



NoiStudiamo

APPUNTI

Interazioni e Multimedia

Percezione, acquisizione ed elaborazione delle immagini digitali, colore, trasformata di Fourier e compressione

A CURA DI

Carola Aiello

Ottobre 2026

Indice degli argomenti

Gli argomenti principali degli appunti e la pagina in cui iniziano.

1. Il linguaggio visivo e la percezione umana	3
2. L'immagine digitale: rappresentazione matematica e storia	4
3. Immagini come matrici e operazioni fondamentali	5
4. Acquisizione di immagini digitali	11
5. La risoluzione	15
6. L'occhio umano e la percezione delle immagini	16
7. Campionamento e quantizzazione	21
8. Il colore: percezione, misura, modelli spazio colore	25
9. Operazioni puntuali sulle immagini digitali	32
10. Operatori locali e convoluzione	36
11. I bit plane	43
12. Fourier e miglioramento delle immagini	47
13. Compressione	56

Interazioni e Multimedia

Il Linguaggio Visivo e la Percezione Umana

Sin dalle sue origini, l'uomo ha utilizzato le immagini come principale forma di espressione e comunicazione. Prima ancora della scrittura, il linguaggio visivo ha rappresentato un modo immediato ed efficace per trasmettere informazioni, emozioni e conoscenza, confermando che "un'immagine vale più di mille parole."

Viviamo oggi immersi in un flusso costante di immagini. L'occhio umano interpreta l'immagine trasformando colore, movimento e profondità in vere e proprie dimensioni aggiuntive dell'esperienza. Tuttavia, la **percezione visiva** è un processo soggettivo, in quanto è il **cervello** a elaborare e dare significato a ciò che vediamo. Tale processo è influenzato da fattori biologici (come il daltonismo) e cognitivi.

Le Leggi della Gestalt e i Limiti Fisiologici

Il cervello organizza gli stimoli visivi seguendo principi definiti, noti come **Leggi della Gestalt** o **Leggi della Percezione Visiva**: gli elementi vengono raggruppati per **Vicinanza** (elementi vicini sono percepiti come unità), **Somiglianza** (elementi simili sono raggruppati), **Chiusura** (tendenza a percepire figure complete) e **Continuità/Buona Forma** (tendenza a seguire linee armoniose).

Il sistema visivo umano presenta anche limiti e peculiarità fisiologiche:

1. **Non Linearità della Luminosità:** Il sistema visivo umano non opera contemporaneamente su tutto il range delle intensità percepite, ma solo su una porzione di esso, non cresce linearmente con quella reale, ma segue una **funzione logaritmica**. Questo rende l'occhio più sensibile alle variazioni di luminosità nelle aree scure.
2. **Illusioni Ottiche:** Fenomeni come le **Bande di Mach** e il **Contrasto Simultaneo** dimostrano come la luminosità o il colore percepito siano alterati dal contesto o dallo sfondo circostante, provando che la percezione non dipende solo dall'intensità fisica della luce.

L'Immagine Digitale: Rappresentazione Matematica e Storia

Matematicamente, un'immagine è descritta come una funzione bidimensionale $f(x,y)$ che associa a ogni punto spaziale un valore di intensità luminosa, dato dal prodotto tra la luce incidente $i(x,y)$ e la riflettanza $r(x,y)$ dell'oggetto.

Per essere elaborata da un computer, l'immagine deve essere digitalizzata attraverso due processi di discretizzazione:

- **Campionamento (Sampling):** L'immagine viene suddivisa in punti discreti, i **pixel** (Picture Element), definendo la risoluzione spaziale.
- **Quantizzazione (Quantization):** A ogni pixel viene assegnato un valore finito di intensità, definendo la risoluzione del livello di grigio (o profondità di colore).

Il piano XY in cui stanno le coordinate dell'immagine è detto **dominio spaziale** e le variabili x,y sono dette variabili spaziali o coordinate spaziali.

Le tipologie di immagini dipendono dalla profondità di bit: le immagini in **bianco e nero** usano 1 bit per pixel (due valori: 0 o 1); le immagini a **scala di grigio** usano tipicamente 8 bit per pixel (256 livelli, da 0 a 255); le immagini **a colori** (modello RGB) utilizzano 8 bit per ciascuno dei tre canali (Rosso, Verde, Blu), per un totale di 24 bit per pixel.

Le immagini digitali possono essere rappresentate in due modi diversi:

- **Raster** (o bitmap), sono la forma più comune e rappresenta l'immagine come una griglia di pixel dove ogni pixel ha un valore preciso di colore o intensità. Il suo punto di forza è l'alto livello di **fotorealismo** e la capacità di rappresentare sfumature e dettagli complessi, uno dei suoi punti deboli è la vulnerabilità all'ingrandimento.
- **Vettoriale**, rappresenta l'immagine come insieme di forme geometriche e funzioni matematiche. È un ottimo vantaggio per quanto riguarda la **scalabilità**, un'immagine può essere ingrandita all'infinito senza perdere qualità ma non sono per nulla adatte a rappresentare scene realistiche e complesse ricche di dettagli.

La scelta tra i due formati dipende dall'uso finale dell'immagine.

In un'immagine raster a seconda della profondità si hanno diverse tipologie di rappresentazione. Le **immagini in bianco e nero** sono rappresentate da una matrice bidimensionale dove ogni posizione contiene un valore di 1 bit (0 per nero, 1 per bianco). Le **immagini a scala di grigio** sono rappresentate da una matrice bidimensionale dove ogni pixel contiene un valore di 8 bit permettendo $2^8=256$ livelli di intensità con valori compresi tra 0 (nero) e 255 (bianco). Le **immagini a colori (RGB)** sono rappresentate da una matrice tridimensionale (o tre matrici bidimensionali separate), l'immagine si basa sul modello RGB e sono necessari tre canali distinti. Ogni canale usa 8 bit di profondità di conseguenza l'immagine a colori utilizza 24 bit per pixel in totale. In una data posizione ci sarà una testa di valori (x, y, z) compresi tra 0 e 255 che definisce la combinazione rosso, verde e blu per quel pixel.

Le Immagini come Matrici e le Operazioni Fondamentali

Un'immagine digitale **raster** può essere concettualmente rappresentata come una **matrice** di conseguenza su un'immagine si possono effettuare tutte le operazioni definite per le matrici, sebbene non tutte abbiano un immediato "senso logico" visivo.

Prodotto tra Matrici e Immagini

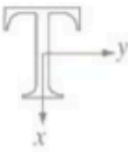
Nell'ambito dell'elaborazione delle immagini (**Image Processing**), si utilizza il **prodotto puntuale** tra due matrici, che consiste nel moltiplicare elemento per elemento i valori corrispondenti. Questa è una differenza fondamentale rispetto alla regola standard del prodotto tra matrici, che segue il principio "riga per colonna".

Operazioni Affini e Trasformazioni

Le operazioni affini sono una classe di trasformazioni applicate alle immagini che hanno l'obiettivo di determinare la **nuova posizione** di un pixel senza variare il valore di intensità.

Le principali operazioni includono:

- Identità
- Rotazione
- Traslazione
- Scaling (ridimensionamento)
- Shear (verticale e orizzontale)

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v \cos \theta - w \sin \theta$ $y = v \sin \theta + w \cos \theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

In generale le trasformazioni affini sono le trasformazioni più ampie che conservano i sottospazi affini. Esse vengono definite prevalentemente tramite una matrice di trasformazione 3×3 o attraverso un'equazione.

Forward Mapping

Il forward mapping è una tecnica per applicare le trasformazioni affini, data una matrice affine T :

1. Si scorre l'immagine di input, per ogni pixel di input (v, w) ;
2. Si calcola la nuova posizione di output (x, y) ;

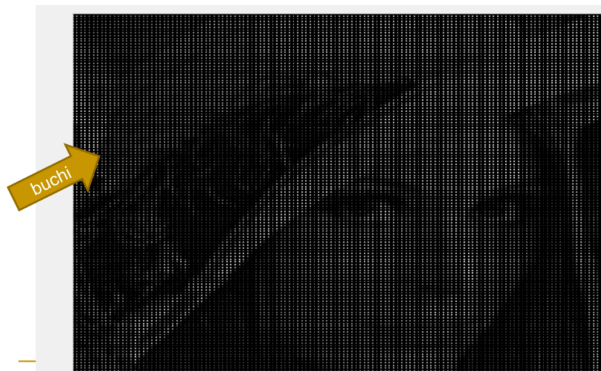
La relazione tra il pixel di input e quello di output è data dalla formula

$$\begin{pmatrix} x & y & 1 \end{pmatrix} = \begin{pmatrix} v & w & 1 \end{pmatrix} \times T$$

Uno dei **problemi principali** di questa tecnica è che può causare dei "buchi" (pixel non assegnati) nell'immagine di output.



Forward mapping (Rotazione)



Forward Mapping (Scaling)

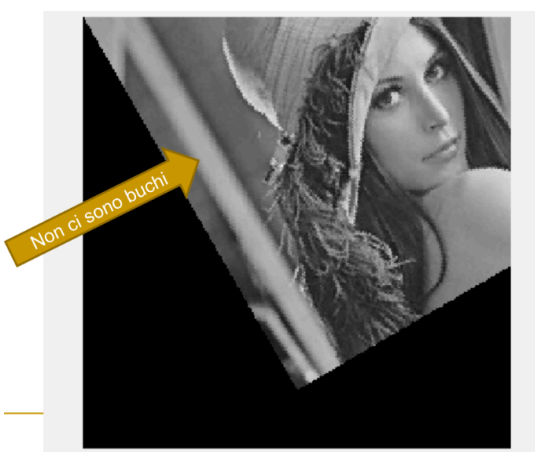
Reverse Mapping

L'inverse Mapping è la tecnica utilizzata per risolvere il problema dei "buchi", data la matrice affine T e T^{-1} la matrice inversa:

1. Si visitano le posizioni spaziali di ciascun pixel di output (x, y) ;
2. Per ogni posizione di output, si calcolano le coordinate corrispondenti (v, w) nell'immagine di input utilizzando la formula inversa;

La relazione inversa è data dalla formula:

$$(v \ w \ 1) = (x \ y \ 1) \times T^{-1}$$



Inverse Mapping (Rotazione)



Inverse Mapping (Scaling)

Le trasformazioni affini possono essere combinate semplicemente moltiplicando tra loro le rispettive matrici T

Interpolazione

Durante le trasformazioni di un'immagine possono esserci posizioni di pixel a cui non è stato assegnato un valore tramite le formule dirette. Per assegnare un valore a queste posizioni si applica un processo di **interpolazione**, ossia il processo che stima i dati ignoti a partire da dati reali noti. È importante ricordare che l'interpolazione **non migliora la qualità** dell'immagine come se riacquisisse i valori mancanti ma effettua una **stima dei valori ignoti**.

Zooming in (Ingrandimento)

L'interpolazione è necessaria anche nei processi di zooming (ingrandimento). Uno zooming 2x raddoppia le dimensioni dell'immagine da $m \times n$ a $2m \times 2n$ quadruplicando il numero totale di pixel. Dopo aver posizionato i valori dei pixel già noti è necessario stimare i valori nelle zone vuote. È possibile effettuare lo zooming in anche per le singole dimensioni, ad esempio triplicare il numero di righe e raddoppiare il numero di colonne.

Esistono diversi tipi di interpolazione:

- **Nearest Neighbor** (o Replication): assegna a ogni nuova posizione l'intensità del pixel più prossimo nell'immagine originale, **semplice** ma introduce possibili **distorsioni**;
- **Bilinear**: utilizza i 4 pixel più vicini per stimare l'intensità della nuova posizione. Il valore è determinato dall'equazione

$v(x, y) = ax + by + cxy + d$. Produce **risultati migliori** ma con un aumento della **complessità di calcolo**;

- **Bicubic**: utilizza i 16 pixel più vicini al punto da stimare, il valore è determinato da una sommatoria doppia
$$v(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j.$$
 Preserva meglio di dettagli rispetto alla bilineare, è la tecnica standard utilizzata in molti programmi di editing (ad esempio photoshop).

Il problema specifico nell'interpolazione è come gestire i **bordi dell'immagine**. Le soluzioni possibili sono:

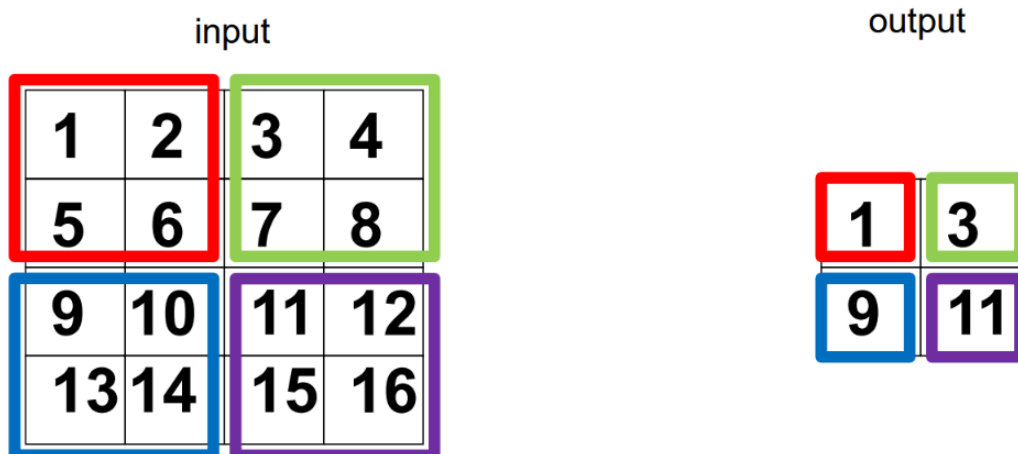
- Non fare nulla e lasciare i bordi vuoti;
- Interpolare usando i valori disponibili anche se in numero inferiore rispetto a quelli usati per i pixel interni;
- Considerare solo le zone centrali non calcolando i valori per i bordi;
- Dopo aver eseguito i calcoli, replicare le ultime righe e colonne.

Zooming out

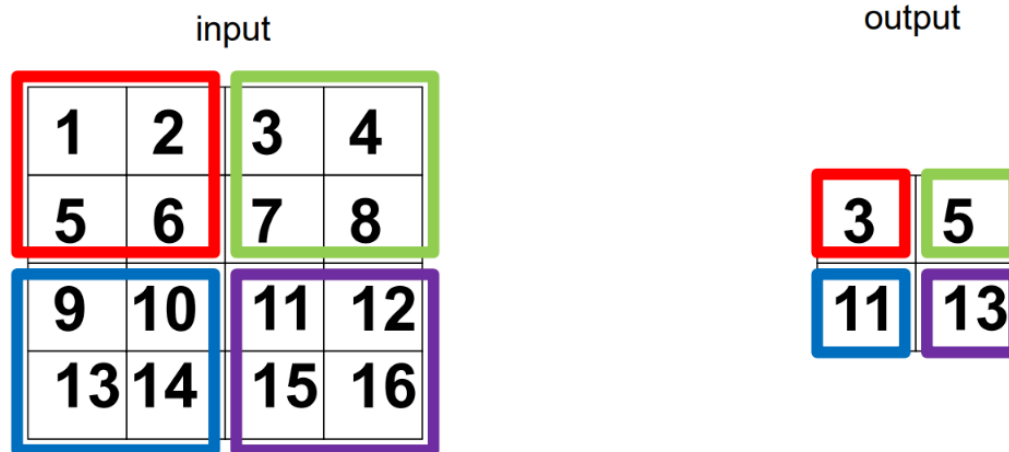
Se lo zooming è fatto con un fattore inferiore a 1, l'immagine risultante è più piccola dell'originale, questo processo è noto come **decimazione**. Ad esempio uno zooming out di 0,5 su un'immagine $m \times n$ risulterà un'immagine $m/2 \times n/2$.

Esistono 2 metodi per la decimazione:

1. **Campionamento**: scegliere un pixel ogni 4;



2. **Media:** calcolare il valore medio di 4 pixel per ottenere il valore del nuovo pixel.



Stima della Qualità di un Algoritmo

Per stimare la qualità di un algoritmo di elaborazione delle immagini o la somiglianza tra due immagini si utilizzano metriche come MSE e PSNR.

Mean Square Error (MSE)

Il parametro MSE (Errore Quadratico Medio) stima l'errore quadratico medio tra due immagini. **Minore** è l'indice MSE, **minore è la differenza** tra le due immagini. La formula per calcolare l'MSE tra un'immagine ottimale I e un'immagine da valutare I' (entrambe $M \times N$) è:

$$MSE = \frac{1}{MN} \sum_{x=1}^M \sum_{y=1}^N [I'(x, y) - I(x, y)]^2$$

Peak Signal to NoiseRatio (PSNR)

Il PSNR (Rapporto Segnale di Picco/Rumore) è un parametro usato per misurare la qualità di un'immagine compressa, dipende dalla differenza tra l'immagine codificata e quella originale. **Maggiore** è il valore del PSNR, **maggiore è la "somiglianza"** con l'immagine originale. Per calcolarlo è necessario avere l'immagine da valutare I' e la sua versione originale I . Si calcola a partire dall'MSE utilizzando una delle seguenti formule equivalenti dove S è il valore massimo possibile per un pixel (solitamente 255):

$$PSNR = 10 \log_{10} \left(\frac{S^2}{MSE} \right) \quad PSNR = 20 \log_{10} \left(\frac{S}{\sqrt{MSE}} \right)$$

Sebbene siano molto diffusi e semplici da calcolare, non sempre forniscono un risultato fedele a quello percepito dal sistema visivo umano (HVS). La sensibilità dell'HVS agli errori varia in base al contesto, e l'MSE non riesce a catturare adeguatamente questa differenza. Immagini con tipi di distorsione molto diversi possono avere lo stesso valore di MSE. Entrambe le metriche sono molto sensibili anche a "impercettibili" movimenti spaziali, come traslazioni, rotazioni o *flipping* di righe/colonne.

Andamento con la qualità JPEG e il rumore

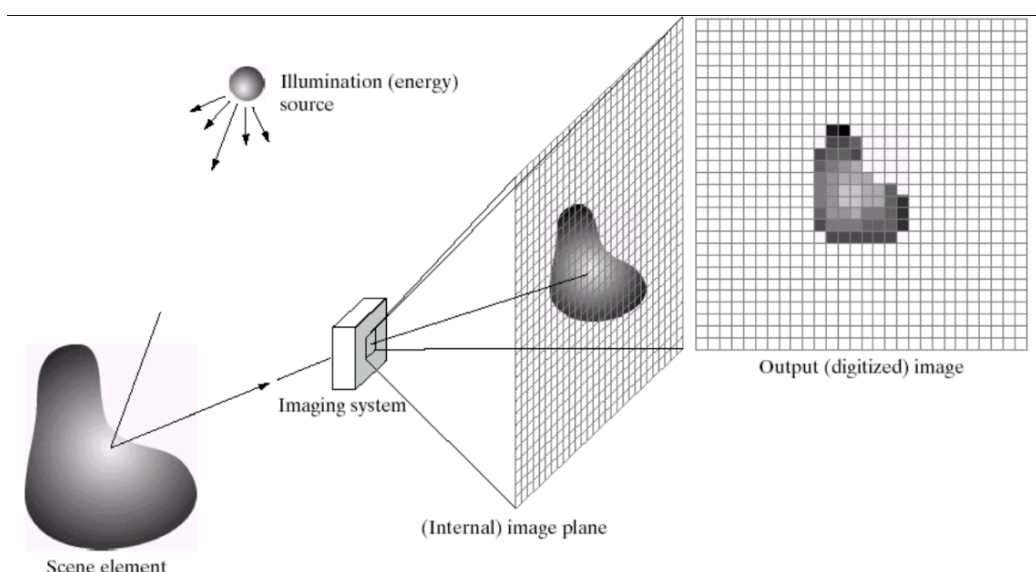
Per calcolare MSE e PSNR per immagini a colori (RGB), si utilizzano solitamente due approcci:

1. La semplice media dei valori MSE (o PSNR) calcolati sui tre canali (Rosso, Verde, Blu).
2. Una combinazione lineare che assegna un peso maggiore alla componente verde, data la sua importanza per la percezione visiva umana.

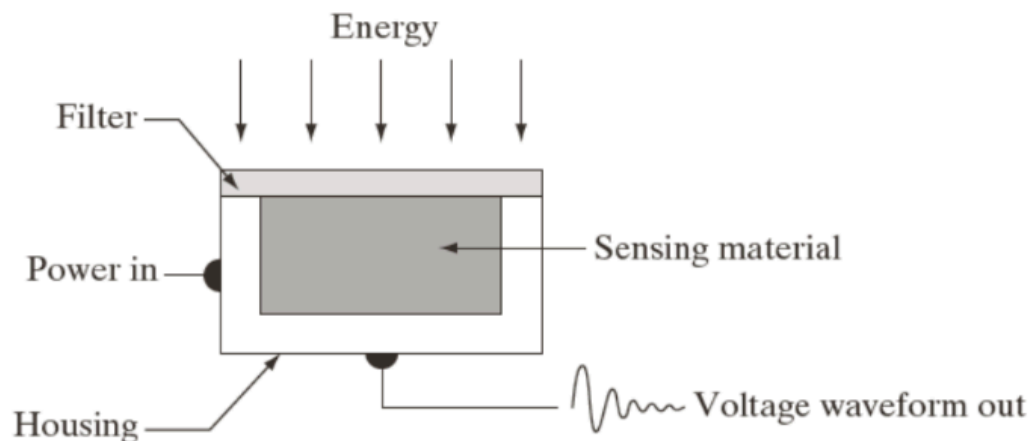
Acquisizione di immagini digitali

L'acquisizione delle immagini digitali inizia con l'interazione della luce con gli oggetti. Quando la luce colpisce un oggetto, una parte viene assorbita e una parte viene riflessa. La luce riflessa dà origine al colore che percepiamo, per creare un'immagine digitale questa luce riflessa deve essere catturata e successivamente elaborata da un sensore.

Il **processo di acquisizione di un'immagine** coinvolge diversi elementi: la sorgente di illuminazione, un elemento della scena, un sistema di imaging, un piano immagine interno ed infine l'immagine digitalizzata in uscita.



Il **sensore** è costituito da un materiale particolarmente sensibile alla luce. L'energia (luce) che lo colpisce viene trasformata in un impulso elettrico. Successivamente questo impulso elettrico viene digitalizzato. Elementi chiave del sensore includono un filtro, il materiale sensibile e l'alloggiamento, con input di alimentazione e un'uscita del segnale di tensione (Voltage waveform out).



A seconda dell'applicazione i sensori possono essere disposti in modi diversi:

- **Singolo sensore:** Gli *scanner* utilizzano un singolo sensore che viene spostato lungo la sorgente da digitalizzare;
- **Sensori in linea:** Macchine come quelle per la Tomografia Assiale Computerizzata (TAC) impiegano sensori disposti in un'unica linea.
- **Sensor in 2D array:** Nelle macchine fotografiche digitali i sensori sono disposti su una matrice. Non è necessario spostare il sensore per effettuare la scansione, i più diffusi di questo tipo sono i CCD (Charged Coupled Device).

Schema di Misurazione del CCD

Dopo che le cariche sono state acquisite da una matrice di celle, devono essere trasferite in una memoria digitale. La scansione avviene in C fasi, dove C è il numero di colonne della matrice. Ad ogni fase:

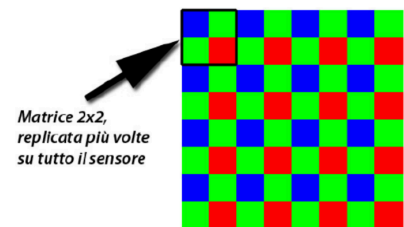
1. La prima colonna della matrice viene trasferita in memoria;
2. Contemporaneamente tutti gli elementi delle colonne successive vengono trasferiti dalla proprio colonna a quella precedente (trasferimento orizzontale verso la prima colonna).

CFA: Color Filter Array

Poiché ogni cella del sensore può memorizzare solo un colore per volta e non la terna completa, è necessario scegliere un modello ottimale di memorizzazione chiamato **CFA**. I due colori mancanti per completare la terna di colori di un pixel vengono ottenuti tramite **interpolazione** di pixel vicini tramite un processo chiamato **Color interpolation**. L'accuratezza del risultato finale dipende dalla sofisticatezza del metodo di interpolazione usato.

Il Bayer Pattern

Tra i vari modelli possibili di CFA, quello più utilizzato è il **bayer pattern**. È stato proposto da Bryce Bayer nel 1976 e viene utilizzato dal 1980 in tutti i dispositivi elettronici. Si basa su una matrice 2×2 replicata su tutto il sensore, rappresenta un rapporto di 1:2:1 per R:G:B con i pixel verdi disposti sulle diagonali. Questo schema privilegia le misurazioni nel canale verde, in quanto è il colore più importante per la percezione umana. Un'immagine in Bayer Pattern è conservata nel formato "**raw**".



Color Interpolation (Demosaicing)

La matrice di dati grezzi rilasciata dal sensore può essere visualizzata come un'immagine in scala di grigi o come un'immagine a "falsi colori" con un effetto a mosaico. Poiché è stata conservata solo una componente di colore per pixel, per ottenere un'immagine a colori è necessario applicare l'algoritmo di Color Interpolation. Questo algoritmo ricava i colori mancanti per ogni singola terna dai dati dei pixel vicini.

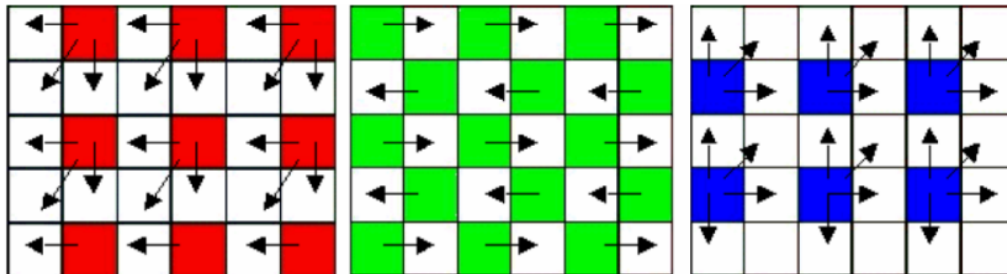
Viene chiamato anche algoritmo di "**Demosaicing**" perché elimina l'effetto a mosaico dell'immagine finale a colori.

Il processo completo di acquisizione di una fotocamera digitale prevede:

1. L'acquisizione del segnale dal sensore in *Bayer colorplan* (formato raw/CFA).
2. L'elaborazione dell'immagine tramite *Color Interpolation* (Col Int. Img Proc.) per ottenere 3 *color planes* (uno per R, G, B).
3. La formazione di un'immagine a colori (come un file TIFF).
4. La compressione in un formato come JPEG, se necessario

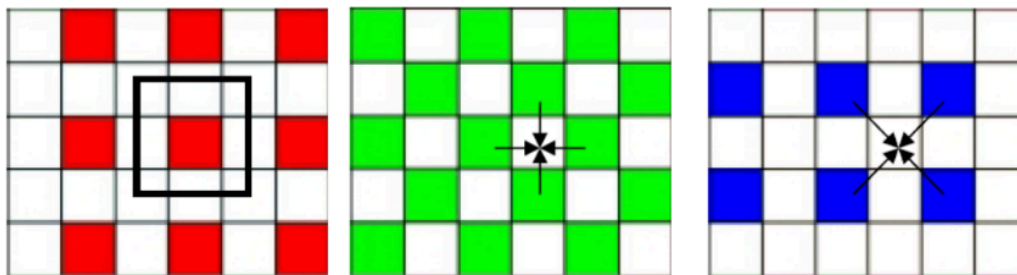
Le principali tecniche di interpolazione sono:

- **Replication (nearest-Neighbor interpolation):** Per ogni singolo pixel gli elementi mancanti della terna di colore vengono copiati dai pixel immediatamente vicini;

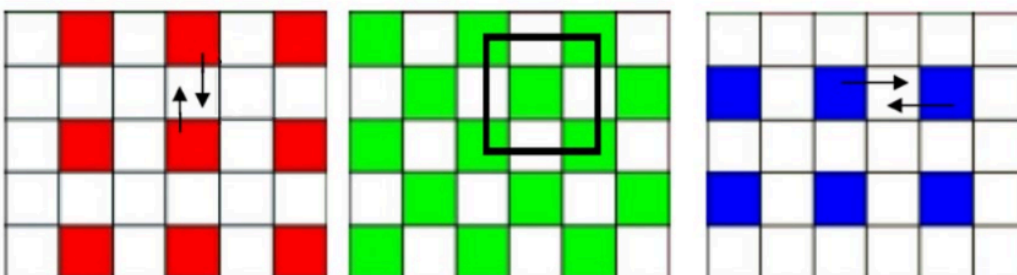


- **Bilinear interpolation:** Questa tecnica ricava i dati mancanti da un intorno selezionando i valori rilasciati dai pixel vicini, utilizzando 2 o 4 valori a seconda della posizione del colore mancante.

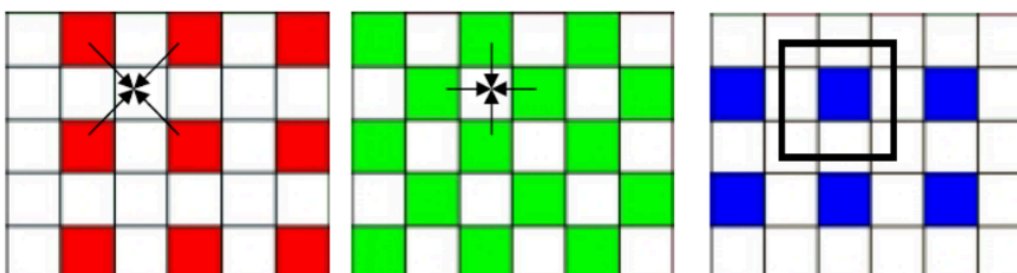
Red position:



Green position:



Blue position:



L'efficacia della tecnica di interpolazione influenza notevolmente la qualità dell'immagine finale.

La Risoluzione

Per definizione, la risoluzione è il **numero di pixel per unità di misura**. Questa può essere misurata in diverse unità, come **pixel al centimetro**, o più comunemente **dpi dots per inch**. In alternativa, la risoluzione può essere espressa come il **numero totale di pixel sull'intera immagine**, ad esempio "16 Megapixel".

È direttamente collegata al **grado di dettaglio che può essere rappresentato** in un'immagine. Lo schermo di un computer non mostra linee o disegni ma è composto da punti (pixel) sufficientemente piccoli da essere inferiori alla risoluzione percepita dall'occhio umano, l'osservatore ha l'impressione di vedere linee continue e disegni anziché una serie di punti allineati.

Il concetto di risoluzione è **relativo** e deve essere considerato a seconda della situazione specifica. Bisogna distinguere tra:

- Risoluzione dell'apparecchiatura di **ripresa**;
- Risoluzione dell'apparecchiatura di **resa**.

Ad esempio la risoluzione equivalente di una normale pellicola fotografica si aggira attorno ai 3.000-4.000 dpi.

Risoluzione apparecchiatura di Ripresa

Nelle apparecchiature di ripresa la risoluzione viene calcolata contando quanti **sensori** ci sono per unità lineare di misura.

- **Scanner**: la risoluzione varia fino a 6.000 dpi e oltre;
- **Fotocamera**: la risoluzione è data dal numero di sensori presenti sul circuito di ripresa e viene misurata in Megapixel.

Risoluzione apparecchiatura di Resa

Anche per le apparecchiature di resa la risoluzione si determina contando quanti **elementi** (o sensori) ci sono per unità lineare di misura.

- **Stampante**: la risoluzione può raggiungere i 3.000 dpi e oltre;
- **Schermo**: la risoluzione è data dal numero di elementi sullo schermo per unità di misura, tipicamente è 72 dpi

Per ottenere la **massima resa** da un'immagine, è fondamentale che questa, nata con una certa risoluzione, venga visualizzata con la stessa risoluzione.

Esistono numerose dimensioni standard di risoluzione:

Sigla	Definizione	Risoluzione	Rapporto
Quarter QVGA	QQVGA	160 x 120	4:3
CGA o Quarter VGA	CGA / QVGA	320 x 240	4:3
Half VGA	HVGA	640 x 240	8:3
Monochrome Display Adapter	MDA	720 x 350	
Enhanced Graphics Array	EGA	640 x 350	
Video Graphics Array	VGA	640 x 480	4:3
Super VGA	SVGA	800 x 600	4:3
Quad VGA	QVGA	1280 x 960	4:3
eXtended Graphics Array	XGA	1024 x 768	4:3
Super XGA	SXGA	1280 x 1024	5:4
Super XGA Plus	SXGA+	1400 x 1050	4:3
Ultra XGA	UXGA	1600 x 1200	4:3
Quad XGA	QXGA	2048 x 1536	4:3
Quad Ultra XGA	QUXGA	3200 x 2400	4:3
(*) Wide XGA	WXGA	1366 x 768	~16:9
(*) Wide XGA	WXGA	1280 x 800	16:10
(*) Wide XGA	WXGA	1280 x 720	16:9
Wide XGA Plus	WXGA+	1440 x 900	16:10
Wide Super XGA Plus	WSXGA+	1680 x 1050	16:10
Wide Ultra XGA	WUXGA	1920 x 1200	16:10

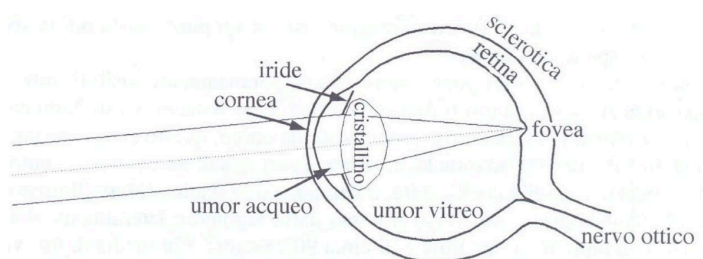
L'occhio Umano e la Percezione delle Immagini

Lo studio delle immagini (image processing) inizia necessariamente dall'analisi di come l'occhio umano percepisce ed elabora un'immagine. È fondamentale comprendere i **limiti della visione umana** per poterli applicare e sfruttare nell'ambito dell'immagine processing.

Anatomia dell'occhio e della retina

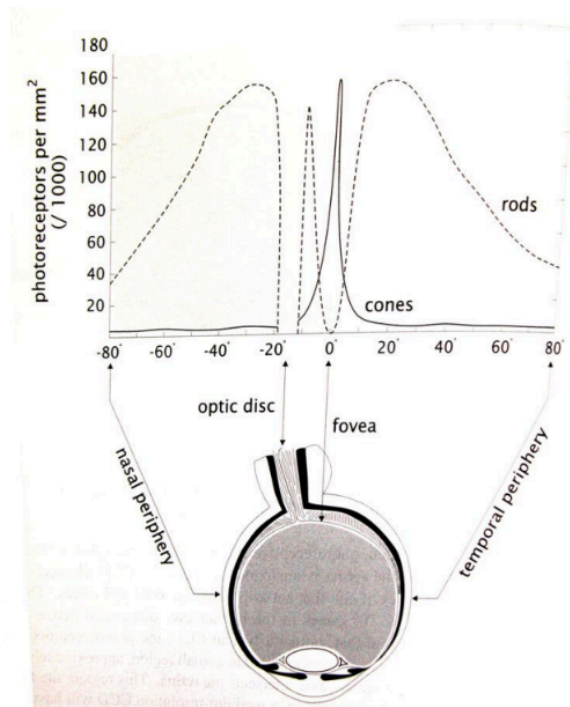
Tra le varie parti dell'occhio la parte interessata è la **retina**.

Una membrana che riveste la parte posteriore dell'occhio ed è costituita dai fotorecettori **coni e bastoncelli**.



I **coni** sono circa **6 o 7 milioni** e si trovano prevalentemente concentrati nella zona centrale della retina chiamata **fovea**. Sono fortemente sensibili al colore e ciascun cono è collegato ad un nervo ottico individuale. Sono responsabili della **vista fotopica** (o diurna), che è di tipo policromo (a colori).

I **bastoncelli** sono molto più numerosi, circa **75/150 milioni**, e sono distribuiti su tutta la retina. Sono poco sensibili al colore e sono collegati al nervo ottico a gruppi. Sono responsabili della **vista scotopica** (o notturna) che è di tipo monocromo (in bianco e nero).



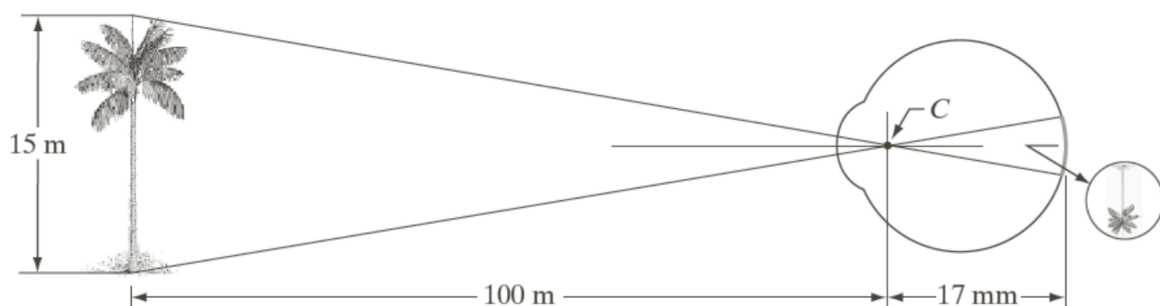
I grafici mostrano la disposizione dei sensori sulla retina, mostrano come i coni sono massimamente concentrati in corrispondenza della fovea, mentre i bastoncelli sono più densi nella periferia e completamente assenti in corrispondenza del disco ottico.

La **fovea** è una regione piccola, di circa **1,5 mm x 1,5 mm¹²**. Al suo interno, la popolazione di coni è di circa **150.000 elementi per mm¹²**, per un totale di circa **337.500 coni**. A titolo di confronto, un sensore CCD può contenere lo stesso

numero di recettori in un'area non inferiore a 5 mm x 5 mm.

Modello Pinhole

Per comprendere come si forma l'immagine nell'occhio, si ricorre al modello teorico del Pinhole (foro stenopeico).



Questo modello approssima l'occhio con una scatola. Su una parete interna viene posizionata una pellicola sensibile alla luce (equivalente al piano dei sensori o di formazione dell'immagine), mentre nella parete opposta si pratica un piccolissimo foro con uno spillo (pin hole). I raggi luminosi provenienti dall'oggetto (**scena**) passano attraverso il foro proiettando un'immagine capovolta sul piano dei sensori.

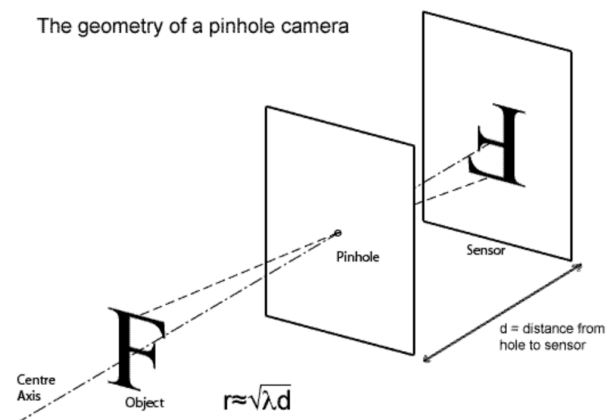
Pinhole ideale

Un pinhole ideale, con un foro puntiforme infinitesimo permetterebbe a un punto luminoso (**impulso**) di produrre un singolo punto sull'immagine, tuttavia un foro così piccolo non farebbe passare un numero sufficiente di fotoni per attivare i sensori e genererebbe fenomeni di diffrazione.

Pinhole reale

Un pinhole reale ha un foro con una precisa forma geometrica, a causa della sua estensione un punto luminoso (**impulso**) non produce un punto ma un'**area** "spalmando" la luce su una zona finita. Questa macchia generata è chiamata **Point Spread Function (PSF)** del sistema di acquisizione delle immagini.

In una telecamera pinhole, il raggio ottimale r del foro è proporzionale alla radice quadrata della distanza d tra il foro e il sensore, moltiplicata per la lunghezza d'onda λ della luce emessa, approssimativamente $r \approx \sqrt{\lambda d}$.



Principio di sovrapposizione

Quando due o più sorgenti luminose puntiformi si trovano vicine, l'effetto delle loro rispettive PSF si sovrappone, un concetto che avrà conseguenze importanti per il filtraggio delle immagini.

Le lenti sottili

I forellini (pinhole) sono inadeguati per i sistemi di acquisizione reali perché raccolgono troppe poche radiazioni per consentire misurazioni precise da parte dei sensori. Sin dal rinascimento si è diffuso l'uso delle **lenti di vetro** di forma lenticolare. Quando il diametro della lente è molto maggiore del suo massimo spessore, essa viene definita **lente sottile**. Tali lenti hanno

proprietà ottiche simili a quelle di un pinhole ma raccolgono molta più luce.

Una lente sottile è definita da alcune proprietà geometriche fondamentali:

- Raggi luminosi paralleli all'asse della lente vengono concentrati in un unico punto chiamato **fuoco**;
- Raggi che si dipartono dal fuoco vengono ri-trasmessi in modo parallelo nella direzione dell'asse della lente;
- Una lente sottile ha sempre **due fuochi equidistanti** da esse.

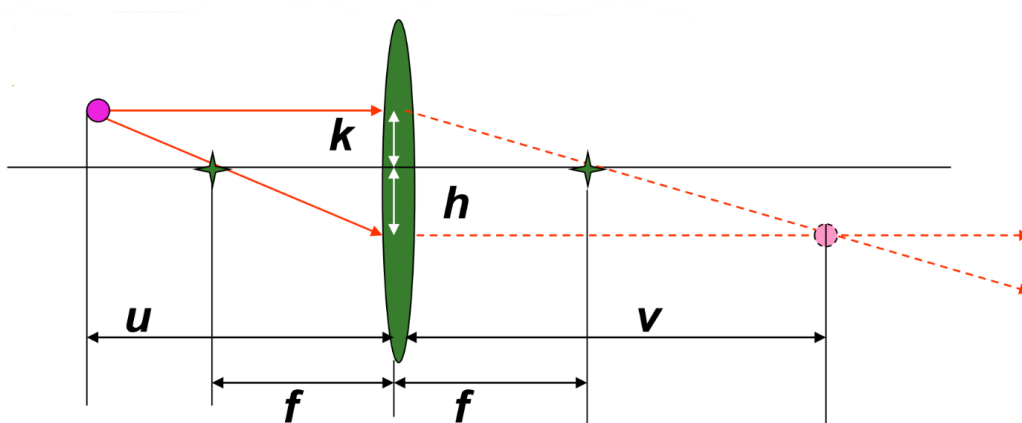
Per la formazione dell'immagine di un oggetto puntiforme:

- Un raggio emesso dall'oggetto parallelo all'asse ottico passa per il fuoco della lente.
- Un raggio emesso dall'oggetto che passa per l'altro fuoco viene rifratto dalla lente in modo parallelo all'asse ottico.
- Il punto in cui questi due raggi (e anche tutti gli altri, per una lente ideale) si intersecano è il **punto di formazione dell'immagine** dell'oggetto puntiforme.
- Se il piano dei sensori (o la retina) non è posizionato in corrispondenza di questo punto, si ottiene un'immagine **sfocata**.

Equazione della Lente Sottile

La relazione fondamentale che lega la distanza dell'oggetto u , la distanza dell'immagine v e la lunghezza focale f di una lente sottile è data da:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f}$$



Se si misura in metri la quantità 1/metro è definita come una **diottria**. La Variabilità della lunghezza focale si misura anch'essa in diottrie.

In una lente fissa la quantità f è costante. Se la distanza dell'oggetto, u , cresce per la relazione di prima v diminuisce, per questo la **messa a fuoco** richiede che il piano dei sensori possa essere avvicinato o allontanato dalla lente.

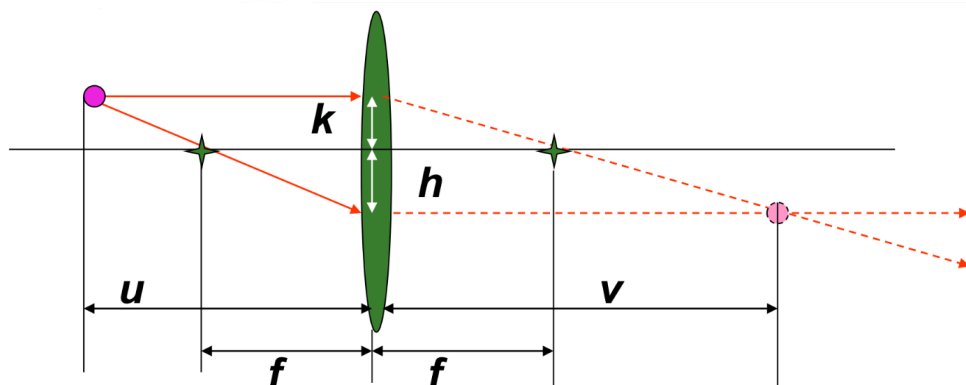
Nel caso dell'occhio umano, il piano dei sensori (retina) è fisso, l'unica soluzione è **regolare la lunghezza focale** f , questa azione è svolta dai muscoli che mantengono in tensione il cristallino.

Se due oggetti si trovano a distanze u_1 e u_2 molto maggiori della lunghezza focale f , le loro immagini v_1 e v_2 si formano su piani molto vicini (possono essere focalizzati approssimativamente nello stesso punto). Se invece u_1 e u_2 sono differenti ma comparabili (meno di 30 volte la distanza dalla lente) allora non possono essere focalizzati contemporaneamente, questo è noto come fenomeno della **profondità di campo**, più è grande f più è accentuato.

Magnificazione

La magnificazione m è il fattore che indica come le distanze vengono trasformate dalla lente sottile, ovvero la relazione tra la misura dell'oggetto k e la misura della sua immagine h . Poiché i triangoli formati sono simili si ha che:

$$m = \frac{h}{k} = \frac{v}{u}$$



Combinando questa relazione con quella dell'equazione della lente sottile si dimostra che:

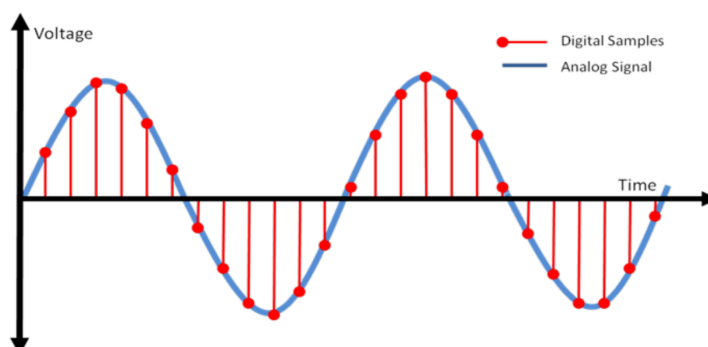
$$f = \frac{u \cdot m}{m+1}$$

Campionamento e quantizzazione

Il processo di conversione di una **segnale continuo** (analogico) in un **formato digitale** richiede due passaggi fondamentali: il campionamento e la quantizzazione.

Campionamento (sampling)

Dato un segnale continuo, è necessario selezionare un **numero finito di "campioni"** che siano rappresentativi del segnale stesso.



Per scegliere il giusto valore di campionamento si ricorre a un teorema fondamentale: il **teorema di Shannon**. Tale teorema si basa sulla misura del **Nyquist rate**.

Il Nyquist rate è il **doppio della più alta frequenza** presente in un segnale continuo e limitato. Si osserva un fenomeno che si svolge in un intervallo $a \dots b$.

- Se il fenomeno è (approssimativamente) **costante** durante l'intero intervallo, la più alta frequenza del segnale è 1 poiché il fenomeno si svolge in un unico ciclo.
- Altrimenti si divide l'intervallo in 2, 3, 4, ... parti e si controlla se il fenomeno si mantiene (approssimativamente) costante all'interno di ciascun intervallo.

Sia N il numero di parti in cui si ottiene una suddivisione tale che entro ciascun intervallino il fenomeno sia in pratica costante $2 \cdot N$ si definisce il Nyquist rate del fenomeno sull'intervallo osservato.

Il **teorema di Claude E. Shannon** (1949) afferma che se si raccolgono campioni con **frequenza più alta del Nyquist rate** il segnale può essere ricostruito **fedelmente** in ogni suo punto.

Errore di Campionamento e Aliasing

Un errore nel tasso di campionamento può stravolgere un segnale in due modi:

- **Campionamento troppo basso fa perdere dettagli e informazioni**, sebbene sia una perdita grave è spesso una necessità per conservare i dati in dimensioni maneggevoli;
- **Aliasing**, un campionamento troppo basso può far apparire nell'immagine dettagli **non presenti nell'originale**. Il segnale viene alterato e cambiato in qualcosa di "altro". È un fenomeno sottile ma poiché imprevedibile richiede attenzione.

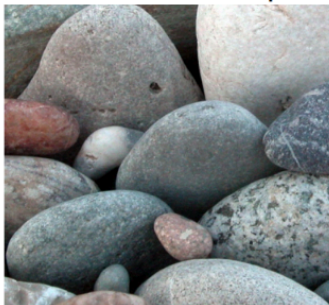
(guarda le slide "Lez 07" pag 15-20 per vedere gli esempi)

Se si campiona ad una frequenza inferiore a quella del Nyquist rate si perdono dettagli significativi e spesso si introducono nuovi dettagli che non sono presenti nella realtà. Questo fenomeno è detto **frequency aliasing** o semplicemente aliasing. Con l'aliasing le alte frequenze (i dettagli fini) sono "mascherate" da basse frequenze e trattate come tali nella fase di campionamento.



Aliasing

Campionamento
Un pixel ogni 256



Ricostruzione con
interpolazione
bicubica



Si perdono dettagli, graffi e disegni sulle rocce sono divenuti indistinguibili e sono apparsi **NUOVI** dettagli!

a) Ovvie scalettature sui bordi dei sassi.

b) Fori che non erano presenti nell'originale!

Nella realtà l'aliasing è **sempre presente**, anche in condizioni minime, perché viene introdotto quando si impone che il segnale sia limitato per essere campionato. Può essere ridotto applicando una **funzione di**

smussamento (blurring) sul segnale originario prima del campionamento detto **anti-aliasing**.

Moiré

L'effetto Moiré si verifica quando due **reticoli** o **pattern regolari** e ripetuti si sovrappongono in modo leggermente disallineato o inclinato.

1. Il Moiré come Fenomeno Fisico

- Si ottiene quando un oggetto semitrasparente con un motivo che si ripete viene sovrapposto a un altro oggetto simile. Il leggero movimento o la rotazione di uno dei due pattern genera delle variazioni su larga scala (figure d'interferenza).
- Queste figure d'interferenza formano bande scure (o colorate) che hanno un'ampiezza **molto più grande** del passo dei reticoli originali.

2. Il Moiré nel Campionamento Digitale

L'effetto Moiré nelle immagini digitali e nei video è una conseguenza diretta dell'**aliasing**, che si verifica quando un segnale viene campionato a una frequenza troppo bassa.

- Quando una fotocamera (o uno *scanner*) inquadra un soggetto con una **trama molto fine** e regolare (es. un tessuto a righe, una recinzione, una griglia), il pattern del soggetto interagisce con il **reticolo regolare dei sensori** (pixel) della fotocamera.
- Se la frequenza spaziale del soggetto (la densità del pattern) supera la risoluzione che il sensore può risolvere senza filtro, si verifica un **campionamento inadeguato**.
- Questo disallineamento e la sovrapposizione dei due pattern (quello del soggetto e quello dei pixel del sensore) creano nuove figure geometriche (le bande Moiré) che non erano presenti nell'originale.

In sostanza, come nell'aliasing generale, l'interazione del Moiré è un caso specifico in cui l'aliasing produce un riversamento delle **alte frequenze** (i dettagli fini della trama) su **basse frequenze** (le bande spurie e ampie).

Quantizzazione

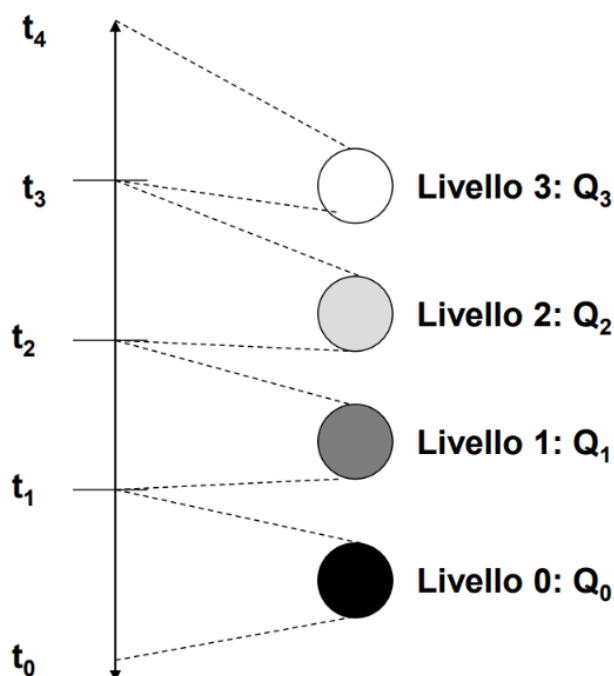
La quantizzazione è il processo necessario perché il valore in ogni singolo punto del segnale (campione) è un **numero reale**. I sensori (come i CCD) sono apparecchiature analogiche e forniscono misure di luminosità come numeri reali. La quantizzazione è utile per **arrotondare** tali valori e mantenerli in un certo range discreto.

Le misure sono sempre soggette a **errore** a causa di difetti nel sensore o perturbazioni termiche ("**rumore**"). Nei CCD anche ad obiettivo chiuso ci sono correnti parassite che introducono rumore, chiamate dark current.

Se i valori da quantizzare sono numeri reali nel range $[a, b]$ e si vuole quantizzare su n livelli:

- Si fissano $n + 1$ numeri soglia $(t_0, t_1, \dots, t_n)$ in $[a, b]$ tali che $t_0 = a < t_1 < t_2 < \dots < t_n = b$
- Il numero x in $[a, b]$ viene assegnato al livello di quantizzazione k se risulta $t_k \leq x < t_{k+1}$.

A ciascun livello si assegna un valore rappresentativo Q_i .



Fissato un numero di livelli di quantizzazione si pone il problema di come rappresentare in memoria tali livelli.

Quanti bit sono necessari per ricordare quale livello di luminosità si misura in un punto? Nell'esempio ne bastano $2 = \log(4)$

In generale se ci sono N livelli occorre rappresentare N etichette numeriche e avremo bisogno di un numero di bit pari a:

$$B = \log(N)$$

Esistono 2 tipi fondamentali di quantizzazione:

1. Quantizzazione Uniforme: i livelli di quantizzazione (le "distanze" tra livelli) sono costanti

- Dato un range in ingresso $0 \dots N - 1$ e un range in uscita $0 \dots K - 1$ ($K \leq N$), se il L è il livello di ingresso, il livello L' corrispondente dopo la quantizzazione è:

$$L' = \left\lfloor \frac{L \cdot K}{N} \right\rfloor$$

Esempio: portare $0 \dots 255$ in $0 \dots 7$ con quantizzazione uniforme. Il livello 10 diviene $(10 \cdot 8)/256 = 0$, il livello 20 diviene

$(20 \cdot 8)/256 = 0$, il livello 30 diviene $(30 \cdot 8)/256 = 0$, il livello 32 diviene $(32 \cdot 8)/256 = 1$ eccetera...

2. Quantizzazione Non Uniforme (spesso **logaritmica**): i livelli di quantizzazione hanno ampiezze diverse.

- La quantizzazione non uniforme e logaritmica è quella effettuata dagli scanner e dalle fotocamere digitali, poiché permette di assegnare più livelli nell'area dei toni scuri e meno livelli nell'area dei toni chiari.
- Il livello L' corrispondente dopo la riquantizzazione è dato da una funzione $L' = f(L, N, K)$, che definisce lo schema di riquantizzazione e può avere forme varie, la più comune è la

Quantizzazione logaritmica: descritta da $f(L, N, K) = \frac{\log_2(L) \cdot K}{\log_2(N)}$

(intendendo per log la parte intera).

Nel caso più comune $N = 256$, $\log(N) = 8$ o in generale $\log(N) = K$ per cui $f(L, N, K) = \log_2(L)$.

Il Colore: Percezione, Misura, Modelli Spazio-Colore

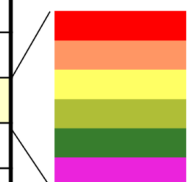
Una volta esaminato il funzionamento dell'occhio e come il cervello elabora le informazioni, ora è fondamentale comprendere la natura fisica della **luce**.

Nel 1666, Sir Isaac Newton dimostrò che facendo attraversare un prisma di vetro a un raggio di luce bianca, non si ottiene luce bianca, ma uno spettro continuo di colori che va dal violetto al rosso. Questo esperimento rivelò che la luce può essere scomposta in **onde luminose** di tipo differente.

I colori sono strettamente legati alle loro specifiche **lunghezze d'onda**. La luce che percepiamo come colore è solo una piccola porzione dello **spettro elettromagnetico**.

La **luce visibile** è costituita da tutte quelle lunghezze d'onda che possono essere percepite dall'occhio umano. Questo spettro oscilla tra il **violetto** e il **rosso**, con lunghezze d'onda comprese approssimativamente tra 400 nm e 700 nm.

Lung. in nanometri	Tipo radiazione
$10^{17} - 10^{13}$	Osc.elettriche
$10^{13} - 10^9$	Onde radio
$10^9 - 10^6$	Micro-onde
$10^6 - 10^3$	Infrarosso
$10^3 - 10^2$	Visibile
$10^2 - 10$	Ultravioletto
$10 - 10^{-3}$	Raggi X
$10^{-3} - 10^{-7}$	Raggi gamma e cosmici



Per comodità, lo spettro del visibile è convenzionalmente diviso in 6 regioni: violetto, blu, verde, giallo, arancio e rosso. È importante notare che queste bande di colore non hanno la stessa ampiezza e degradano in quelle limitrofe.

La percezione del colore di un oggetto da parte dell'occhio umano è dovuta alla luce che l'oggetto **riflette**.

- Se un oggetto riflette tutte le lunghezze d'onda luminose viene percepito come **bianco**;
- Se un oggetto riflette lunghezze d'onda tra 500 nm e 570 nm e assorbe le altre viene percepito come **verde**.

Il colore, a differenza di misure fisiche come la lunghezza o il peso, non ha una scala fisica per la sua misurazione. È una questione di **percezione** o **interpretazione soggettiva**. Il riconoscimento del colore da parte dell'uomo si basa sull'interazione tra:

1. **Radianza**: quantità di luce emessa dalla sorgente luminosa;
2. **Luminanza**: la misura dell'energia percepita dall'utente;
3. **Brillantezza**: valore soggettivo che indica la sensazione di colore.

Nella retina sono presenti 3 tipi di **coni** fondamentali per la visione a colori:

- **Tipo S** (short): sensibili alle lunghezze d'onda corte (colori bluastri);
- **Tipo M** (middle): sensibili alle lunghezze d'onda medie (colori verdastr);
- **Tipo L** (long): sensibili alle lunghezze d'onda lunghe (colori rossastri).

La **Teoria del Tristimolo**, proposta da Young nel 1802, ipotizza che tutti i colori possono essere ottenuti "mescolando" tre colori fondamentali in proporzioni diverse. Dimosteremo che questa ipotesi è FALSA se non in prima approssimazione.

Le differenze di colore percepite possono dipendere da diversi fattori:

- **Sorgenti luminose diverse**: luce solare, lampade a incandescenza, neon, ecc.
- **Osservatore**: sensibilità, vista;
- **Dimensioni**: l'effetto aerea fa apparire i colori che coprono vaste superfici più luminosi e vivaci;
- **Sfondo**: l'effetto contrasto fa apparire i colori più cupi se lo sfondo è scuro;
- **Direzione**: caratteristiche direzionali di alcune vernici che richiedono la costanza degli angoli di osservazione e illuminazione.

Sintesi Additiva (RGB)

Illuminando una superficie bianca con una o più luci monocromatiche si ottiene un colore per Sintesi Additiva.

Qualsiasi luce di spettro complesso può essere ottenuta come “somma” di tre luci monocromatiche che agiscono simultaneamente.

La **CIE** (Commission Internationale de l'Éclairage) ha scelto tre radiazioni primarie con lunghezze d'onda il più possibile distanziate nello spettro:

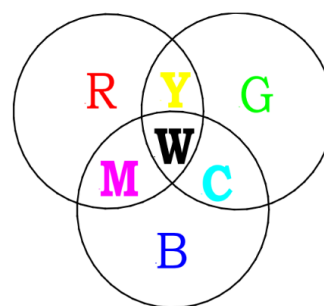
Rosso 700.0 nm, **Verde** 546.1 nm, **Blu** 435.8 nm.

La sovrapposizione di Rosso, Verde e Blu produce il **Bianco**.

La sovrapposizione di due luci produce i colori secondari **Giallo** ($Y, R + G$),

Magenta ($M, R + B$) e **Ciano** ($C, B + G$). Su questo principio si basano monitor e schermi televisivi.

I colori **primari** in sintesi additiva sono rosso, verde e blu (RGB), combinandoli a due a due si ottengono i colori **secondari** (magenta, giallo e ciano). I colori complementari sono quelli la cui somma produce il bianco, ad esempio il giallo è complementare del blu.

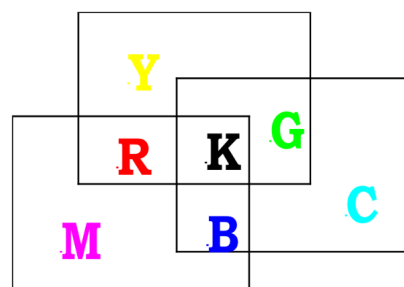


Sintesi Sottrattiva (CMY/CMYK)

La riproduzione del colore con pigmenti o inchiostri si basa sulla Sintesi Sottrattiva.

Un filtro ottico colorato è un mezzo a facce piane e parallele che attraversato dalla luce bianca ne assorbe selettivamente una porzione. Il filtro assume il colore prodotto dalla radiazione complementare di quella assorbita. Sovrapponendo tre filtri di colore **Giallo** (Y), **Magenta** (M), e **Ciano** (C) su un visore si ottiene il **Nero** (K) poiché tutta la radiazione visibile viene assorbita. Sovrapponendo due filtri si ottiene il colore corrispondente alla componente luminosa non assorbita da nessuno dei due: $Y + M = R$, $Y + C = G$, $M + C = B$.

Questo modello è usato nei sistemi passivi nei quali, la luce viene assorbita selettivamente la luce alle diverse lunghezze d'onda, riflettendo solo alcune che comporranno i colori desiderati.



Spazi di colore

In realtà nel mondo naturale raramente percepiamo colori puri, ma piuttosto vediamo **miscele di radiazione** in ogni lunghezza d'onda. Il nostro cervello non funziona come uno spettrometro: sebbene sia incredibilmente abile, si basa su un'elaborazione complessa delle informazioni ricevute dai tre tipi di coni. Questo può portare al fenomeno del **metamerismo**: spettri di luce differenti possono produrre sensazioni cromatiche percepite come identiche. Nonostante i cambiamenti nella luce che illumina una superficie manteniamo una percezione costante del colore di tale superficie.

Ogni lunghezza d'onda trasporta differenti quantità di energia e l'energia totale della radiazione è la somma di tutti i contributi energetici provenienti dalle diverse lunghezze d'onda. Lo **spettro di un illuminante** è il diagramma dei contributi di energia che esso apporta per ciascuna differente lunghezza d'onda.

Spazi Colore Colorimetrici CIE (XYZ, xyY)

La colorimetria si basa sulla sensibilità spettrale dell'occhio umano per definire i **valori del tristimolo X, Y, Z**:

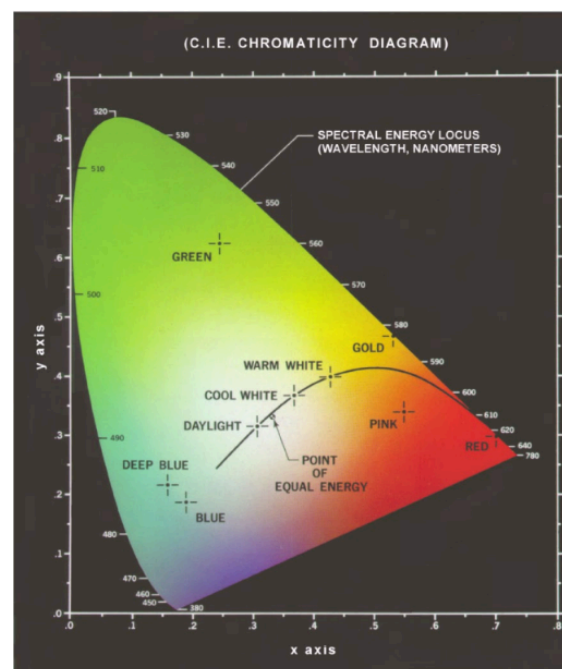
$$X = \int_{380}^{780} L_{e,\lambda} \bar{x}(\lambda) d\lambda \quad Y = \int_{380}^{780} L_{e,\lambda} \bar{y}(\lambda) d\lambda = \int_{380}^{780} L_{e,\lambda} V(\lambda) d\lambda \quad Z = \int_{380}^{780} L_{e,\lambda} \bar{z}(\lambda) d\lambda$$

Queste formule definiscono lo spazio colorimetrico **CIE 1931**.

Il principale difetto del sistema CIE XYZ è che **non è dotato di uniformità percettiva**, ossia benchè le distanze tra due colori sono uguali saranno percepite come differenti.

Il **Diagramma Cromatico CIE** è la proiezione dello spazio CIE 1931 su un piano xy.

- le coordinate x e y si calcolano come $x = X/(X + Y + Z)$, $y = Y/(X + Y + Z)$. Il valore z si calcola come $z = 1 - (x + y)$.
- Tutti i colori delle lunghezze d'onda visibili sono disposti lungo i bordi.
- Il punto di uguale energia è il **bianco**.



- Qualsiasi colore lungo il bordo non ha il bianco quindi è **puro**.
- Unendo due colori con una linea, tutti i colori lungo quella linea sono ottenibili mescolandoli.

Un **illuminante** è energia con distribuzione spettrale di energia relativa definita nel capo di lunghezza d'onda capace di influenzare la visione del colore degli oggetti. La CIE definisce diversi **illuminanti Standard** colorimetrici, come A, B, C, e D65, ciascuno con una distribuzione spettrale di energia definita. L'illuminante classico per la maggior parte degli usi è D65.

Spazi uniformi CIE (**L*a*b*** e **L*u*v***)

Per ovviare al problema della non uniformità percettiva del CIE X,Y,Z, la CIE ha standardizzato nel 1976 due spazi di colore percettivamente uniformi: **CIE L*u*v*** e **CIE L*a*b*** (CIELAB).

Le seguenti formule permettono di passare dallo spazio colore CIE XYZ allo spazio colore CIELAB e viceversa:

$$L^* = 116 \left(\frac{Y}{Y_n} \right)^{\frac{1}{3}} - 16 \quad a^* = 500 \left[\left(\frac{X}{X_n} \right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n} \right)^{\frac{1}{3}} \right] \quad b^* = 200 \left[\left(\frac{Y}{Y_n} \right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n} \right)^{\frac{1}{3}} \right]$$

L* rappresenta la **luminanza** mentre a* e b* rappresentano la **crominanza**. L'asse a* va dal verde (valore negativo) al rosso (valore positivo) e l'asse b* va dal blu (valore negativo) al giallo (valore positivo). L*, a* e b* sono trasformazioni dei tre valori di tristimolo X, Y, e Z dello spazio CIE XYZ.

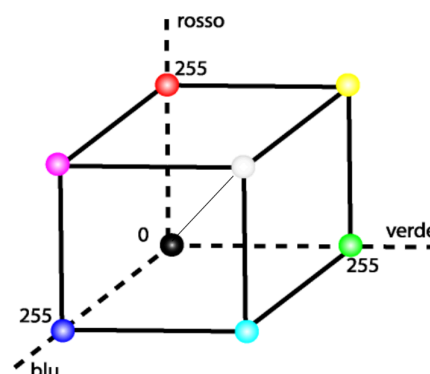
Nello spazio CIELAB, le differenze di colore sono definite come la distanza (ΔE^*_{ab}) tra due punti nello spazio L*a*b* attraverso la formula:

$$\Delta E^*_{ab} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2}$$

Uguali differenze corrispondono a **uguali differenze di percezione**, il che è il pregio principale di questo sistema. Esiste anche lo spazio L*C*h* che deriva dal CIELAB, in cui L* è la **Luminosità**, C* è la **Croma** (saturazione, la distanza dal centro del piano a*b*) e h è l'**angolo di tinta** (hue, l'angolo sul piano a*b*).

Modello RGB

Nel modello RGB ogni colore si ottiene mescolando i tre colori fondamentali. Se ogni componente è intesa come una coordinata cartesiana, il modello RGB può essere descritto geometricamente da un **cubo**. Dove i contributi Rosso, Verde e Blu sono assunti indipendenti e rappresentano gli assi perpendicolari.



L'origine (0,0,0) è il **nero** e il vertice opposto (ad esempio 255, 255, 255 in una codifica a 8 bit per canale) è il **bianco**. La retta che congiunge il nero e il bianco è la **retta dei grigi**.

Spazi Luminanza-Crominanza (YUV-YC_bC_r)

Questi spazi hanno una componente per la **luminosità** (luminanza) e due componenti legati alla **crominanza** (colore): la luminanza fornisce una versione in scala di grigi, mentre la crominanza aggiunge le informazioni di colori. Sono importanti nella compressione, poiché l'occhio umano è più sensibile alla luminanza che ai colori permettono di registrare la luminanza con più bit e risparmiarne sulle crominanze.

Lo spazio **YUV** è utilizzato per la codifica di immagini video analogiche, tenendo separate luminanza (Y) e crominanza (U , V).

- La luminanza può essere ottenuta mediante una combinazione lineare delle intensità luminose dei canali rosso, verde e blu di RGB. Un'approssimazione abbastanza fedele della luminanza Y si ottiene attraverso la somma: $Y = 0.299R + 0.587G + 0.114B$;
- Il termine crominanza U e V è definito come la differenza tra il bianco di riferimento alla stella luminanza opportunamente pesato:
$$U = 0.564(B - Y) \rightarrow U = -0.169r - 0.331G + 0.5B$$
$$V = 0.713(R - Y) \rightarrow V = +0.5R - 0.418G - 0.081B$$

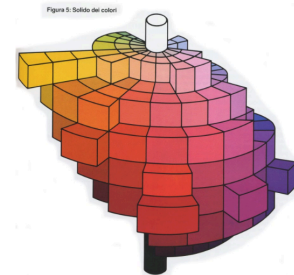
Gli spazi YC_bC_r sono le controparti digitali ottenute normalizzando e quantizzando i canali YUV. In questo spazio, Y è la luminanza e C_b e C_r rappresentano la crominanza del blu e del rosso. Questo spazio è largamente utilizzato nella compressione, noi lo ritroveremo nel formato JPEG.

Spazio HSV/HSI (Hue-Saturation-Value/Intensity)

Questo modello si basa sul “Modello del Pittore”, nonostante l'estrema complessità della luce reale, è stato dimostrato che ogni spettro ha un **metamero** (uno spettro equivalente in termini di colore percepito) che può essere semplificato e descritto in termini di tre caratteristiche fondamentali, che sono alla base del **Modello del Pittore**. Questo modello utilizza parametri con una perfetta e intuitiva interpretazione nelle nostre percezioni:

- **Tinta H** (Hue): corrisponde alla lunghezza d'onda in cui si ha il **picco** di energia (e_{picco}) ed è responsabile del “colore percepito”;
- **Luminosità/Valore V o I** (Value): corrisponde all'energia **media** (e_{media}) della radiazione ed è considerata una misura della “luminosità” di una radiazione. Esso da un contributo di “bianco” al colore percepito.
- **Saturazione S** (Saturation): indica quanto il colore è **puro** o vivace ed è dato dal rapporto $(e_{picco} - e_{media}) / (e_{picco} + e_{media})$. Minore è il contenuto di luce bianca (quindi maggiore è il valore del rapporto), maggiore sarà la purezza o saturazione del colore.

Lo spazio HSV/HSI è concettualmente rappresentato come una **piramide esagonale** (o cono) ottenuta attaccando i triangolo di diverse HUE lungo un asse centrale (la linea dei grigi, da nero a bianco).



Colori e Memoria

Lo schema più comune per rappresentare i colori in RAM è il **True Color** (o Colore Reale): 8 bit per il Rosso + 8 bit per il Verde + 8 bit per il Blu per un totale di 24 bit, circa 16 milioni di colori.

In termini di spazio risulta essere **troppo costoso** e non è il metodo in cui i colori vengono conservati in memoria di massa e compressi in formati JPEG, GIF o altro. Dato che le immagini “naturali” raramente hanno un colore differente per ogni pixel si adotta spesso la modalità a **Colori Indicizzati** (o a **Palette**, o a **Look-Up-Table/LUT**).

La **Palette** è una tabella che associa delle “etichette” (o indici con le corrispondenti componenti RGB. Per ogni pixel nell'immagine, invece di memorizzare il suo valore RGB completo (i 24 bit), si memorizza un numero che indica la sua **posizione (indice)** all'interno della Palette.

I software (come GIF) usano spesso una palette di 256 colori:

- Se un'immagine True Color ha meno di 256 colori alcuni di essi vengono replicati;
- Se un'immagine True Color ha più di 256 colori, questi vengono ridotti scegliendo 256 rappresentanti che garantiscano una buona qualità visiva (esistono algoritmi specifici per questo).
- La palette può essere **custom** (adattata all'immagine) o **standard** (come MAC, Windows Web_Safe ecc.)

Un'ulteriore tecnica di ottimizzazione è il **re-indexing**. Questa operazione cambia la posizione dei colori all'interno della palette (e di conseguenza gli indici associati) per minimizzare l'entropia della matrice degli indici. L'idea generale è che se ai **colori di pixel adiacenti** associo **indici numericamente vicini** la differenza tra tali indici diventa più piccola rendendo così la codifica differenziale più efficiente e garantendo un **miglior rapporto di compressione**. Sebbene trovare la disposizione ottimale sia un problema computazionalmente difficile (NP-hard): per una palette con M colori bisogna cercare tra M! possibili ordinamenti. Pertanto algoritmi di reindexing si basano sulla ricerca che si avvicinano a quella ottima.

Operazioni Puntuali sulle Immagini Digitali

Il punto di partenza per l'elaborazione di un'immagine è considerare i suoi pixel come una vera e propria **popolazione statistica**. Su questa popolazione è possibile calcolare tutte le quantità statistiche descrittive standard come la Media, la Mediana, la Varianza, la deviazione standard, i quartili e percentili.

Particolarmente importante è la conoscenza della distribuzione delle frequenze dei toni di grigio, nota come **Istogramma**.

Istogramma

L'istogramma è una rappresentazione che per ogni livello di grigio k (un tono di colore), riporta il numero di pixel che possiedono quel valore.

Per un'immagine di $I[m, n]$ la funzione dell'istogramma è data da:

$$H(k) = \text{numero di pixel di valore } k$$

La somma di tutti gli H è esattamente $m \times n$. L'istogramma è uno strumento fondamentale che permette di comprendere immediatamente le caratteristiche di un'immagine.

L'istogramma non tiene conto della **distribuzione spaziale dei pixel**, infatti immagini diverse potrebbero avere istogrammi simili.

- Un'**immagine chiara** avrà un istogramma più denso concentrato verso **destra** (toni di grigio più alti vicino al bianco);
- Un'**immagine scura** avrà un istogramma più denso concentrato verso **sinistra** (toni di grigio più bassi, più vicini al nero).

L'espansione del contrasto è un'operazione che serve ad **aumentare la dinamica** di un'immagine il cui istogramma è concentrato su un intervallo limitato dei valori possibili:

- **Funzionamento:** Si ottiene spostando, tramite appositi algoritmi, i valori di un "bin" (barra o livello) dell'istogramma verso un "bin" non utilizzato.
- **Risultato:** L'istogramma dopo lo **stretching** apparirà a forma di "pettine". Ciò è fatto per mettere in risalto che i bin mancanti sono stati distribuiti lungo altri livelli.

immagine originale

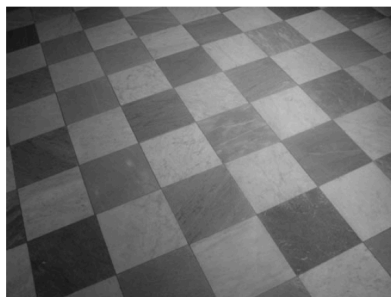
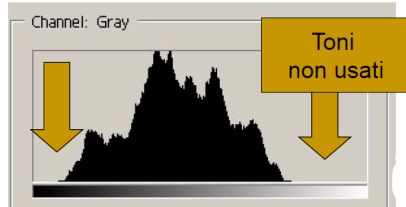
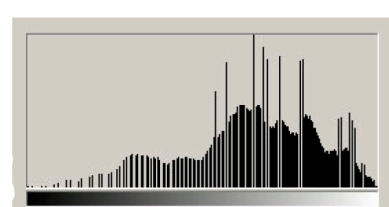


immagine "corretta"



Aritmetica e Normalizzazione sulle immagini

L'applicazione di operazioni aritmetiche sui pixel può generare i cosiddetti **problemi di range**. Un pixel infatti potrebbe assumere un valore:

- Negativo
- Maggiore del valore massimo (tipicamente 255)
- Non intero (risolvibile con un troncamento o approssimazione)

Le soluzioni più comuni per risolvere problemi di range sono 2:

1. **Clamping:** impostare i valori negativi a 0 (nero) e i valori maggiori di 255 a 255 (bianco);

2. **Ri-Normalizzazione:** trasformare ciascun valore utilizzando la seguente equazione:

$$v_{\text{nuovo}} = 255 \cdot \frac{v_{\text{vecchio}} - \min_{\text{osservato}}}{\max_{\text{osservato}} - \min_{\text{osservato}}}$$

Equalizzazione dell'Istogramma

Un'immagine si definisce **equalizzata** quando il contributo di ogni differente tonalità di grigio è pressoché uguale. Si parla in questo caso di "Istogramma uniforme" o "appiattito".

L'equalizzazione si ottiene usando appositi algoritmi, non sempre però produce un miglioramento visivo dell'immagine.

L'**algoritmo di equalizzazione** si basa sul calcolo della probabilità normalizzata del livello di grigio e sulla sua sommatoria cumulata:

- Se r_k è un livello di grigio e n_k il numero di pixel nell'immagine $M \times N$ di quel livello di grigio si può definire $p_r(r_k) = n_k/MN$ per $k = 0, 1, 2, \dots, L - 1$

La visualizzazione di r_k contro $p_r(r_k)$ è di fatto l'istogramma dell'immagine.

- I nuovi valori di grigio dell'istogramma equalizzato s_k sono definiti

$$\text{dalla trasformazione } T(r_k): s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) = \frac{L-1}{MN} \sum_{j=0}^k n_j$$

Dove L è il numero massimo di livelli di grigio.

Classificazione delle Operazioni sulle Immagini

Le operazioni sulle immagini alterano i valori dei pixel, facendo sì che l'immagine finale appaia diversa da quella iniziale. Queste operazioni agiscono sia su immagini a colori che su immagini a toni di grigio.

Nel **dominio spaziale**, l'elaborazione può essere espressa come: $g(x, y) = T[f(x, y)]$, dove f è l'immagine di ingresso, g quella di uscita e T un operatore definito in un intorno del pixel (x, y) .

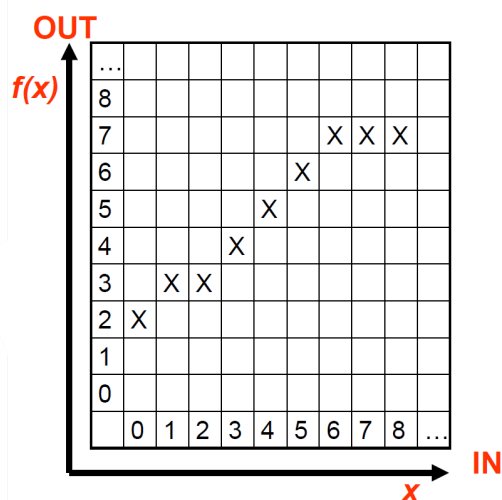
La dimensione dell'intorno (x, y) definisce il tipo di elaborazione:

- **Puntuale:** l'intorno coincide col pixel stesso;
- **Locale:** l'intorno è una piccola regione, ad esempio un quadrato sul pixel;
- **Globale:** l'intorno coincide con l'intera immagine di f .

Operatori Puntuali

Si dice operatore puntuale, un operatore che preso in input un valore di un pixel ne restituisce uno cambiato che dipende esclusivamente dal valore del pixel in ingresso.

Un' operatore puntuale può essere rappresentato da una funzione $g(x,y) = T(f(x,y))$ dove $f(x,y)$ e $g(x,y)$ appartengono allo stesso capo di definizione (ad esempio entrambi tra 0 e 255). Poiché dipende solo dal valore del pixel, è completamente descritto da una tabella o un grafico chiamato **Look-Up-Table (LUT)**.



IN	0	1	2	3	4	5	6	7	...
OUT	T(0)	T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)	...

Questo grafico è l'interfaccia standard usata nei programmi di elaborazione immagini.

Le tipiche operazioni puntuali sono:

- **Aggiunta o sottrazione di una costante:** per compensare sotto o sovraesposizioni
- **Inversione della scala dei grigi (negativo):** l'operazione puntuale più semplice che associa al valore $f(x,y)$ del pixel il valore $255 - f(x,y)$
- **Espansione del contrasto**
- **Modifica dell'Istogramma:** equalizzazione o specificazione
- **Presentazione in falsi colori**
- **Trasformazione logaritmica:** Comprime la gamma dinamica di immagini con escursioni di intensità molto ampie, consentendole la visualizzazione con una scala di grigi usuale, l'espressione è $g(x,y) = c \cdot \log(1 + f(x,y))$ dove c è una costante positiva per normalizzare il risultato tra 0 e 255
- **Trasformazioni di potenza (Gamma Correction):** espressa come $g(x,y) = c \cdot (f(x,y))^y$

- Per $\gamma < 1$, gli effetti sono analoghi alla trasformazione logaritmica (espansione per valori bassi di f , compressione per altri valori).
- Per $\gamma > 1$, gli effetti sono opposti. Questa trasformazione è cruciale nell'applicazione della correzione Gamma alla resa sui monitor.
- **Binarizzazione**: produce un'immagine con solo due livelli (nero e bianco). Si ottiene scegliendo una soglia T : i pixel con valore minore di T sono posti a nero, tutti gli altri a bianco
- **Variazioni di contrasto**: aumentare il contrasto significa rendere più evidenti le differenze di colore, modificando il valore di un pixel con un più scuro o più chiaro. In termini di LUT, un aumento di contrasto si ottiene con una curva a forma di S, mentre una diminuzione di contrasto si ottiene con una curva a forma di S invertita
- **Curve non monotone**: è possibile applicare variazioni alle curve in modo che non siano monotone, un esempio è l'effetto di solarizzazione

(per guardare come variano le curve guarda le slide "lez 09" dalla 39 in poi)

Operatori Locali e Convoluzione

Il concetto fondamentale da cui partiamo è la rappresentazione dei dati. Un vettore di lunghezza N può essere pensato come un elemento di uno spazio N -dimensionale. Di conseguenza è possibile scomporre questo vettore utilizzando la base canonica di tale spazio.

Questo ragionamento si applica anche alle **immagini**.

Operatori Locali

Un'operazione locale è un processo in cui il valore d'uscita di ogni singolo pixel dipende solo da un **limitato intorno** del corrispondente punto in input. Questi operatori sono utilizzati per diversi scopi:

- Migliorare la **qualità** delle immagini.
- Estrarre **informazioni** specifiche dall'immagine.

In sostanza gli operatori locali possono essere visti come **filtraggi** dell'immagine. Un filtraggio si ottiene in particolare eseguendo l'operazione di **convoluzione** tra l'immagine e una specifica matrice.

Operatori lineari

Un operatore $F: V \rightarrow W$ si definisce **lineare** se per qualsiasi coppia di vettori v_1 e v_2 in V e per ogni coppia di scalari a e b vale la proprietà:

$$F(av_1 + bv_2) = aF(v_1) + bF(v_2).$$

Se un operatore è lineare, conoscerne il comportamento su una base di V (come la base canonica per le immagini) è sufficiente per calcolarne il comportamento su qualsiasi elemento di V .

Tuttavia, alcuni operatori lineari non hanno lo stesso comportamento su tutti gli elementi della base canonica, il che significa che il loro effetto varia a seconda della posizione all'interno dell'immagine. Questi operatori sono definiti **NON invarianti per traslazione**. Descrivere questi operatori è "difficile" in quanto richiede di conoscere il comportamento su ciascun impulso in ogni singola locazione dell'immagine.

Un operatore si dice **invariante per traslazione** (o shift invariant) quando il suo comportamento sulle immagini impulsive è sempre lo stesso, indipendentemente dalla posizione del pixel.

Tutti gli operatori puntuali sono invarianti per traslazione anche se non sono lineari.

Ad esempio, l'operatore negativo $f(x, y) = (255 - I(x, y))$ sottrae il valore di grigio di un punto a 255. Poiché questo comportamento è identico per tutti i medesimi livelli di grigio, indipendentemente dalla loro posizione, l'operatore è invariante per traslazione.

Quindi riassumendo:

- Se un operatore F è **lineare**, per descriverlo è necessario conoscerne il comportamento su **tutte le immagini impulsive**.
- Se F è **shift invariant**, si comporta allo stesso modo su tutti gli impulsi indipendentemente dalla loro posizione.
- Se F è **sia lineare che shift invariant**, per descriverlo basta conoscere come si comporta su **un solo impulso**.

La "**risposta all'impulso**" o "**point spread function**" di F è la carta di identità di tale operatore. A un operatore lineare e shift invariant corrisponde una matrice e viceversa. Questa matrice, che descrive la risposta all'impulso, è chiamata anche **kernel** o **maschera** dell'operatore. Spesso un operatore su un'immagine in questo contesto viene chiamato **filtro**.

Convoluzione e Filtraggio

I filtri lineari e invarianti per traslazione sono chiamati anche **filtri convolutivi**.

L'operazione di **convoluzione** ($\otimes$) è fondamentale nel signal processing e nella descrizione di numerosi eventi fisici.

- È **commutativa**: $f \otimes h = h \otimes f$.
- È **associativa**: $(f \otimes h) \otimes h_1 = f \otimes (h \otimes h_1)$.

Applicare un filtro lineare e shift invariante a un'immagine è equivalente a calcolare la convoluzione del kernel del filtro con l'immagine stessa.

Per ragioni pratiche si utilizzano kernel di **dimensioni finite**. La dimensione del kernel influenza la complessità di filtraggio la quale dipende anche dal numero di pixel dell'immagine.

Durante l'implementazione del filtraggio tramite convoluzione sorge il problema di come calcolare l'operazione sui **bordi** dell'immagine. Le soluzioni sono diverse:

1. **Filtrare solo le zone centrali**: si calcola l'output solo per i pixel che hanno un intorno completo all'interno dell'immagine originale. Le aree per cui il kernel "sfora" non vengono calcolate.

input							output					
0	0	0	0	0	0	0	11	19	17	22	11	
0	1	2	2	3	1	0	25	30	45	30	31	
0	3	2	2	1	4	0	25	46	27	37	19	
0	2	5	2	7	1	0	35	34	41	28	29	
0	9	0	1	1	2	0	16	27	12	18	10	
0	3	1	2	4	1	0						
0	0	0	0	0	0	0						

2. **Supporre lo zero intorno all'immagine**: si considera che tutti i pixel esterni all'immagine di input abbiano valore 0.

input					output				
Le aree in grigio non verranno calcolate									
1	2	2	3	1					
3	2	2	1	4		30	45	30	
2	5	2	7	1		46	27	37	
9	0	1	1	2		34	41	28	
3	1	2	4	1					

3. **Topologia toroidale**: quando il kernel "sfora a destra" si rientra a sinistra, quando "sfora in basso" si rientra in alto e viceversa (l'immagine viene trattata come se fosse avvolta su un toro).

input							output				
1	3	1	2	4	1	3	27	30	29	32	33
1	1	2	2	3	1	1	33	30	45	30	40
4	3	2	2	1	4	3	38	46	27	37	45
1	2	5	2	7	1	2	41	34	41	28	48
2	9	0	1	1	2	9	28	35	24	27	40
1	3	1	2	4	1	3					
1	1	2	2	3	1	1					

4. **Aggiungere righe/colonne con valori adiacenti:** si aggiungono righe/colonne ai bordi replicando i valori più vicini (es. una riga all'inizio uguale alla prima riga, una colonna alla fine uguale all'ultima colonna).

input							output				
1	1	2	2	3	1	1	25	27	29	31	33
1	1	2	2	3	1	1	34	30	45	30	39
3	3	2	2	1	4	4	51	46	27	37	32
2	2	5	2	7	1	1	54	34	41	28	35
9	9	0	1	1	2	2	48	32	24	34	26
3	3	1	2	4	1	1					
3	3	1	2	4	1	1					

Esempi di Operatori Locali (filtri)

Filtri statici (Order statics), questi filtri sono non lineari.

- **Filtro Mediano:** Fornisce in uscita il **valore mediano** dell'intorno del pixel in esame. È molto efficace nella rimozione del **rumore impulsivo** (sale e pepe). Non sfoca i lati ma elimina i picchi con base piccola rispetto al kernel.
- **Filtro di Minimo:** Sostituisce il valore del pixel con il **valore minimo** di tutti i valori osservati nell'intorno $m \times m$. Causa un incupimento dell'immagine (es. elimina macchioline chiare).
- **Filtro di Massimo:** Sostituisce il valore del pixel con il **valore massimo** di tutti i valori osservati nell'intorno $m \times m$. Causa uno schiarimento dell'immagine (es. elimina punti neri).
- **Filtro Olimpico:** Si scartano i valori di massimo e minimo da un dato intorno e si calcola la media sul resto dei valori.
- **Filtro Outlier:** Il valore del pixel centrale viene confrontato con il valore della media dei suoi 8 vicini. Se il valore assoluto della

differenza è maggiore di una certa soglia, allora il punto viene sostituito dal valore medio. Altrimenti il pixel non viene modificato.

Filtri di Smoothing (Smussamento), questi filtri sono generalmente lineari.

- **N-box (o di media)**: Sono definiti da kernel $N \times N$ dove ogni elemento è pari a $1/N^2$ (con N generalmente dispari). Hanno l'effetto di **sfocare** le immagini. La sfocatura è molto forte in orizzontale e verticale ma meno in diagonale.
- **N-binomiale (o Gaussiano)**: Filtri di smussamento il cui kernel è derivato dalla distribuzione binomiale, che è un'approssimazione discreta della distribuzione Gaussiana. Hanno il pregio di **smussare in modo eguale in tutte le direzioni**, risultano meno vigorosi degli N-box.

Questi filtri (N-box e N-binomiali) sono anche usati per ridurre il **rumore** in un'immagine. In generale, più grande è il kernel, migliore sarà il risultato anche se aumenta il rischio di sfocatura.

Un filtro "**conserva l'energia**" (con un abuso di linguaggio derivato dalla Fisica) se la somma dei suoi pesi è uguale a 1. Tutti i filtri di smoothing visti in precedenza vengono considerati "energy preserving" in quanto la somma dei valori totali di luminanza nell'immagine non cambia.

Il Rumore e la sua Rimozione

Esistono 2 tipi principali di rumore:

- **Rumore impulsivo** (o sale e pepe): caratterizzato dalla frazione di immagine modificata (in %). Si verifica quando i valori alterati sono valori "saturi" come il minimo ($a = 0$) e il massimo ($b = 255$).
 - **Rimozione**: il filtro mediano funziona meglio del filtro media per questo tipo di rumore.
- **Rumore Gaussiano Bianco**: caratterizzato dalla media ($\bar{z}$) dalla varianza (σ^2). La sua funzione di probabilità è data da
$$p(z) = 1/\sqrt{2\pi\sigma} e^{-(z-\bar{z})^2/2\sigma^2}.$$
 - **Rimozione**: il filtro media funziona meglio del filtro mediano per questo tipo di rumore.

Media vs. Mediano: Il filtro media tende a creare livelli di grigio prima inesistenti e attenua indiscriminatamente le alte frequenze spaziali, causando immagini sfocate. Il filtro mediano, invece, non deteriora i lati (bordi).

Per un rumore molto diffuso, applicare iterativamente un kernel più piccolo (es. mediano 3 × 3 due volte) produce risultati migliori, riducendo la sfocatura rispetto all'applicazione di un singolo kernel più grande (es. mediano 5 × 5).

(guarda la slide lez 10 pag 32-55 per vedere gli esempi)

Estrazione dei Contorni (Edge Detection)

Gli operatori locali sono essenziali per l'estrazione dei contorni da un'immagine. I contorni sono definiti come delle **discontinuità locali della luminanza**. Gli edge detector generano immagini in cui sono preservate le variazioni di luminanza e tutte le altre informazioni vengono eliminate. Esistono diversi tipo di contorni come lo Step Edge, il Ramp Edge, il Roof Edge e il Line Edge.

Edge detector basati sulla Derivata Prima

In un segnale monodimensionale, i lati (edges) si trovano in corrispondenza dei **massimi della derivata prima**. Pertanto i filtri devono calcolare la derivata prima in direzione x e in direzione y e poi combinarle.

Kernel Notevoli per Derivata prima (**Lati Orizzontali**)

$$Sobel_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad Prewitt_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Kernel Notevoli per Derivata prima (**Lati Verticali**)

$$Sobel_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad Prewitt_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Magnitudo del Gradiente

- $Sobel_x$ fornisce una matrice con valori non nulli per i lati orizzontali (e le componenti orizzontali dei lati obliqui).
- $Sobel_y$ fornisce una matrice con valori non nulli per i lati verticali (e le componenti verticali dei lati obliqui).

Le due matrici possono essere combinate per calcolare la **magnitudo** del gradiente mediante la seguente formula: $magnitudo = \sqrt{Sobel_x^2 + Sobel_y^2}$.

La matrice ottenuta ha valori non nulli per i pixel di lato. Fissando una soglia adeguata si può ottenere una matrice binaria che indica se un pixel è un bordo o meno. Lo stesso vale per i filtri Prewitt.

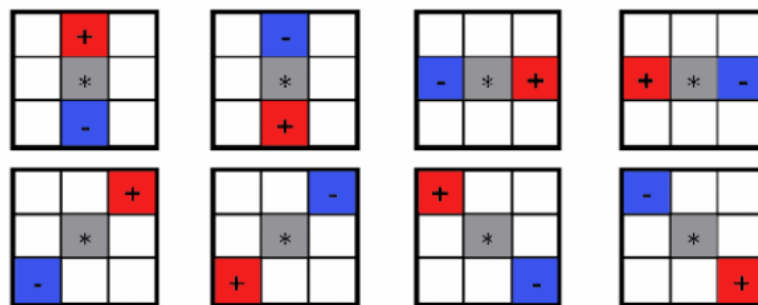
Edge Detector basati sulla Derivata Seconda

In un segnale monodimensionale, la derivata seconda passa per lo **zero** in corrispondenza del lato (zero-crossing).

Kernel notevole per la Derivata Seconda (**Laplaciano**)

$$Laplaciano = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

Dopo l'applicazione del Laplaciano, è necessario che si verifichi la condizione di **Zero-crossing**, ovvero che nell'intorno del punto in questione ci sia un valore positivo e uno negativo.



Filtri di Sharpening

I filtri di sharpening hanno lo scopo di **incrementare la nitidezza** di un'immagine, aumentando il contrasto locale. Questa operazione è l'opposto dello sfocamento (smoothing).

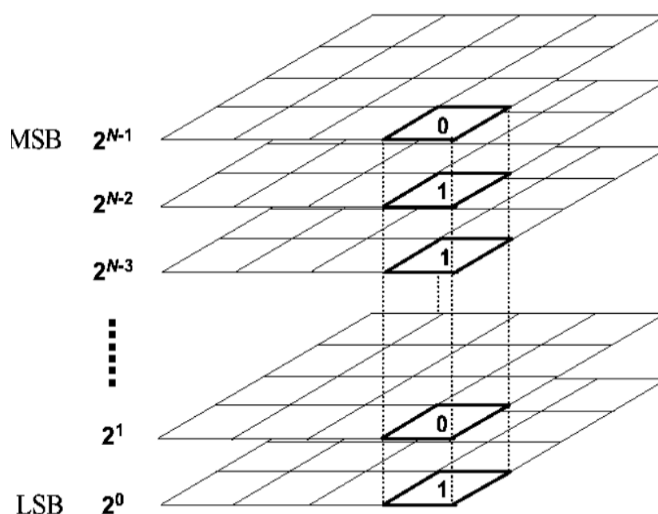
Per ottenere questo effetto, si può adottare una maschera derivata dal Laplaciano che "rinforza" i lati presenti nell'immagine. Un esempio di maschera per lo sharpening è:

-1	0	-1
0	5	0
-1	0	-1

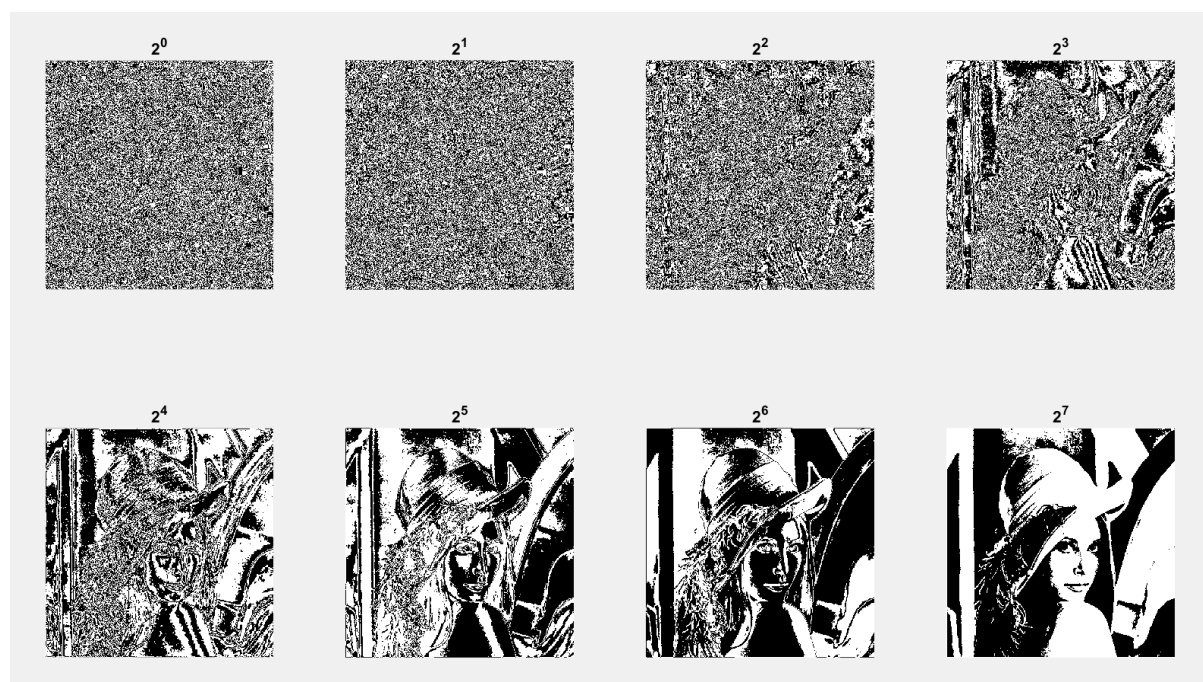
Tuttavia questi filtri rinforzano anche il rumore presente nell'immagine.

I Bit Plane

Un'immagine con una profondità di colore di N bit (ad esempio $N = 8$ per una scala di grigi) può essere scomposta e rappresentata come N distinti **piani di bit (bit-planes)**. Ciascuno di questi piani può essere visto come una singola immagine binaria. Il **bit-plane** di un'immagine digitale a N bit è definito come un insieme di N immagini binarie (piani). In questo insieme, l'immagine i -esima contiene specificamente i valori dell' i -esimo bit della codifica scelta per l'immagine.



Si può stabilire un ordine per questi piani che varia dal **Most Significant bit (MSB)**, che corrisponde al peso 2^{N-1} , fino al **Least Significant Bit (LSB)**, che corrisponde al peso 2^0 . Ad esempio per un'immagine a 8 bit avremo piani che vanno da 2^7 (MSB) a 2^0 (LSB).



Codifica in Binario Puro e Proprietà

Quando si utilizza la codifica in binario puro per rappresentare i livelli di grigio dei pixel, i piani di bit rappresentano le seguenti caratteristiche:

- **Piani Significativi (MSB):** I piani di bit più significativi (quelli con peso maggiore, come $2^7, 2^6$, ecc.) contengono l'informazione relativa alla **struttura generale** dell'immagine.
- **Piani Meno Significativi (LSB):** I piani di bit via via meno significativi forniscono i **dettagli** sempre più piccoli dell'immagine.
- **Rumore ed Errori:** Si osserva che il rumore presente nell'immagine e gli errori derivanti dall'acquisizione sono **più evidenti** e concentrati nei piani a peso più basso (LSB). Ad esempio, in un'immagine a 8 bit, solo i piani dal 7 al 3 potrebbero contenere dati visivamente significativi.

Un'applicazione di questa scomposizione in binario puro è la possibilità di **eliminare tutti i valori compresi in un determinato range** di intensità. Questo si ottiene azzerando uno o più piani di bit specifici. Ad esempio per eliminare tutti i livelli di grigio compresi tra 32 e 63, è sufficiente porre a 0 il quinto bit (relativo al piano $2^5 = 32$), e quindi tutto il piano 5.

Un altro uso del binario puro è la **ricostruzione dell'immagine** utilizzando solo un sottoinsieme di piani di bit, tipicamente quelli più significativi. La ricostruzione assume solo alcuni valori discreti (livelli di grigio) corrispondenti alla somma dei pesi dei piani utilizzati.

Problema della Codifica Binario Puro

Sebbene intuitiva, la codifica in binario puro presenta uno svantaggio significativo nel contesto dei bit-plane: **Propagazione degli Errori**. Una piccola variazione nell'intensità di un pixel può ripercuotersi su tutti i piani di bit.

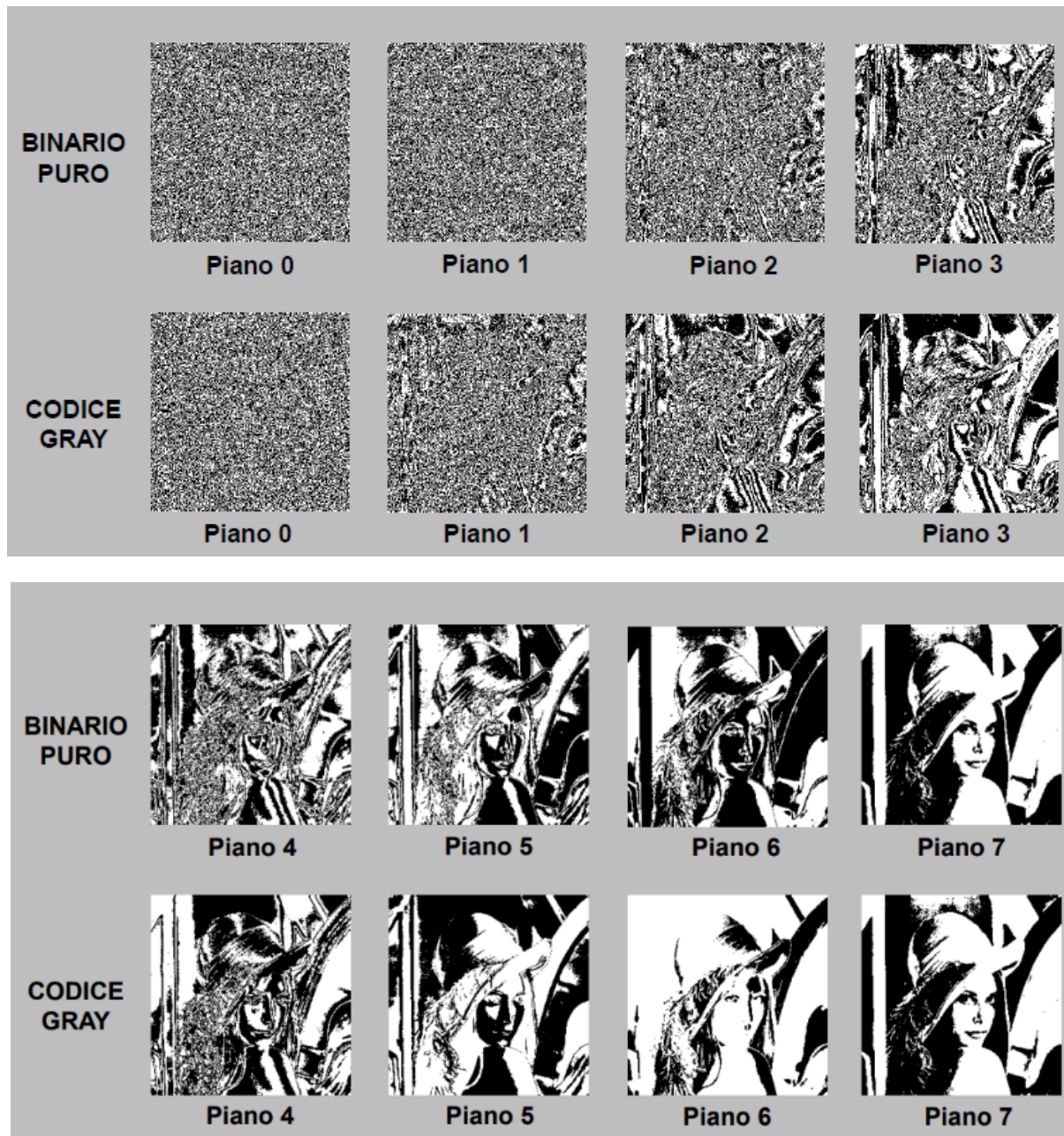
Esempio: se un pixel ha intensità 127 (01111111) e il suo adiacente ha intensità 128 (10000000) allora la transizione 0 e 1 si ripercuote su tutti i piani di bit. Questo crea molta "discontinuità" in tutti i piani di bit per una transizione di grigio minima.

Per mitigare questo problema è necessario un codice in cui i valori di intensità molto vicini abbiano rappresentazioni binarie che siano **molto simili**. La soluzione è l'adozione del **Codice Gray**.

Il Codice Gray a m bit $g_{m-1} \dots g_1 g_0$ che corrisponde al numero in binario puro $a_{m-1} \dots a_1 a_0$ può essere calcolato con la formula:

$$g_i = a_i \oplus a_{i+1}, 0 \leq i \leq m - 2, g_{m-1} = a_{m-1}$$

dove $\oplus$ denota l'operatore XOR (OR esclusivo)



Questo codice gode di una proprietà fondamentale: **Distanza di Hamming=1**. Ogni codeword (il codice binario di un numero) differisce dalla codeword del numero precedente per la variazione di **un solo bit**.

Prendendo il caso di prima, 127 e 128 hanno 8 bit di variazione usando il Binario Puro. Usando il Codice Gray 127 diventa 01000000 e 128 diventa 11000000 e così variano di un solo bit.

L'utilizzo del Codice Gray produce dei bit-plane con caratteristiche diverse e vantaggiose rispetto al binario puro:

- **Coerenza:** I bit-plane in Codice Gray risultano tra loro più "coerenti". Se si aumenta l'intensità di un pixel di 1, varia solo un bit, il che significa che l'impatto si limita a **un solo piano**.
- **Complessità Descrittiva:** Il numero di transizioni da bianco a nero all'interno di un singolo piano (la sua complessità descrittiva) è **inferiore** quando si usa il Codice Gray.
- **Entropia e Compressione:** Queste caratteristiche implicano una **minore entropia** (e quindi una **maggiore ridondanza**) nell'immagine codificata in Gray. Di conseguenza diventa più semplice e potenzialmente più efficiente da comprimere immagini codificate in questo modo.

È fondamentale ricordare che le proprietà dei piani di bit (come quelle relative all'eliminazione di range di valori) viste per la codifica in binario puro **non valgono allo stesso modo** per il Codice Gray, in quanto il significato (peso) associato a ciascun bit è differente tra le due codifiche.

- Se si azzerano dei piani di bit in Codice gray, si eliminano **range di valori diversi** (e meno significativi) rispetto al binario puro.
- Nonostante dettagli e rumore si concentrino comunque nei piani meno significativi anche con il Codice Gray, l'eliminazione diretta di tali piani potrebbe introdurre artefatti indesiderati.

Fourier e Miglioramento delle immagini

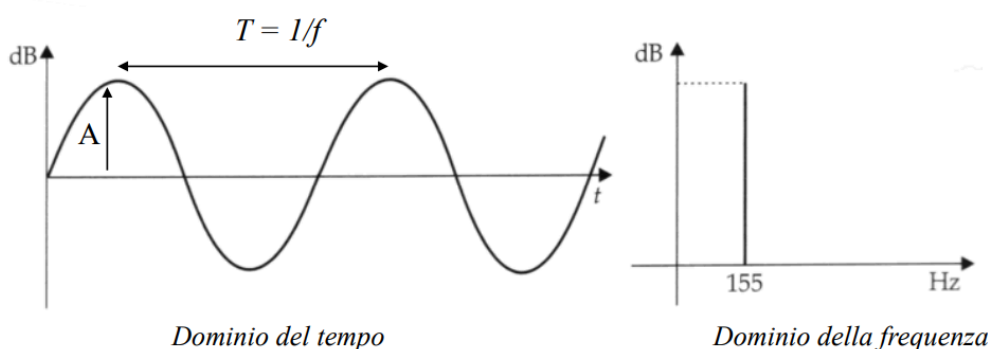
L'elaborazione delle immagini nel dominio delle frequenze affonda le sue radici nelle scoperte di **Jean Baptiste Fourier**, il quale dimostrò che ogni funzione periodica può essere espressa come una somma di seni e coseni con ampiezze e frequenze differenti. Questo concetto si estende anche alle funzioni non periodiche attraverso la **Trasformata di Fourier**, dove la funzione viene rappresentata come un integrale di seni e coseni moltiplicati per specifiche funzioni-peso.

Un aspetto fondamentale di questa analisi è la **reversibilità**: è possibile passare dal dominio originale (spaziale) a quello di Fourier e tornare indietro senza perdere informazioni. Inizialmente l'analisi di Fourier era utilizzata per lo studio del calore ma negli anni '60 con l'introduzione della **FFT (Fast Fourier Transform)** il settore dell'elaborazione digitale dei segnali (**DSP - Digital Signal Processing**) ha subito una vera e propria rivoluzione.

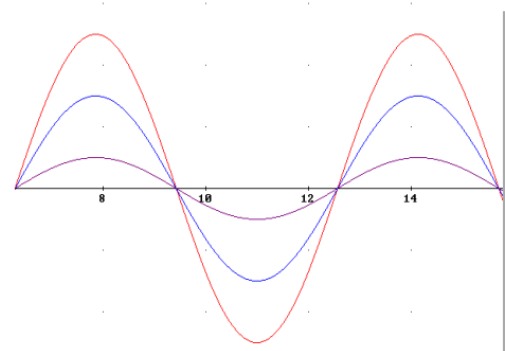
Immagine come segnale

Nel contesto multimediale, un'immagine non è altro che una **funzione discreta bidimensionale** $f(x, y)$, dove i valori rappresentano i livelli di grigio dei singoli pixel. Possiamo quindi trattare un'immagine come un segnale con una propria frequenza.

Nel dominio del tempo e dello spazio misuriamo ampiezza, periodo e frequenza, nel dominio della frequenza invece analizziamo come questi componenti si distribuiscono.



- Ampiezza (A) espressa in decibel dB;
- Periodo (T) espresso in secondi;
- Frequenza (f) numero di cicli (onde) al secondo; si misura in Hertz Hz



Per gestire digitalmente queste informazioni, utilizziamo la **DFT (Discrete Fourier Transform) 2-D**. La formula trasforma l'immagine spaziale in una matrice di coefficienti complessi:

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

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

dove u e v sono gli indici relativi agli assi frequenze discretizzati, mentre M e N sono le dimensioni (in pixel) dell'immagine.

Da questi coefficienti complessi possiamo estrarre 3 informazioni chiave:

- Lo **Spettro (o Modulo)**: indica quali frequenze sono presenti e con quale intensità. Esso è descritto dalla seguente formula:

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

- L'**Angolo di Fase**: definisce dove si trovano le strutture periodiche nell'immagine originale. La fase è cruciale, ricostruire l'immagine usando solo il modulo produce risultati irriconoscibili, mentre la sola fase conserva gran parte della struttura visiva. Esso è descritto dalla seguente formula: $\phi(u, v) = \tan^{-1}[\frac{I(u, v)}{R(u, v)}];$

- La **Potenza Spettrale**: rappresenta il quadrato del modulo. Essa è definita dalla seguente formula: $P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v).$

Tutti i valori della $f(x)$ contribuiscono alla costruzione di ciascuno dei campioni della $F(u)$. Analogamente tutti i campioni della trasformata contribuiscono a ciascuno dei valori della $f(x)$.

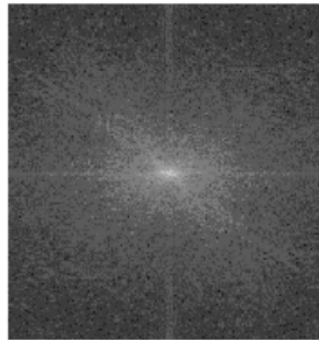
Visualizzazione e Proprietà

Uno dei problemi principali dello spettro di Fourier è il suo enorme **range dinamico**. Ad esempio lo spettro dell'immagine di Lena può variare da 0 a oltre 6 milioni. Per visualizzarlo correttamente su schermi con soli $L=256$ livelli di grigio bisogna applicare una **compressione logaritmica** per visualizzare invece che lo spettro una funzione del tipo: $D(u, v) = c \log(1 + F(u, v))$, dove c è una costante di scala che va scelta opportunamente per far ricadere i valori trasformati nel range voluto, in

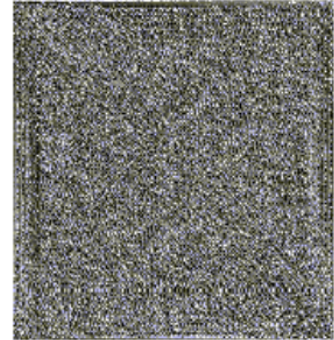
questo caso $[0, L-1]$. Senza questa operazione vedremmo solo un puntino luminoso al centro dello spettro.



$f(x,y)$



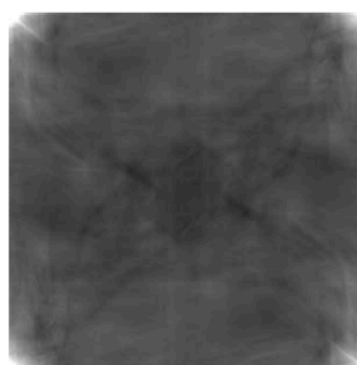
$|F(u,v)|$



$\Phi(u,v)$

L'**Ampiezza** (figura al centro) contiene l'informazione relativa al fatto che una certa struttura periodica è presente nell'immagine. L'ampiezza contiene informazioni su *quanto* contrasto e *quali* pattern ripetitivi ci sono (le texture), ma **non sa dove metterli**. Senza la fase, i "mattoni" dell'immagine ci sono tutti, ma sono sparsi a caso. La **Fase** contiene l'informazione relativa al **dove sono collocate** le strutture periodiche evidenziate nella DFT.

Ricostruzione da solo modulo



Ricostruzione da sola fase

Vantaggi delle Trasformate di Fourier

Nello spazio delle frequenze è possibile **sopprimere frequenze indesiderate**, **ridurre lo spazio occupato** dai dati limitando la degenerazione del segnale ed è possibile anche **rigenerare segnali degradati**.

Esistono diverse trasformate utilizzate nell'immagine processing, con largo impiego nel restauro e soprattutto nella compressione. Le più importanti sono: La trasformata discreta di Walsh (DWT), La trasformata discreta di Hadamard (DHT), La trasformata discreta del Coseno (DCT), La trasformata discreta di Karhunen Loeve (KLT).

Proprietà della DFT (Discrete Fourier Transform)

Tra le proprietà più importanti della DFT troviamo:

- **Separabilità:** la DFT può essere espressa in forma **separabile**. In particolare vale la seguente espressione:

$$F(u, v) = \frac{1}{M} \sum_{x=0}^{M-1} k(x, v) e^{\frac{-i2\pi ux}{M}}$$
$$k(x, v) = \left[\frac{1}{N} \sum_{y=0}^{N-1} f(x, y) e^{\frac{-i2\pi vy}{N}} \right]$$

è dunque possibile calcolare la trasformata 2-D applicando in successione 2 trasformate 1-D;

- **Traslazione:** è spesso utile prima di operare sulla trasformata applicare una traslazione dell'origine nel centro della matrice. In questo modo i dati vengono traslati in maniera tale che $F(0, 0)$ risulti il centro del rettangolo delle frequenze definito tra $[0, M - 1]$ e $[0, N - 1]$. Essa **non modifica lo spettro** della trasformata, una rotazione invece ruota anche lo spettro;
- **Valor Medio:** il valore della trasformata nell'origine, cioè nel punto $(u, v) = (0, 0)$ è dato da:

$$F(0,0) = \frac{1}{NxN} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \quad \bar{f}(x, y) = \frac{1}{NxN} F(0,0)$$

non è altro che la media di $f(x, y)$. Il valore della trasformata di Fourier nell'origine è uguale alla media dei valori di grigio contenuti nell'immagine e viene chiamato **componente DC**.

Nella sua forma classica implementare la Trasformata di Fourier richiederebbe una complessità di $O(n^2)$, mentre con la FFT (Fast Fourier Transform) la complessità si riduce a $O(n \log_2 n)$.

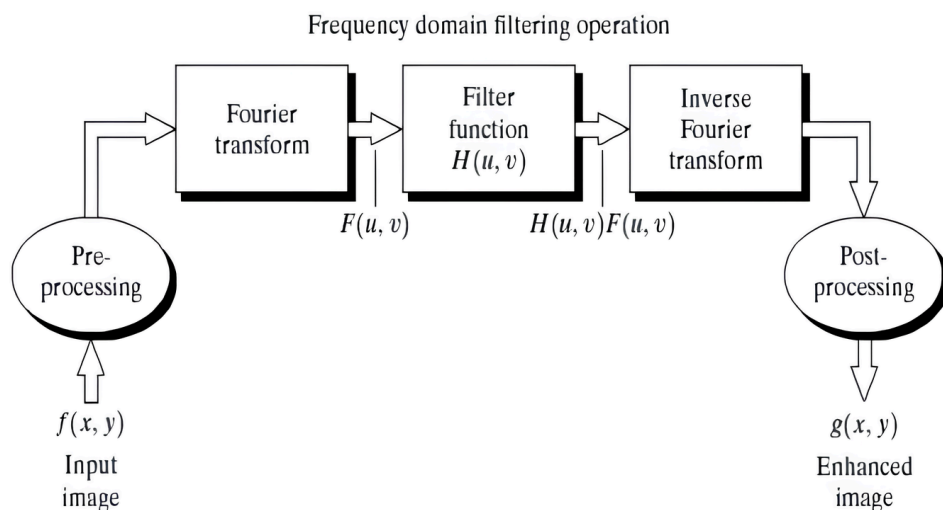
Filtraggio nel Dominio della Frequenza

Il vero vantaggio della trasformata di Fourier risiede nella possibilità di manipolare l'immagine operando direttamente sulle frequenze:

- **Basse frequenze:** associate a zone uniformi e sfumate.
- **Alte frequenze:** associate a variazioni brusche, bordi e rumore.

Il **filtraggio** nel dominio della frequenza si basa su una procedura fondamentale che trasforma l'immagine originale per elaborarla e poi ricostruirla. Il flusso di lavoro si articola in tre fasi principali:

1. **Pre-processing e Trasformata:** L'immagine di input $f(x, y)$ viene trasformata tramite la trasformata di Fourier per ottenere la sua rappresentazione in frequenza, $F(u, v)$.
2. **Filtraggio:** Questa rappresentazione viene moltiplicata, punto per punto, per una funzione filtro $H(u, v)$. Questa funzione H è detta "filtro" proprio perché agisce selettivamente sulle frequenze, sopprimendone alcune e lasciandone inalterate altre. Spesso H è una funzione reale che moltiplica sia la parte reale che quella immaginaria di F senza introdurre sfasamenti (filtri *zero-phase-shift*).
3. **Post-processing e Antitrasformata:** Il risultato del prodotto, $H(u, v)F(u, v)$, viene riportato nel dominio spaziale tramite la trasformata inversa di Fourier, ottenendo l'immagine elaborata (o *enhanced*) $g(x, y)$.



Il **Teorema della Convoluzione** afferma: “La trasformata della convoluzione di due segnali nel dominio spaziale equivale al prodotto delle trasformate dei due segnali”. Ciò significa che la convoluzione di due segnali nel dominio spaziale è equivalente alla semplice moltiplicazione delle loro trasformate nel dominio della frequenza.

È possibile dimostrare matematicamente perché conviene passare alle frequenze, prendiamo l'operazione di convoluzione nel dominio spaziale:

$$g(x, y) = f(x, y) \oplus h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) h(x - m, y - n)$$

- $f(m, n)$: rappresenta l'**immagine di input**. Le coordinate (m, n) scorrono su tutti i pixel dell'immagine.
- $h(x - m, y - n)$: rappresenta la **maschera** (il filtro). Il termine $(x - m, y - n)$ indica che il filtro viene spostato attraverso l'immagine.

Essa nel dominio delle frequenze si trasforma in una semplice moltiplicazione diretta: $G(u, v) = F(u, v)H(u, v)$

- $G(u, v)$: rappresenta la **trasformata dell'immagine finale**.
- $F(u, v)$: rappresenta la **trasformata di Fourier dell'immagine originale**.
- $H(u, v)$: rappresenta la **trasformata di Fourier del filtro** (chiamata anche funzione di trasferimento).

Per ottenere l'immagine filtrata finale $g(x, y)$ nel dominio spaziale bisogna fare l'antitrasformazione del prodotto $(F \cdot H)$ e applicare la **Trasformata inversa di Fourier** (F^{-1}) e ottenere dunque $g(x, y) = F^{-1}\{F(u, v)H(u, v)\}$.

La scelta tra i due metodi dipende dalle dimensioni del filtro:

- Se il filtro ha dimensioni confrontabili con quelle dell'immagine, è decisamente più efficiente lavorare nel dominio delle frequenze.
- Se invece utilizziamo maschere molto piccole, il calcolo diretto nel dominio spaziale è più vantaggioso.

Definire le proprietà di un filtro risulta spesso più intuitivo ragionando nel dominio delle frequenze.

Creazione di un filtro da una maschera spaziale

Se possediamo una maschera definita nel dominio spaziale e vogliamo convertirla in un filtro di frequenza H bisogna seguire 4 passaggi:

1. Si crea una matrice per il filtro H avente le stesse dimensioni dell'immagine I ;
2. Si posizionano i valori della maschera spaziale nell'angolo in alto a sinistra di H , riempiendo il resto della matrice con 0;
3. Si effettua un'operazione di shift su H per centrare le componenti;

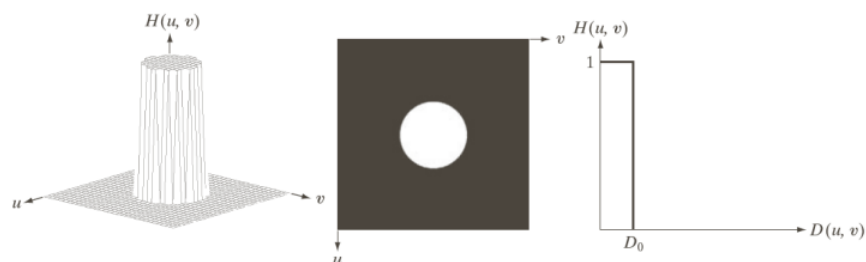
4. Si calcola la trasformata di Fourier di questa matrice per ottenere il filtro H da utilizzare nella moltiplicazione.

Tecniche di Filtraggio nel dominio delle frequenze

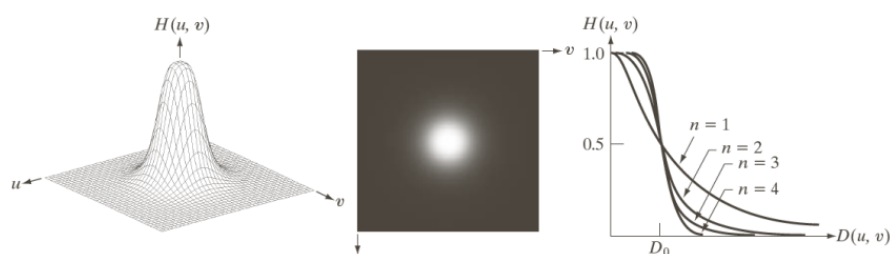
Il concetto fondamentale è che l'immagine viene modificata moltiplicando la sua trasformata di Fourier $F(u, v)$ per una funzione filtro $H(u, v)$. A seconda di come è definita $H(u, v)$, possiamo **attenuare** o **accentuare** determinate frequenze. Le due categorie principali trattate sono i filtri Low Pass e i filtri High Pass.

I filtri **Low Pass** attenuano le frequenze alte (associate ai dettagli fini e al rumore) e lasciano passare le basse frequenze (associate alle zone uniformi). L'effetto visivo risultante è una **sfocatura** o "**smoothing**" dell'immagine. La variabile $D(u, v)$ rappresenta la distanza di un punto (u, v) dal centro del rettangolo delle frequenze. Esistono 3 tipi principali di filtri Low Pass:

- **Low Pass Ideale:** taglia in modo netto tutte le frequenze che superano una certa distanza D_0 dal centro. La sua funzione di trasferimento è binaria, vale 1 se la distanza $D(u, v)$ è minore o uguale a D_0 e 0 se è maggiore. Sebbene teoricamente semplice può provocare artefatti indesiderati nel dominio spaziale.



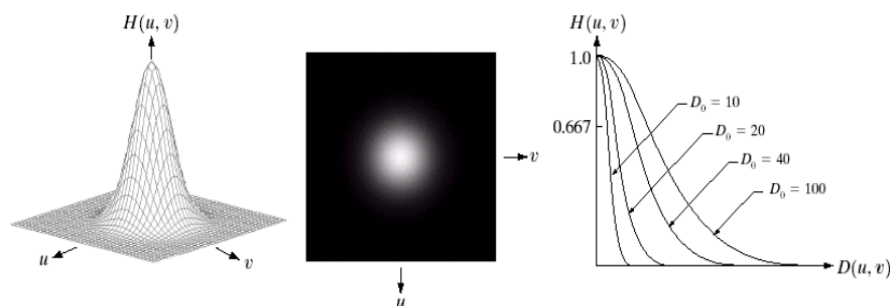
- **Low pass di Butterworth:** a differenza dell'ideale, questo filtro non presenta una discontinuità netta tra le frequenze passanti e quelle tagliate. La transizione è gestita da un parametro n , chiamato "ordine" del filtro. La formula è data da: $H(u, v) = 1/(1 + [D(u, v)/D_0]^{2n})$. All'aumentare dell'ordine n il comportamento del filtro si avvicina sempre più a quello ideale mentre per valori bassi di n la transizione è più morbida (riduce l'effetto ringing).



- **Low pass Gaussian:** questo filtro utilizza una funzione gaussiana. La sua caratteristica principale è che la trasformata di Fourier di una gaussiana è anch'essa una gaussiana, il che evita il fenomeno del “ringing”. La formula è:

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

dove D_0 controlla l'ampiezza della campana gaussiana e quindi la severità del filtraggio.



Il **ringing** è causato dal taglio troppo brusco delle frequenze (come nel filtro ideale). Questo "muro" nel dominio delle frequenze si traduce matematicamente in oscillazioni ondulatorie nel dominio spaziale (funzione Sinc), che "sporcano" i bordi dell'immagine. Più morbida è la curva del filtro (come nel Gaussiano), minore (o nullo) sarà il ringing.

I filtri **High Pass** operano in modo opposto: attenuano le basse frequenze e lasciano passare le alte. Questi filtri vengono utilizzati per l'estrazione dei bordi o per rendere più **nitida** l'immagine. Esistono 3 tipi principali di filtri High Pass:

- **High Pass Ideale:** azzerà tutte le frequenze all'interno del cerchio di raggio D_0 e lascia passare tutto ciò che passa all'esterno. La formula è l'opposto del caso low pass: vale 0 se la distanza $D(u, v)$ è minore o uguale a D_0 e 1 se è maggiore.
- **High Pass Butterworth:** come per il low pass, offre una transizione più graduale. La formula è leggermente diversa: $H(u, v) = 1 / (1 + [D_0 / D(u, v)]^{2n})$. Anche qui n determina la ripidezza della curva di transizione.
- **High Pass Gaussian:** derivato dalla gaussiana offre un risultato naturale senza artefatti di ringing marcati. La formula è:

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

Questo comportamento è complementare a quello del filtro low pass.

Inoltre, vengono citati brevemente anche i **Filtri Band-Reject** (elimina banda), che rimuovono una specifica banda di frequenze definita da una larghezza W attorno a una frequenza centrale D_0 .

Le loro formule combinano logiche simili a quelle viste sopra, bloccando le frequenze intermedie e lasciando passare quelle molto basse o molto alte.

TABLE 4.4

Lowpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

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

TABLE 4.5

Highpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

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

TABLE 4.6

Bandreject filters. W is the width of the band, D is the distance $D(u, v)$ from the center of the filter, D_0 is the cutoff frequency, and n is the order of the Butterworth filter. We show D instead of $D(u, v)$ to simplify the notation in the table.

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

Compressione

Con il termine **compressione di dati** si intende una tecnica di elaborazione che, tramite specifici algoritmi, riduce la quantità di bit necessari per rappresentare un'informazione in forma digitale. L'obiettivo principale è duplice:

- Ridurre lo **spazio di memorizzazione** necessario per il file;
- Diminuire l'**occupazione di banda** richiesta per la trasmissione dei dati.

Ridondanza

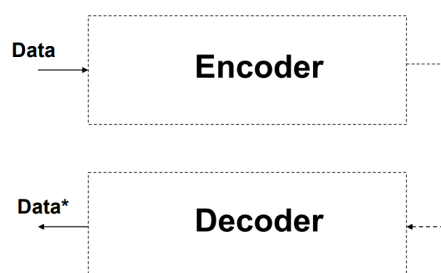
Il principio fondamentale della compressione è l'eliminazione della **ridondanza**. Poiché quantità diverse di dati possono rappresentare la stessa informazione, le rappresentazioni che contengono dati ripetuti o irrilevanti sono considerate ridondanti. Esistono diverse forme di ridondanza:

- **Ridondanza della codifica**: si verifica quando la lunghezza del codice usato per rappresentare i simboli non è ottimizzata rispetto alla loro frequenza. Un codice è un sistema di simboli utilizzati per rappresentare una certa quantità di informazioni, ad ogni "pezzo" di informazione e ad ogni evento è assegnata una sequenza di simboli codificati chiamati **codeword**.
- **Ridondanza spaziale e temporale**: è tipica delle immagini e dei video. In un'immagine, i pixel vicini tendono ad avere valori simili (correlazione spaziale); nei video, i frame consecutivi spesso variano poco (correlazione temporale);
- **Informazione percettivamente irrilevante**: riguarda dati che il sistema sensoriale umano non è in grado di percepire e che quindi possono essere scartati senza una perdita apparente di qualità.

Algoritmi di compressione

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

Il processo di compressione e decompressione



Esistono due tipi di classificazione dei metodi di compressione:

- Basata sul tipo di dati (generico, audio, immagini, video):
- Basata sul tipo di compressione (lossless, lossy).

Gli algoritmi di compressione basati sul tipo di compressione si dividono in due macro-categorie:

- **Lossless (reversibile):** permette di ricostruire i dati originali esattamente, bit per bit, senza alcun errore. È essenziale per file come documenti di testo o programmi eseguibili.
- **Lossy (irreversibile):** ottiene tassi di compressione più elevati accettando una perdita di informazione. È utilizzata per immagini, audio e video dove una ricostruzione approssimata è accettabile.

Compressione Lossless

Per valutare l'efficacia di una compressione lossless, ci rifacciamo alla **teoria dell'informazione**. La **frequenza** di un simbolo è data dal numero delle sue occorrenze diviso per il numero totale di caratteri. Da qui definiamo l'**entropia** (E), che misura la quantità media di informazione (o incertezza) associata alla generazione di un simbolo. La formula dell'entropia è: $E = - \sum f_i \log_2(f_i)$ dove f_i è la frequenza del simbolo i . L'entropia è massima quando tutti i simboli sono equiprobabili.

Primo Teorema di Shannon

Il primo Teorema di Shannon (1948) stabilisce un limite teorico:

Per una sorgente discreta e a memoria zero, il bitrate minimo è pari all'entropia della sorgente.

Cioè 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. Il teorema tuttavia fissa il numero minimo di bit ma non ci dice come trovarli. Un algoritmo che a ciò è stato proposto da Huffman.

Codifica di Huffman

David Huffman (1953) propose un algoritmo greedy che permette di avvicinarsi al limite teorico di Shannon. Si tratta di una codifica a **lunghezza variabile**: i simboli più frequenti vengono codificati con meno bit, mentre quelli meno frequenti con più bit. Inoltre, è un codice a prefisso: nessun codice è l'inizio di un altro codice, garantendo l'**univocità della decodifica**.

Funzionamento algoritmo

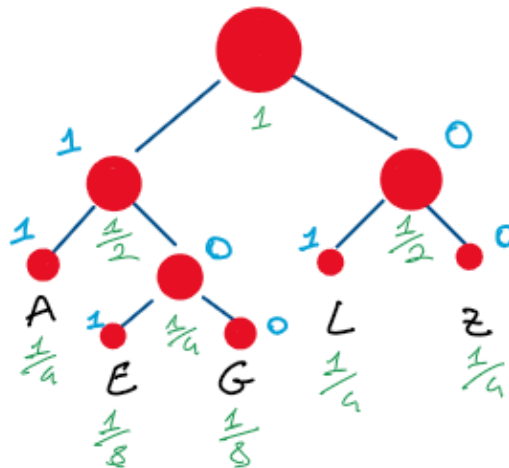
La codifica di Huffman è un algoritmo **iterativo**.

1. Ad ogni iterazione si scelgono due nodi con frequenza più bassa.
2. Questi due nodi vengono collegati per formare un sottoalbero, la cui frequenza del nodo padre è la somma delle frequenze dei due nodi.
3. Si ripete il processo fino ad avere un unico albero (la cui radice avrà peso 1).
4. Si etichettano i rami, assegnando 1 ai rami di sinistra e 0 a quelli di destra.

Sebbene ottimale, Huffman richiede di memorizzare o trasmettere anche la "tabella" (dizionario) caratteri-codici, che può rappresentare un costo aggiuntivo. Huffman viene usato per comprimere alcune informazioni nella **fase finale** della codifica JPEG, è usato negli standard di compressione CCITT, JBIG2, JPEG, MPEG-1,2,4.

- Applicare la codifica di Huffman alla stringa "gazzella". Quanti bit sono effettivamente usati?
- Quanti bit si dovrebbero usare secondo il teorema di Shannon per la codifica della stringa "gazzella"?

GAZZELLA = 8
A = 2/8 = 1/4 = 11
E = 1/8 = 1/8 = 101
G = 1/8 = 1/8 = 100
L = 2/8 = 1/4 = 01
Z = 2/8 = 1/4 = 00



Tramite la codifica di Huffman avremo il codice **100110000101010111** per la parola gazzella che occupa 18 bit. Applicando il teorema di Shannon avremo: $8(-\frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4}) = 18$. Abbiamo provato così che per questo caso specifico la codifica di Huffman è ottimale.

Altri algoritmi di compressione Lossless

Esistono altri approcci lossless che sfruttano specifiche regolarità strutturali dei dati come la ripetizione consecutiva di valori o la loro lenta variazione nel tempo tra questi troviamo:

- **Run-Length Encoding** (RLE).
- **Codifica differenziale**.

Run-Length Encoding

Efficace per sequenze con lunghe **ripetizioni di simboli uguali** (dette **run**). Invece di scrivere "00000", si scrive la coppia (simbolo, lunghezza), o semplicemente la lunghezza se il simbolo è noto. Tuttavia, se le *run* sono brevi, si può addirittura aumentare la dimensione del file.

Si voglia comprimere la sequenza formata da 29 bit:

00000111001011101110101111111

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 uguali 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.

Questa codifica è usata in CCITT, JBIG2, JPEG, M-JPEG-1, 2, 3, BMP.

Codifica differenziale

Invece di memorizzare i valori assoluti (es. pixel di un'immagine che varia lentamente), si memorizza il primo valore e successivamente solo le **differenze tra un valore e il precedente**. Poiché le differenze tendono a essere piccole, richiedono meno bit (hanno minore entropia).

Esempio:

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

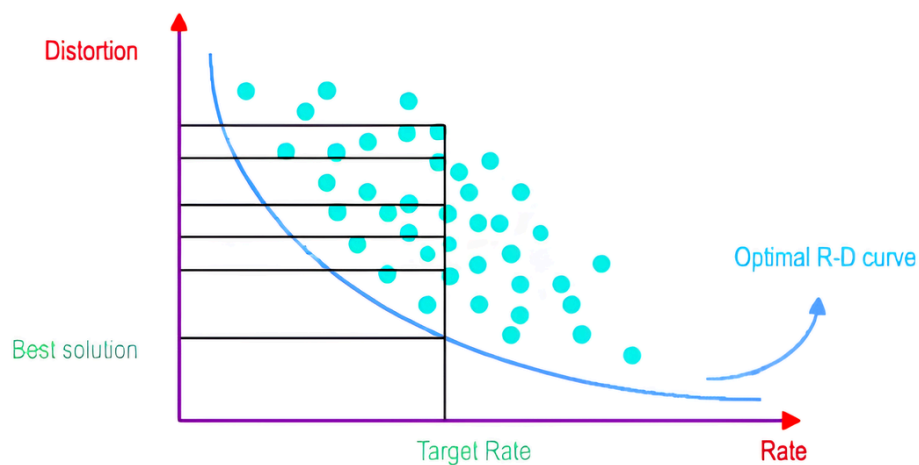
ricorderò il valore iniziale 134 e poi la sequenza delle differenze successive:

-3, +2, 7, -2, -4, 22, ...

Compressione Lossy

La compressione Lossy sfrutta i limiti della **percezione umana**: se un dettaglio è percettivamente irrilevante, viene scartato perdendo dunque alcune informazioni che però non possono essere ricostruite per questo viene definita una compressione **irreversibile**.

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



Algoritmo Requantization

Consiste in una riduzione del numero di livelli disponibili per risparmiare bit per pixel. Si tengono solo i **bit più significativi** per ogni canale, per esempio:

- Rosso: da 8 bit si conservano solo i 4 più significativi;
- Verde: da 8 bit si conservano solo i 6 più significativi;
- Blu: da 8 bit si conservano solo i 2 più significativi.

Risparmiando così il 50% dei bit inizialmente necessari ma perdendo la qualità delle immagini. Se ci sono meno simboli la compressione di Huffman è più efficiente.

(slide 43 per esempio di riferimento)

Lo standard JPEG

Lo standard JPEG (Joint Photographic Experts Group), approvato nel 1992 è il metodo più diffuso per la compressione di immagini fisse. È un metodo di compressione lossy.

Il processo di codifica JPEG si articola in 3 fasi macroscopiche:

- **Pre-processing:**
 - Color transform ($RGB \rightarrow YC_bC_r$);
 - Sottocampionamento della cromaticanza;
 - Suddivisione dell'immagine in sottoimmagini.
- **Trasformazione:**
 - Discrete Cosine Transform;
 - Quantization.
- **Codifica:**
 - Zig-zag ordering of AC coefficients;
 - DC Coefficient Encoding;
 - Entropy Coding (Huffman).

Pre-processing

1. Trasformazione dello Spazio Colore:

Il primo passo consiste nel convertire l'immagine dallo spazio colore RGB allo spazio YC_bC_r . Dove ricordiamo:

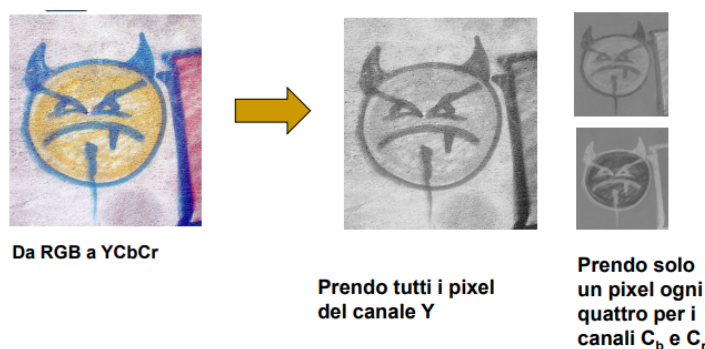
- **Y (luminanza):** rappresenta l'intensità luminosa (l'immagine in scala di grigi);
- **C_bC_r (cromaticanza):** codificano le informazioni sul colore.

Questa trasformazione è reversibile e matematica:

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

2. Sottocampionamento della Crominanza:

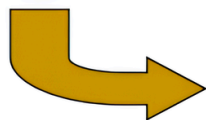
Qui avviene la prima perdita di informazioni. Poiché l'occhio umano è molto più sensibile alle variazioni di luminosità che a quelle di colore, il JPEG mantiene **intatte le informazioni della Luminanza Y** ma **sottocampiona i colori della Crominanza C_bC_r** . Solitamente si sceglie di mantenere un solo valore ogni quattro per le componenti di colore, questo passaggio è irreversibile.



3. Partizione e Shift dei Livelli:

L'immagine viene suddivisa in blocchi, detti **quadrotti**, di dimensioni 8×8 pixel **non sovrapposti**. Questa scelta serve a semplificare i calcoli sfruttando la ridondanza locale, ma è anche la causa del tipico "effetto quadrettatura" visibile nelle immagini molto compresse. Prima di passare alla trasformata, si effettua uno **shift dei livelli di grigio**: a ogni pixel del blocco viene sottratto un valore pari a 2^{n-1} , dove n è la profondità di bit. Per un'immagine a 8 bit (valori 0-255), si sottrae 128 portando il range nell'intervallo $[-128, +127]$ e concentrando il **grigio medio sullo 0**.

52	55	61	66	70	61	64	73
63	59	66	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



-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-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

Trasformazione

1. Trasformazione Discreta del coseno (DCT):

La DCT appartiene alla famiglia delle Trasformate di Fourier e serve per passare dal dominio spaziale al dominio delle frequenze. Matematicamente si considera il blocco di 64 pixel (quadrotto 8×8) come un vettore in uno spazio a **64 dimensioni**. La DCT esprime questo blocco come **somma pesata** di 64 immagini base (funzioni coseno di varie frequenze), invece che come somma di impulsi sui singoli pixel. La DCT opera un cambio di base, proiettando il vettore dei pixel dallo spazio canonico a uno spazio delle frequenze. Statisticamente, questa operazione **decorrela i dati**: trasforma un insieme di pixel altamente dipendenti tra loro in un insieme di coefficienti spettrali tendenzialmente indipendenti, permettendo una maggiore efficienza nelle fasi successive di quantizzazione ed entropy coding.

Le formule della DCT per un blocco $N \times N$, dove $N = 8$ è:

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

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

where :

$$C(u) = \frac{1}{\sqrt{2}} \text{ for } u = 0; C(u) = 1 \text{ otherwise}$$

$$C(v) = \frac{1}{\sqrt{2}} \text{ for } v = 0; C(v) = 1 \text{ otherwise}$$

dove u, v sono le coordinate orizzontali e verticali nel dominio delle frequenze (0 ... 7);

x, y sono le coordinate spaziali dei pixel (0 ... 7);

$f(x, y)$ è il valore del pixel (shiftato) nel blocco originale;

$C(u), C(v)$ sono costanti di normalizzazioni.

Dopo aver applicato la formula otteniamo una nuova matrice 8×8 .

Ma i valori non sono più pixel ma **coefficienti**.

Il primo coefficiente di questa matrice è detto **coefficiente DC** e rappresenta il valore medio della luminanza dell'immagine, i restanti 63 coefficienti sono detti **coefficienti AC** e rappresentano i dettagli e la texture dell'immagine.

-76	-73	-67	-62	-58	-67	-64	-55	-415	-29	-62	25	55	-20	-1	3
-65	-69	-62	-38	-19	-43	-59	-56	7	-21	-62	9	11	-7	-6	6
-66	-69	-60	-15	16	-24	-62	-55	-46	8	77	-25	-30	10	7	-5
-65	-70	-57	-6	26	-22	-58	-59	-50	13	35	-15	-9	6	0	3
-61	-67	-60	-24	-2	-40	-60	-58	11	-8	-13	-2	-1	1	-4	1
-49	-63	-68	-58	-51	-60	-70	-53	-10	1	3	-3	-1	0	2	-1
-43	-57	-64	-69	-73	-67	-63	-45	-4	-1	2	-1	2	-3	1	-2
-41	-49	-59	-60	-63	-52	-50	-34	-1	-1	-1	-2	-1	-1	0	-1

2. Quantizzazione:

Questa è la fase principale in cui avviene la compressione, dunque anche la perdita di qualità. L'obiettivo è ridurre la precisione dei coefficienti DCT per massimizzare l'efficienza della compressione, diminuendo così la lunghezza delle stringhe di bit generate dai **codici di Huffman**.

Dato un **fattore di quantizzazione** Q e il **coefficiente DCT** F , il valore $F_{\text{quantizzato}}$ si ottiene arrotondando all'intero più vicino il rapporto F/Q .

L'effetto immediato di questa operazione è una drastica **riduzione del numero di livelli discreti** su cui i coefficienti possono variare, diminuendo la precisione del dato per aumentarne la comprimibilità. È stato scoperto che non sarebbe efficiente utilizzare un **unico fattore** di quantizzazione per tutti i 64 coefficienti DCT della luminanza o per quantizzare i valori provenienti dalla DCT delle crominanze. Si preferisce adottare per il coefficiente $F(i, j)$ un fattore di quantizzazione $Q(i, j)$ scelto a priori (fornito dallo standard) o scelto dall'utente (in questo caso bisogna trasmettere anche la tabella di quantizzazione). I fattori $Q(i, j)$ costituiscono la **tabella di quantizzazione**. Un esempio di tabelle standard sono:

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

Un fattore di compressione maggiore comporta una maggiore perdita di informazione. Maggiore è il grado di quantizzazione minore sarà la perdita di informazioni, questo grado di quantizzazione detto **quality factor QF** va da 1 a 100.

-415	-29	-62	25	55	-20	-1	3	16	11	10	16	24	40	51	61
7	-21	-62	9	11	-7	-6	6	12	12	14	19	26	58	60	55
-46	8	77	-25	-30	10	7	-5	14	13	16	24	40	57	69	56
-50	13	35	-15	-9	6	0	3	14	17	22	29	51	87	80	62
11	-8	-13	-2	-1	1	-4	1	18	22	37	56	68	109	103	77
-10	1	3	-3	-1	0	2	-1	24	35	55	64	81	104	113	92
-4	-1	2	-1	2	-3	1	-2	49	64	78	87	103	121	120	101
-1	-1	-1	-2	-1	-1	0	-1	72	92	95	98	112	100	103	99

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

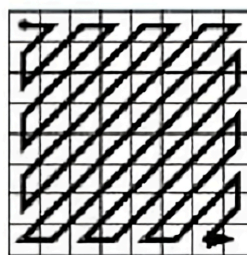
Codifica

L'ultimo passaggio è la codifica dei coefficienti quantizzati per generare il file binario finale.

1. Zig-zag ordering:

64 coefficienti del blocco vengono letti in un **ordine zig-zag** partendo dall'alto a sinistra, facendo ciò è possibile raggruppare i coefficienti nulli (che di solito sono nelle alte frequenze dopo la quantizzazione) in lunghe sequenze consecutive alla fine del vettore, ottimizzando la compressione successiva.

```
-26 -3 -6 2 2 0 0 0
 1 -2 -4 0 0 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
```



ordinamento a zigzag

-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

2. Codifica dei coefficienti DC:

I coefficienti DC tendono ad avere valori simili tra blocchi adiacenti di un'immagine. Per questo motivo vengono codificati usando una **codifica differenziale**. Non si memorizza il valore assoluto ma la differenza Δ rispetto al coefficiente del blocco precedente.

Se consideriamo l'esempio riportato sopra abbiamo come coefficiente DC -2, supponiamo che quello del blocco precedente sia -17. L'evento da codificare sarà $\Delta = -26 - (-17) = -9$. Per trasformare questo valore in bit si utilizza un sistema basato su **categorie (SSSS)** e **codici di Huffman**.

SSSS	Δ
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

Si consulta la tabella standard dove in base al range in cui cade Δ si ottiene una categoria SSSS. Si deve poi trovare la corrispondenza con la 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

Il numero della categoria SSSS rappresenta anche il numero di bit mancanti che occorre aggiungere, nell'esempio di prima $n = SSSS = 4$.

Regola per i bit aggiuntivi:

- $\Delta > 0$: si usano gli n bit meno significativi del valore di Δ in binario;
- $\Delta < 0$: si usano gli n bit meno significativi del valore in binario di Δ (con complemento a due) ai quali occorre sottrarre il valore 1.

3. Codifica coefficienti AC:

Per i coefficienti AC si utilizza la tecnica **RLE** abbinata a Huffman. Poiché la quantizzazione tende ad azzerare molti coefficienti AC, 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. Anche in questo caso per trasformare questo valore in bit si utilizza un sistema basato su **Categorie (SSSS) e codici di Huffman**.

Nell'esempio di prima, tolti il primo coefficiente DC -26, il coefficiente successivo da codificare è -3 e viene rappresentato come (0,-3). Abbiamo trovato che il valore -3 corrisponde alla categoria $SSSS = 2$. L'evento da cercare

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

(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

nella tabella di Huffman è la **combinazione (Run, Categoria)** ovvero (0,2).

Ora bisogna aggiungere i bit mancanti, il numero totale di bit necessari è riportato nell'ultima colonna della tabella.

I bit che completano il codice base sono scelti con lo **stesso criterio** della codifica dei coefficienti DC in base al valore v del coefficiente AC.

Una volta che il flusso di bit è stato ricevuto o letto dal file, il processo deve essere invertito per visualizzare l'immagine. Siamo nella **fase di ricostruzione**. L'obiettivo è tornare indietro dai codici compressi ai valori dei pixel "ricostruendo" i dati originali. Avendo effettuato una compressione lossy, è fondamentale notare che non potremmo ottenere il valore esatto originale ma una sua **approssimazione**. Confrontando i blocchi si può notare la differenza tra l'immagine di input e quella compressa:



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

(guarda le slide 89-94 per vedere gli esempi)