(u, v)} \right]$$

Ricorda, il  $\lim_{x \rightarrow +\infty} \arctan x = \frac{\pi}{2}$ .

### Potenza Spettrale

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$$

#### 20.1.3 Range dinamico

Applicata la trasformata di Fourier su un'immagine, andremo a ottenere un range di valori molto ampio. Molti valori molto piccoli, pochi numeri molto grandi, nell'ordine dei miliardi. Per visualizzare lo spettro senza anomalie, sfruttiamo la trasformazione logaritmo puntuale, piuttosto che una normalizzazione standard, ovvero: *Che normalizziamo usando*

$$D(u, v) = c \cdot \log(1 + F(u, v))$$

#### 20.1.4 Vantaggi della trasformata di Fourier

Nel dominio spaziale, le coordinate fanno riferimento ad un pixel nella posizione data dalla coordinata. Nello spazio delle frequenze, le coordinate esprimono una determinata frequenza e il suo contributo. In questo modo, possiamo andare a sopprimere frequenze indesiderate o eliminare informazioni superflue per comprimere in maniera lossy. Inoltre, dallo spettro della trasformata e dall'angolo di fase, è possibile analizzare degli elementi relativi all'immagine originale:

- Dallo spettro della trasformata, è possibile capire se sono presenti pattern ripetuti (strutture periodiche) all'interno dell'immagine originale.
- All'interno della fase, troviamo informazioni relative alla posizione delle strutture periodiche in questione.

#### 20.1.5 Operare sullo spettro

I contributi dalle basse frequenze si troveranno quindi al centro dell'immagine, le alte frequenze si troveranno all'esterno.

Con opportuni filtri, che introdurremo successivamente, sarà possibile migliorare l'immagine operando proprio nel dominio delle frequenze, e sopprimendo determinate frequenze (rimuovendo anche pattern ripetuti indesiderati!).

#### 20.1.6 Proprietà della trasformata di Fourier

##### Separabilità

La trasformata di Fourier discreta può essere espressa in forma separabile. È possibile andare a separare le due dimensioni, lavorandoci separatamente. *Le formule diverte:  $F(u, v) = \frac{1}{n} \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} k(x, y) e^{-j2\pi \frac{ux}{n}}$*

##### Traslazione

$$k(x, y) = \frac{1}{n} \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} f(x, y) e^{-j2\pi \frac{uv}{n}}$$

Operazioni di traslazione (non rotazione!) sull'immagine nel dominio spaziale non alterano lo spettro. Inoltre, solo nello spettro, è possibile andare a effettuare una scambio tra il primo e il terzo quadrante, e il primo e il secondo, per visualizzare al centro le basse frequenze, e ai bordi le alte, dando un'immagine più semplice da analizzare ad occhio.

##### Valore continuo

Il valore medio (o valore continuo) è il valore di  $F(0, 0)$ .

### 20.1.7 Fast Fourier Transform

*prozie alle proprietà di separabilità*

Esiste un'algoritmo per la trasformata di Fourier, di complessità  $\Theta(n \log n)$ . A causa della doppia sommatoria, la complessità della formula classica è  $n^2$ . Non lo studieremo, ma è buono sapere che esista.

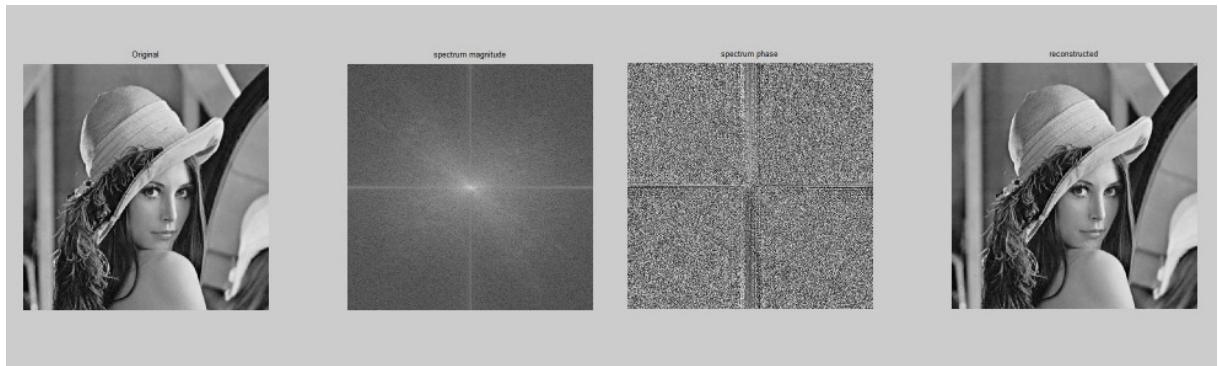


Figura 20.1: Lenna, spettro, fase e risultato dell'antitrasformata

### 20.1.8 Frequenze

All'interno delle immagini, le frequenze sono legate alla velocità di variazione dei toni. Per questo motivo, le basse frequenze sono quelle legate al tono generale dell'immagine, mentre le alte frequenze contengono informazioni relative a variazioni molto veloci, come nel caso del rumore.

## 20.2 Teorema della convoluzione

La trasformata della convoluzione di due segnali nel dominio spaziale equivale al prodotto delle trasformate dei due segnali.

Ciò significa anche che la convoluzione di due segnali nel dominio spaziale equivale all'antitrasformata del prodotto delle trasformate di due segnali.

$$F(f \otimes h) = F(f) \cdot F(h)$$

Con  $f$  immagine,  $h$  kernel,  $\otimes$  convoluzione,  $F$  trasformata di Fourier.

Applicare questo metodo nel dominio di frequenze ha complessità  $O(n \log n)$ .

### 20.2.1 Conseguenza di questo teorema

Questo teorema è importantissimo, in quanto ci permette di applicare filtri convolutivi senza utilizzare la convoluzione (un'operazione notoriamente lenta!).

Basterà infatti:

1. Adattare il kernel alla dimensione e forma dell'immagine su cui andrà applicato, aggiungendo zeri in maniera opportuna.
2. Trasformare l'immagine da modificare e il kernel (passaggio non richiesto quando si lavora con i filtri Guassiani)
3. Effettuare il prodotto puntuale tra la trasformata dell'immagine e quella del Kernel.
4. Calcolare l'antitrasformata di Fourier dell'immagine ottenuta dal prodotto puntuale, per tornare nel dominio spaziale.

### Filtri di smoothing o passa-basso (low-pass)

I filtri di *smoothing* sono dei filtri *low-pass (passa-basso)* che eliminano il rumore; ciò corrisponde nel dominio delle frequenze ad attenuare le componenti ad alta frequenza.

I principali filtri passa-basso sono:

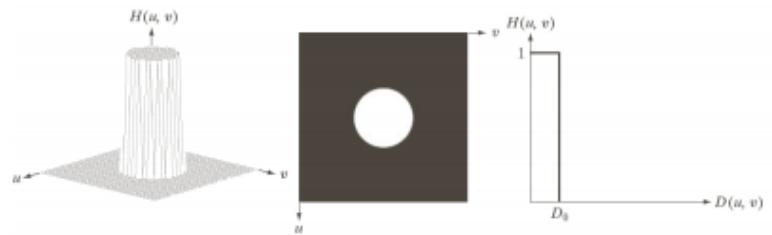
Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$	$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}}$	$H(u, v) = e^{-D^2(u,v)/2D_0^2}$

- Filtri ideali:** tali filtri eliminano totalmente tutte le componenti di frequenza che nel rettangolo delle frequenze distano dall'origine più di  $D_0$ , la cosiddetta *frequenza di taglio*:

$$H(u, v) = \begin{cases} 1 & \text{se } D(u, v) \leq D_0 \\ 0 & \text{altrimenti} \end{cases}$$

Dove  $D(u, v)$  è la distanza dall'origine ed è data da:

$$D(u, v) = \sqrt{\left(u - \frac{P}{2}\right)^2 + \left(v - \frac{Q}{2}\right)^2}$$



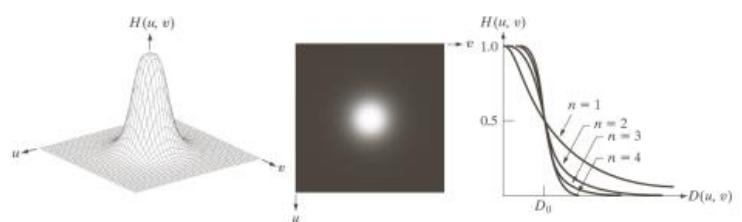
Il termine "ideale" indica che tutte le frequenze all'interno di un cerchio di raggio  $D_0$  sono trasferite senza attenuazione, mentre tutte quelle esterne sono, invece, annullate.

Tali filtri causano un forte fenomeno di sfocatura ad anello detto *ringing*, legato alla funzione  $h(x, y)$  dell'anti-trasformata. Gli anelli generati hanno un raggio inversamente proporzionale alla frequenza di taglio.

- Filtri butterworth:** la funzione di trasferimento di tale filtro di ordine  $n$  e frequenza di taglio  $D_0$  è:

$$H(u, v) = \frac{1}{1 + \left(\frac{D(u, v)}{D_0}\right)^{2n}}$$

A differenza dei filtri passa-basso ideali, il filtro low-pass di butterworth non determina un taglio netto in frequenza



in quanto la sua funzione di

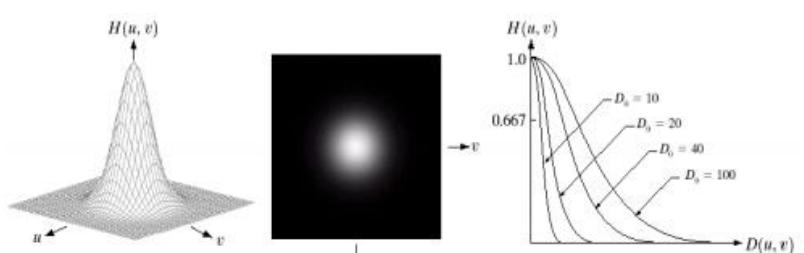
trasferimento non introduce una discontinuità brusca, possiamo dire quindi che viene effettuata una attenuazione graduale.

Al crescere dell'ordine  $n$  del filtro l'attenuazione diventa sempre più netta e il filtro sempre più simile a quello ideale, cominciando ad assumere le stesse caratteristiche e gli stessi difetti;

- Filtri gaussiani:** la funzione di trasferimento di tale filtro di frequenza di taglio  $D_0$  è:

$$H(u, v) = e^{\frac{-D^2(u, v)}{2D_0^2}}$$

Il principale vantaggio di questi filtri è avere come trasformata di Fourier ancora una gaussiana. Presentano caratteristiche simili ai filtri di Butterworth di piccolo ordine, ma con una più semplice realizzazione.



I principali impieghi di un filtro passa-basso sono:

- Il riconoscimento di caratteri;
- La riduzione del rumore;
- L'elaborazione di immagini aeree e satellitari;

## Filtri di sharpening o passa-alto (high-pass)

Poiché i bordi e gli altri rapidi cambiamenti nei livelli di grigio sono legati alle componenti ad alta frequenza, è possibile attuare lo sharpening dell'immagine attraverso i filtri *high-pass* (*passa-alto*) che eliminano le componenti a bassa frequenza.

I principali filtri passa-alto sono:

Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$	$H(u, v) = \frac{1}{1 + [D_0/D(u, v)]^{2n}}$	$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2}$

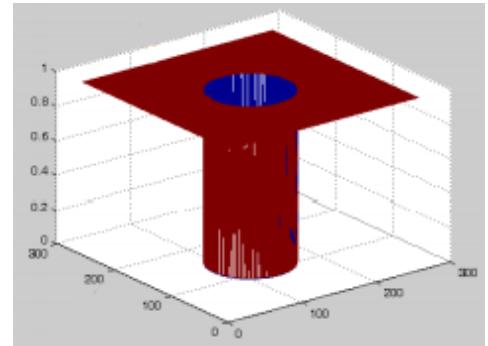
- Filtri ideali:** tali filtri eliminano totalmente tutte le componenti di frequenza che nel rettangolo delle frequenze distano dall'origine meno di  $D_0$ , la cosiddetta *frequenza di taglio*:

$$H(u, v) = \begin{cases} 0 & \text{se } D(u, v) \leq D_0 \\ 1 & \text{altrimenti} \end{cases}$$

Dove  $D(u, v)$  è la distanza dall'origine ed è data da:

$$D(v, u) = \sqrt{\left(u - \frac{P}{2}\right)^2 + \left(v - \frac{Q}{2}\right)^2}$$

Il termine "ideale" indica che tutte le frequenze all'esterno di un cerchio di raggio  $D_0$  sono trasferite senza attenuazione, mentre tutte quelle interne sono, invece, annullate.



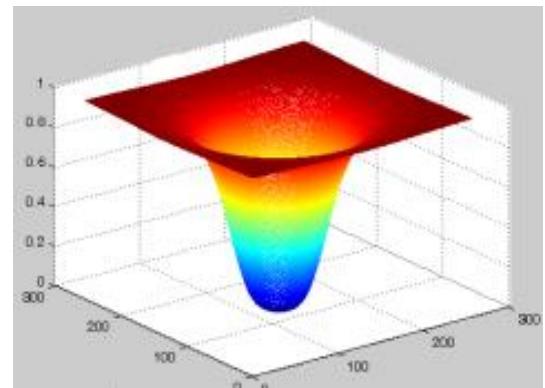
Tali filtri causano un forte fenomeno di sfocatura ad anello detto *ringing*, legato alla funzione  $h(x, y)$  dell'anti-trasformata. Gli anelli generati hanno un raggio inversamente proporzionale alla frequenza di taglio.

- Filtri butterworth:** la funzione di trasferimento di tale filtro di ordine  $n$  e frequenza di taglio  $D_0$  è:

$$H(u, v) = \frac{1}{1 + \left(\frac{D_0}{D(u, v)}\right)^{2n}}$$

A differenza dei filtri passa-basso ideali, il filtro high-pass di butterworth non determina un taglio netto in frequenza in quanto la sua funzione di trasferimento non introduce una discontinuità brusca, possiamo dire quindi che viene effettuata una attenuazione graduale.

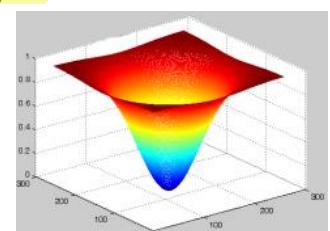
Al crescere dell'ordine  $n$  del filtro l'attenuazione diventa sempre più netta e il filtro sempre più simile a quello ideale, cominciando ad assumere le stesse caratteristiche e gli stessi difetti;



- Filtri gaussiani:** la funzione di trasferimento di tale filtro di frequenza di taglio  $D_0$  è:

$$H(u, v) = 1 - e^{-\frac{-D^2(u, v)}{2D_0^2}}$$

Il principale vantaggio di questi filtri è avere come trasformata di Fourier ancora una gaussiana. Presentano caratteristiche simili ai filtri di Butterworth di piccolo ordine, ma con una più semplice realizzazione. Introduce smoothing.



## Filtri di band-reject

Individuato un range di frequenze su cui operare, il filtro *band-reject* costruisce una maschera opportuna per eseguire tale filtraggio.

I principali filtri band-reject sono:

- Filtri ideali:** tali filtri eliminano totalmente tutte le componenti di frequenza che nel rettangolo delle frequenze non entrano nel range desiderato di centro  $D_0$ , la cosiddetta *frequenza di taglio*:

$$H(u, v) = \begin{cases} 0 & \text{se } D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & \text{altrimenti} \end{cases}$$

Dove  $D(u, v)$  è la distanza dall'origine ed è data da:

$$D(v, u) = \sqrt{\left(u - \frac{P}{2}\right)^2 + \left(v - \frac{Q}{2}\right)^2}$$

- Filtri butterworth:** la funzione di trasferimento di tale filtro di ordine  $n$  e frequenza di taglio  $D_0$  è:

$$H(u, v) = \frac{1}{1 + \left(\frac{DW}{D^2 - D_0^2}\right)^{2n}}$$

A differenza dei filtri passa-basso ideali, il filtro high-pass di butterworth non determina un taglio netto in

frequenza in quanto la sua funzione di trasferimento non introduce una discontinuità brusca, possiamo dire quindi che viene effettuata una attenuazione graduale.

Al crescere dell'ordine  $n$  del filtro l'attenuazione diventa sempre più netta e il filtro sempre più simile a quello ideale, cominciando ad assumere le stesse caratteristiche e gli stessi difetti;

- **Filtri gaussiani:** la funzione di trasferimento di tale filtro *di frequenza di taglio  $D_0$*  è:

$$H(u, v) = 1 - e^{-\left[\frac{D^2(u,v)-D_0^2}{D*W}\right]^2}$$

Il principale vantaggio di questi filtri è avere come trasformata di Fourier ancora una gaussiana. Presentano caratteristiche simili ai filtri di Butterworth di piccolo ordine, ma con una più semplice realizzazione.

Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u, v) = \frac{1}{1 + \left[\frac{DW}{D^2 - D_0^2}\right]^{2n}}$	$H(u, v) = 1 - e^{-\left[\frac{D^2 - D_0^2}{DW}\right]^2}$

# Capitolo 21

# Compressione

## 21.1 Introduzione alla compressione

Definiamo la compressione come la rielaborazione dei dati da rappresentare, col fine di ridurre le dimensioni in termini di memoria. Le varie strategie di compressione che conosciamo, sfruttano strategie differenti

### 21.1.1 Approcci alla compressione

- **Sfruttare le ridondanze tra i dati**

Riducendo il numero di bit usato per rappresentare in memoria sequenze di dati ridondanti, preservando tuttavia tutte le informazioni (lossless).

- **Eliminare dati "trascurabili"**

Sfruttando informazioni relative alla percezione del media, è possibile comprimere i file ad hoc. Ad esempio, nei file audio è possibile tagliare estremi di banda. Ci sarà quindi una perdita di dati (lossy) accuratamente scelti per ridurre al minimo le differenze percettive. Non applicabile su testi.

### 21.1.2 Cos'è un codice

Un codice è un sistema di simboli utilizzati per rappresentare una certa quantità di informazioni. Ad ogni pezzo di informazioni e a ogni evento coincide una sequenza di simboli codificati, chiamata codeword, di lunghezza variabile.

### 21.1.3 Encoder e Decoder

Il processo di compressione e decompressione implica l'ausilio di un encoder, per effettuare la compressione, e di un decoder, per decomprimere il file.

Il file in entrata all'encoder e in uscita al decoder, sarà diverso nel caso della compressione lossy, uguale nella compressione lossless.

La compressione lossy è chiaramente non reversibile.

## 21.2 Compressione lossless

La compressione lossless si basa sulla rimozione della ridondanza, senza perdite di dati veri e propri. Tuttavia, esiste un limite teorico sotto cui la compressione lossless non può andare, ed è espresso dal teorema di Shannon per la compressione, trattato in un paragrafo successivo.

### 21.2.1 Frequenza

Data una sequenza  $S$  di  $N$  caratteri tratti da un alfabeto di  $M$  possibili caratteri  $(a_1, \dots, a_m)$ .

$$f_i = \frac{\text{numero di occorrenze di } a_i}{N}$$

### 21.2.2 Entropia

Definiamo entropia  $E$  della sequenza di dati  $S$  la quantità media di informazione associata alla singola generazione di un simbolo nella sequenza  $S$ .

$$E = - \sum_{i \in S} f_i \log_2(f_i)$$

Breve esempio sulla stringa 'AABBAACB'.

$$E = -\left(\frac{4}{8} \log_2\left(\frac{4}{8}\right) + \frac{3}{8} \log_2\left(\frac{3}{8}\right) + \frac{1}{8} \log_2\left(\frac{1}{8}\right)\right)$$

L'entropia è anche chiamata informazione media, e indica il numero di bit medi utilizzati per rappresentare un carattere della sequenza. Intuiamo che  $N \cdot E$  è il limite teorico della compressione lossless, ovvero il numero minimo di bit utilizzabile per comprimere la stringa.

#### Teorema del cambio di base

$$\log_a(b) = \frac{\log_c(b)}{\log_c(a)}$$

Potrebbe servire in futuro.

### 21.2.3 Teorema di Shannon (per la compressione)

Per una sorgente discreta e a memoria zero, il bitrate minimo è pari all'entropia della sorgente. I dati possono essere rappresentati senza perdere informazione (lossless) usando almeno un numero di bit pari a

$$N \cdot E$$

#### Sorgente discreta?

Una sorgente discreta implica che i possibili caratteri sono in quantità finita, come in un alfabeto o nella codifica di un'immagine. La sorgente deve essere discreta, ma nell'ambito del digitale è sempre così, quindi nulla di cui preoccuparsi realisticamente.

#### Memoria zero?

Stringhe a memoria zero, sono costituite da simboli indipendenti tra loro. In dati non a memoria zero, questa dipendenza tra i simboli implica che la probabilità che un simbolo si presenti nella stringa, aumenta o diminuisce in funzione di altri simboli.

### 21.2.4 Codifica di Huffman

Si tratta di una codifica a lunghezza variabile che associa a simboli meno frequenti codeword (codici) più lunghi e viceversa.

In questa codifica, nessuna codeword è prefisso di altre codeword. In questo modo, la decodifica è garantita come univoca, non ambigua. Questa codifica tenderà al limite imposto dal teorema di Shannon.

## Come funziona?

1. Viene creato un albero binario bilanciato
2. Al termine delle iterazioni la radice avrà peso 1
3. Si etichetteranno i rami a sinistra con codice 1 e quelli a destra con codice 0 (o viceversa)
4. Il codice che si forma procedendo dalla radice alla foglia è il codice abbinato al carattere presente nella foglia stessa

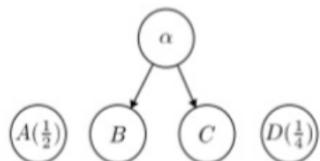
### Esempio di Codifica di Huffman

Dati : 'AABABCAACAAADDDD'

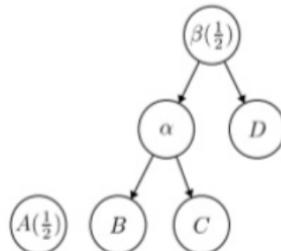
1. Questi nodi contengono i simboli e la loro frequenza all'interno del testo.



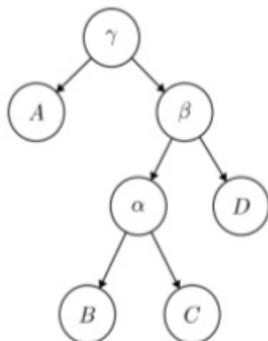
2. Rendo, due nodi di frequenza più bassa, figli di un nuovo nodo di frequenza uguale alla somma dei figli, che chiamerò  $\alpha$ .



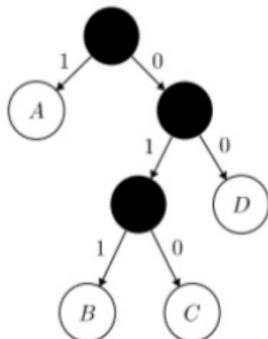
3. Continuo iterativamente, creo un nodo  $\beta$  che ha per figli  $\alpha$  e  $D$ .



4. Un'ultima volta.



5. Ecco il nostro albero di Huffman, completo di archi 0 e 1! I codici delle foglie andranno letti dal basso verso l'alto.



$A : 1; B : 011; C : 010; D : 00$

## Implicazioni della codifica di Huffman

Per decodificare il file, bisogna avere una struttura dati contenente, appunto, la associazione stringa-simbolo della decodifica scelta.

### 21.2.5 Altri algoritmi di compressione lossless

- **Run-Length-Encoding.**

Si basa sul comprimere una sequenza di bit (0 e 1) esprimendola in termini di numeri che indicano il numero di volte in cui si presenta 1, poi 0, poi 1, poi 0, poi .... Non è una codifica sempre vantaggiosa, ma nel caso in cui le sequenze di caratteri uguali sono molto lunghe, è il miglior modo per procedere. Agendo direttamente su i bit-planes, è un buon algoritmo.

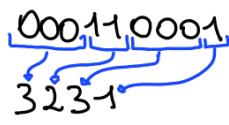
- **Codifica "differenziale".**

Per memorizzare una successione di valori, ad esempio numerici, potremmo memorizzare tutti i singoli elementi di questa successione. In alternativa, con la codifica differenziale, possiamo memorizzare il primo termine, poi la differenza tra il primo e il secondo, poi il secondo e il terzo, e così via.

$$134, 137, 135, 128, \dots \rightarrow (134) - 3, +2, -7, \dots$$

Come nella buona parte dei casi, usare questa codifica in combinazione con Huffman ci da ottimi risultati. La sequenza della codifica differenziale di una sequenza di numeri ha entropia più bassa dell'originale, aumentando l'efficacia di Huffman.

In generale, è sempre meglio si presentino elementi ridondanti per applicare compressioni di questo tipo.



## Compressione lossy e criterio per buona compressione lossy

Si parla di **compressione lossy** quando i dati possono essere **trasformati** in modo da essere memorizzati con un risparmio di memoria ma con una perdita irreversibile di informazione.

Il criterio per una buona compressione di tipo lossy ha due modalità:

- Fissata la massima distorsione accettabile, l'algoritmo di compressione deve trovare la rappresentazione con il più basso numero di bit;
- Fissato il massimo numero di bit accettabile, bisogna trovare il miglior algoritmo di compressione che a parità di numero di bit fornisce la minima distorsione.

Generalmente l'idea principale di una compressione lossy per le immagini è: se non è percettivamente rilevante, buttalo via.

## Compressione lossy: Requantization

Il metodo di compressione **requantization** è alquanto semplice in quanto si tratta di una riduzione del numero di livelli in modo da risparmiare bit per pixel: vengono scartati *n* pixel meno significativi per canale.



## Compressione lossy: JPEG (Joint Photographic Experts Group)

La codifica JPEG è abbastanza complessa poiché viene eseguita in 3 passi, i quali costituiti da sotto-passi:

### 1) Il pre-procesing:

- **Trasformazione dei colori:** si tratta di passare dallo spazio di colore *RGB* a quello *YC<sub>b</sub>C<sub>r</sub>*, questo è necessario per il passaggio successivo.
- **Sotto-campionamento della crominanza:** dato che l'occhio umano è più sensibile alla luminanza che alla crominanza si decide di mantenere tutti i valori della luminanza e prendere, invece, 1 valore ogni 4 per il *C<sub>b</sub>* e *C<sub>r</sub>*. Questo passaggio comporta una perdita delle informazioni ed è perciò irreversibile.
- **Partizione dell'immagine:** per approfittare al meglio della ridondanza (che è localmente maggiore che sull'intera immagine) e per semplificare i calcoli, JPEG procede dividendo l'immagine in quadranti 8x8 da 64 pixel non sovrapposti;
  - i. Quadranti diversi subiranno una elaborazione differente e indipendente dagli altri, da qui nasce la problematica della quadrettatura caratteristica delle immagini compresse in JPEG.

### 2) Trasformazione:

- i. 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. Tale processo prende il nome di *shift dei livelli di grigio*. In un'immagine a 8 bit il valore di grigio 128 diventerebbe 0;

# Usando le Formule delle trasformate inverse

- a. **Discrete Cosine Trasform:** viene applicata la DCT a ogni quadrante  $8 \times 8$  da 64 pixel al fine di decorrelare al massimo di dati permettendo così maggiori rapporti di compressione nella fase di codifica;

matematicamente descriviamo un'immagine attraverso le sue base impulsive, ovvero decomponendo il valore di luminosità pixel per pixel, dopo la DCT lo ridisegniamo come le somme di 64 onde, i coefficienti ottenuti ci dicono quante di quelle specifiche onde servono per ricreare l'immagine.

- vi. Il coefficiente DC è il primo valore in alto a sinistra;  
 vii. Tutti gli altri 63 valori sono detti coefficienti AC;

La trasformata e l'anti-trasformata sono:

$$\text{trasformata} \rightarrow F(u, v) = \frac{1}{N} \left[ \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C(u) * C(v) * f(x, y) * \cos\left(\frac{(2x+1)u\pi}{2N}\right) * \cos\left(\frac{(2y+1)v\pi}{2N}\right) \right]$$

$$\text{anti-trasformata} \rightarrow f(x, y) = \frac{1}{N} \left[ \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u) * C(v) * F(u, v) * \cos\left(\frac{(2x+1)u\pi}{2N}\right) * \cos\left(\frac{(2y+1)v\pi}{2N}\right) \right]$$

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

- b. Quantizzazione: un ulteriore passo lossy che consiste nel ridurre i livelli. Si ottiene col formalismo

$$F_{\text{quantizzato}} = \text{round}\left(\frac{F}{Q}\right)$$

*più è alto più info si perdono*

## 3) Codifica:

- a. Codifica dei coefficienti DC:  
 b. Codifica dei coefficienti AC:

alle fine del PDF

le quantizzazioni ha reso  
a quasi tutti i valori un buon  
e dx (alta frequenza). L'apprendere a 3p  
3p rappresentano alle fine  
gli zeri

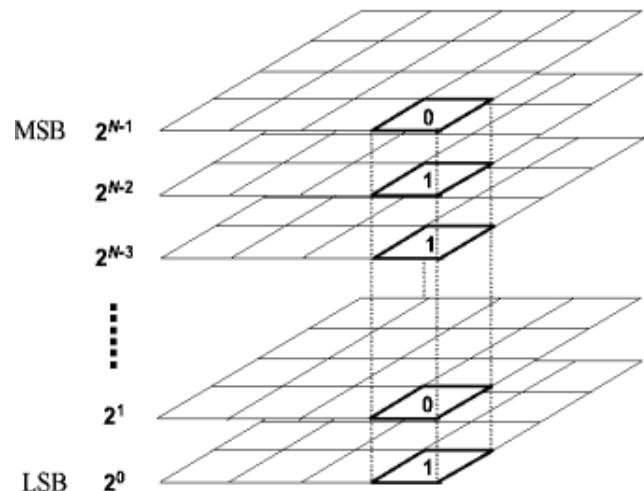
- c. Entropy Coding: si tratta di applicare Huffman sulla sequenza ottenuta;

## I bit-plane

Un'immagine con una profondità di colore di  $N$  bit può essere rappresentata da  $N$  piani di bit, ognuno dei quali può essere visto come una singola immagine binaria. In particolare si può indurre un ordine che va dal **Most Significant Bit (MSB)**, il bit più significativo, fino al **Less Significant Bit (LSB)**, il bit meno significativo.

Quindi, si definisce **bit-plane** di un'immagine digitale a  $N$  bit l'insieme di  $N$  immagini binarie, o piani, in cui l'immagine  $i$ -esima è costituita dai valori dell' $i$ -esimo bit della codifica scelta.

Verranno studiate due codifiche: **binario puro** e **gray code**.



## Codifica binario puro, caratteristiche e problematiche

La codifica **binario puro** consiste nel rappresentare i valori dei pixel nella loro normale codifica binaria. Questa semplice codifica gode delle seguenti caratteristiche:

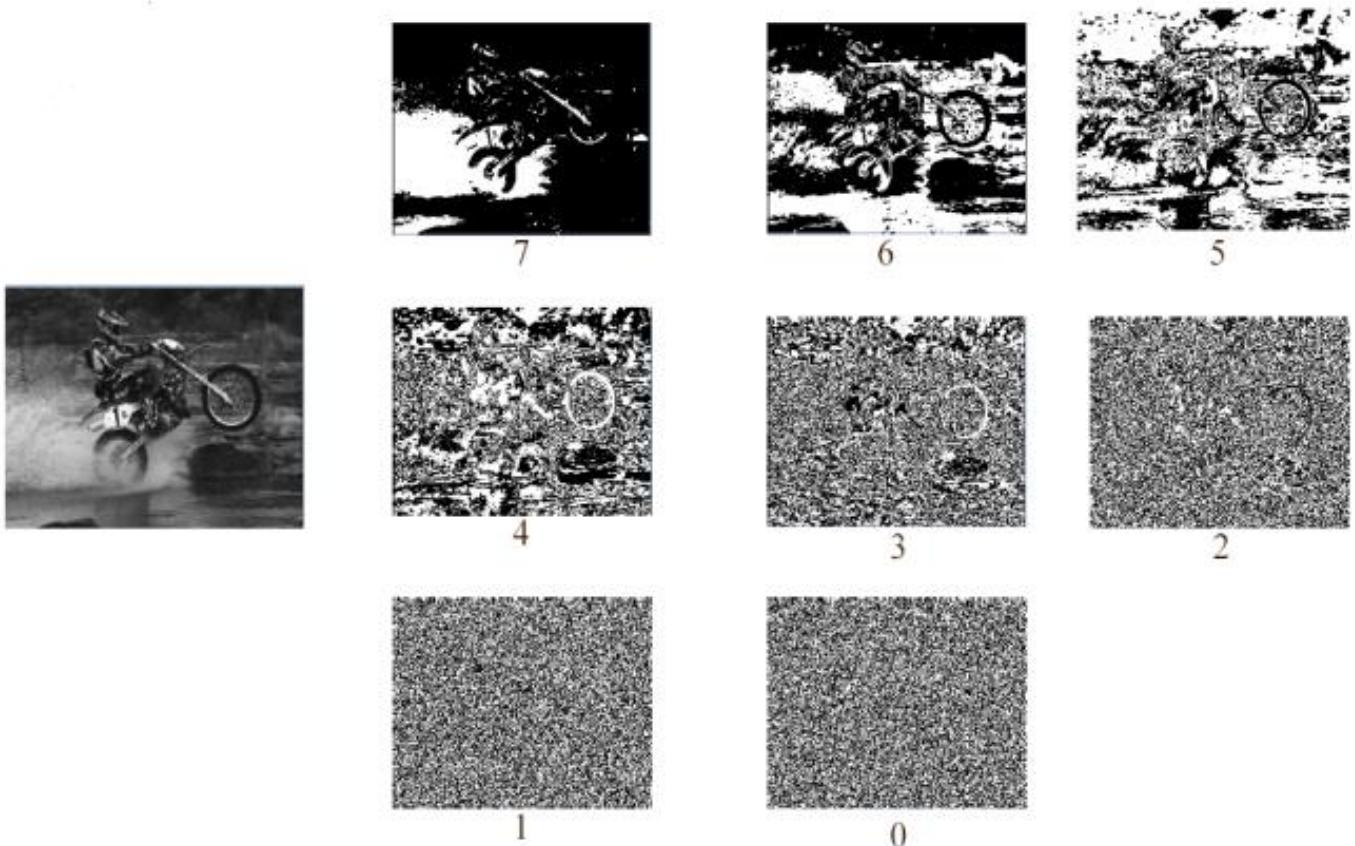
- I piani di bit più significativi contengono maggiori informazioni sulle geometrie dell'immagine, infatti dal punto di vista visuale tendono ad avvicinarsi all'immagine originale;

- I piani di bit meno significativi forniscono, invece, dettagli sempre più piccoli e di minore importanza, in tali piano sono più evidenti il rumore dell'immagine ed eventuali errori di acquisizione;

Queste caratteristiche ci permettono di:

- Attenuare il rumore in un'immagine eliminando i piani meno significativi. Per eliminazione si intende l'azzeramento del piano.
- Rimuovere specifici valori compresi fra un certo range:
  - Azzerando uno specifico piano  $x$  si elimineranno tutti i valori nel range  $2^{x+1} > \text{valori} \geq 2^x$

Esempio di bit-plane di un'immagine a 8bit:



La codifica binario puro soffre di una problematica per cui una piccola variazione può ripercuotersi su tutti i piani: per esempio nel passaggio da 127 (0111 1111) a 128 (1000 0000) la variazione di 0 e 1 si ripercuote su tutti gli 8 piani.

*Quindi nasce il Gray code*

Codifica Gray (Gray code) e confronto con il binario puro

La codifica gray a  $m$  bit  $g_{m-1} \dots g_1 g_0$  di un numero in binario puro  $a_{m-1} \dots a_1 a_0$  può essere calcolato con la formula:

$$g_i = a_i \oplus a_{i+1} \quad \text{con } 0 \leq i \leq m-2$$

$$g_{m-1} = a_{m-1}$$

Per codificare un  $i$ -esimo bit, eccetto per il più significativo che resta invariato, si effettua uno  $XOR$  fra l' $i$ -esimo bit e il successivo bit.

Il codice Gray gode della seguente proprietà:

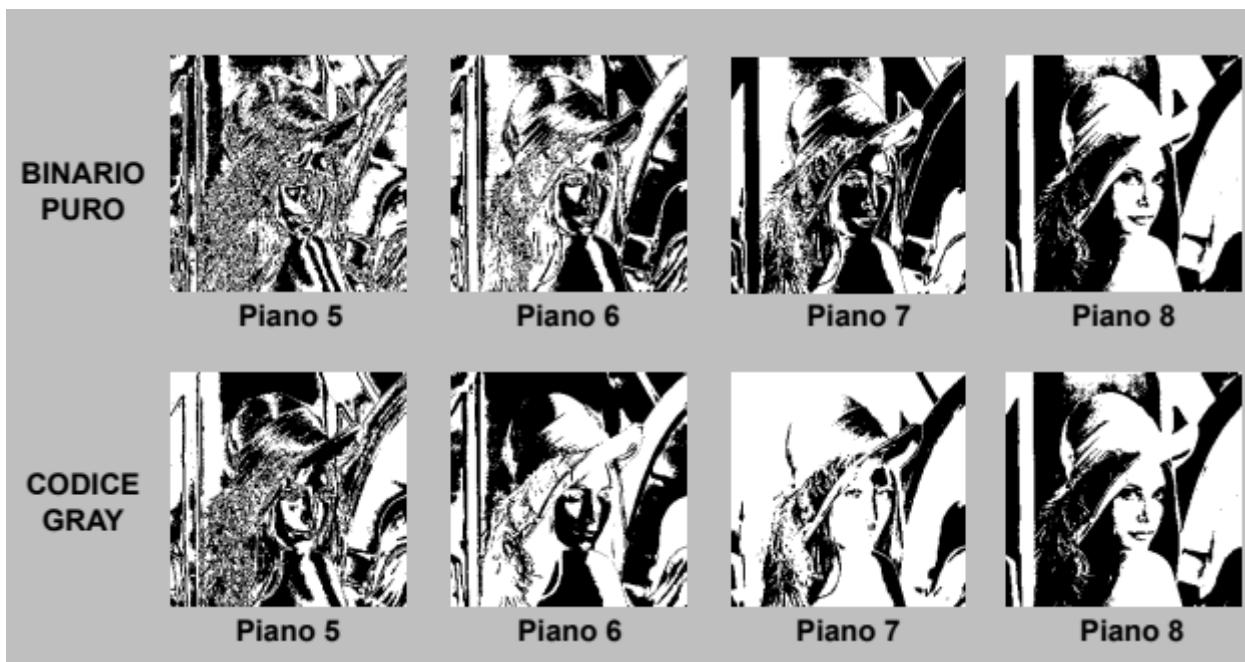
- La codifica di un dato numero differisce da quello del numero successivo per un solo bit;

I bit-plane di un'immagine codificata in gray code differiscono da quelli della stessa immagine codificata in binario puro:

- I bit-plane delle immagini in codice Gray risultano fra loro più coerenti se confrontati con i rispettivi in binario puro, questo perché all'aumentare dell'intensità del pixel di 1 varierà sempre solo un piano (ciò grazie alla proprietà di cui gode il gray code);

A	B	A <b>XOR</b> B
0	0	0
0	1	1
1	0	1
1	1	0

- Il numero di *transizioni bianco-nero* nel singolo piano risulta inferiori se si usa la codifica Gray;



Queste differenze comportano una *minore entropia*, ovvero una *maggior ridondanza*, che favoriscono l'impiego delle immagini codificate in gray per la compressione.

A causa del *diverso significato attribuito ai bit a seconda della codifica* alcune proprietà del binario puro non valgono per la codifica gray e viceversa:

- L'azzeramento di un piano di bit in Gray code comporta l'eliminazione di un *range di valori diversi* rispetto a quello del binario puro;
- Nonostante i dettagli minori e il rumore dell'immagine continuino a concentrarsi nei piani con bit meno significativi, al contrario del binario puro l'eliminazione di questi piano potrebbe introdurre artefatti;



# Differenti codifiche

- A questo punto si hanno due 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 Huffmann ottenuti sulla base di calcoli statistici preventivi e sono fornite dallo standard.



# Codifica coefficienti DC

- I coefficienti del blocco 8x8 messi in sequenza sono:

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

- Il coefficiente DC è il primo e vale -26.
- Assumendo che il coefficiente DC del blocco successivo sia -17, otterremo l'**evento**  
 $\Delta = -26 - (-17) = -9$



# Codifica coefficienti DC

- Nella tabella delle categorie, -9 sta nella posizione  $n=SSSS=4$ .

SSSS	$\Delta$
0	0
1	-1, 1
2	-3 -2, 2 3
3	-7 ... -4, 4 ... 7
4	-15 ... -8, 8 ... 15
5	-31 ... -16, 16 ... 31
6	-63 ... -32, 32 ... 63
7	-127 ... -64, 64 ... 127
8	-255 ... -128, 128 ... 255
9	-511 ... -256, 256 ... 511
10	-1023 ... -512, 512 ... 1023
11	-2047 ... -1024, 1024 ... 2047

SSSS = categoria

$\Delta$  = differenza tra due coefficienti DC  
(evento)

n.b.: le tabelle complete sono disponibili su teams



# Codifica coefficienti DC

- Il valore  $n=SSSS=4$  ha come codice base 101. Ed  $n$  è anche il numero di bit mancanti che occorre aggiungere.
- La corrispondenza tra il valore e il codice è fissata dalla tabella dei codici di Huffman che varia in base al fatto che stiamo trattando la luminanza o la crominanza.

SSSS	Codice base
0	010
1	011
2	100
3	00
4	101
5	110
6	1110
7	11110
8	111110
9	1111110
10	11111110
11	111111110

n.b.: le tabelle complete sono disponibili su teams



# Codifica coefficienti DC

A questo punto occorre completare il codice.

Per fare ciò si usa la seguente regola:

- Se  $\Delta > 0$  allora i bit da aggiungere sono gli n bit meno significativi del valore  $\Delta$  in binario.
- Se  $\Delta < 0$  allora i bit da aggiungere sono gli n bit meno significativi del valore in binario di  $\Delta$  (con complemento a due) ai quali occorre sottrarre il valore 1.
- $\Delta = 0$  allora anche SSSS è uguale a zero, pertanto non viene aggiunto alcun bit.

Per rappresentare l'opposto di un numero binario in complemento se ne invertono, o negano, i singoli bit: si applica cioè l'operazione logica NOT. Si aggiunge infine 1 al valore del numero trovato con questa operazione.



# Codifica coefficienti DC

- Nell'esempio considerato i quattro bit meno significativi del valore in binario di  $\Delta$  (-9) sono 0111; essendo  $\Delta < 0$  si sottrae il valore 1 e si ottengono i quattro bit 0110 che completano il codice base trovato in precedenza (101).
- Il codice completo del coefficiente DC è 1010110.



# Codifica coefficienti AC

-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

- Dalla sequenza si è eliminato il primo coefficiente e si passa alla codifica di tutti gli altri.
- Poiché i coefficienti quantizzati AC sono spessissimo nulli, si usa una trasformazione in skip-value. Cioè, data una sequenza di valori, si memorizza il numero degli zeri seguito dal primo valore non zero che si incontra.
- Esempio: si debba codificare  
0,0,0,0,11,0,0,0,3,0,0,0,0,0,0,0,0,12,17...  
memorizzo:  
(4,11),(3,3),(8,12),(0,17),...



# Nel nostro caso

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

## ■ Nel nostro caso sarebbe

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



# Codifica coefficienti AC

- La prima coppia del tipo (0,v) da codificare è (0,-3).
- Il coefficiente -3 ha come SSSS il valore 2.

SSSS	Coefficiente AC
1	-1, 1
2	-3 -2, 2 3
3	-7 ... -4, 4 ... 7
4	-15 ... -8, 8 ... 15
5	-31 ... -16, 16 ... 31
6	-63 ... -32, 32 ... 63
7	-127 ... -64, 64 ... 127
8	-255 ... -128, 128 ... 255
9	-511 ... -256, 256 ... 511
A	-1023 ... -512, 512 ... 1023

n.b.: le tabelle complete sono disponibili su teams



# Codifica coefficienti AC

- La classe dell'evento è espressa mediante una coppia del tipo (run, categoria).
- Nel nostro caso abbiamo (0,2)

(run, category)	Codice base	Lunghezza codice completo
(0, 0)	1010 (= EOB)	4
(0, 1)	00	3
(0, 2)	01	4
(0, 3)	100	6
....	....	....
(F, 0)	111111110111	12
....	....	....
(F, A)	1111111111111110	26

n.b.: le tabelle complete sono disponibili su teams



# Codifica coefficienti AC

Alla coppia (0, 2) corrisponde il codice base 01, tale codice sarà completato dall'aggiunta di un numero di bit fino a raggiungere il numero totale di bit che è riportato nell'ultima colonna della tabella.

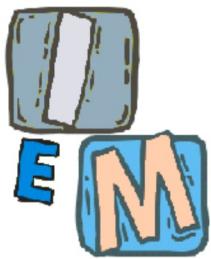
I bit che completano il codice base sono scelti con lo stesso criterio enunciato per la codifica dei coefficienti DC, in base al valore  $v$  del coefficiente AC ( $v > 0$ ,  $v < 0$ ,  $v = 0$ ).

- Se  $v > 0$  allora i bit da aggiungere sono i bit meno significativi del valore  $v$  in binario.
- Se  $v < 0$  allora i bit da aggiungere sono i bit meno significativi del valore in binario di  $v$  (con complemento a due) ai quali occorre sottrarre il valore 1.
- $v = 0$  allora anche SSSS è uguale a zero, pertanto non viene aggiunto alcun bit.



# Codifica coefficienti AC

- Nell'esempio considerato i due bit meno significativi del valore in binario del coefficiente AC v (-3) sono 01, essendo  $v < 0$  si sottrae il valore 1 e si ottengono i due bit 00 che completano il codice base trovato in precedenza (01).
- Il codice completo è quindi 0100.



# Sequenza finale del blocco 8x8

- 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 100001 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.