



NoiStudiamo

DISPENSA DI TEORIA



Interazione e Multimedia

Teoria: immagini digitali, colore, elaborazione nel dominio spaziale e delle frequenze, compressione e standard JPEG.

A CURA DI

Gabriele Fontanarosa

Ottobre 2026

Indice dei capitoli

Dall'immagine digitale alla compressione JPEG, passando per colore, filtri e frequenze.

1. Introduzione alle immagini digitali	4
2. Immagini raster e vettoriali, rappresentazione delle immagini raster	7
3. L'occhio umano e la formazione dell'immagine	10
4. L'equazione della lente sottile	13
5. Percezione visiva e illusioni ottiche	16
6. I sensori digitali	19
7. Il Bayer pattern e la color interpolation	22
8. Campionamento e teorema di Shannon	26
9. La quantizzazione	29
10. Aliasing, antialiasing e moiré	33
11. La risoluzione delle immagini digitali	36
12. Interpolazione: replication, bilineare e bicubica	39
13. Il PSNR e la misura della qualità	43
14. Il colore	46
15. Gli spazi di colore RGB e CMY	50
16. Gli spazi di colore HSV e Munsell	53
17. Gli spazi di colore YUV e YCbCr	56

18. Immagini indicizzate, palette e reindexing	59
19. L'istogramma di una immagine	62
20. Le operazioni puntuali e le LUT	65
21. I bit planes	71
22. Operatori lineari e invarianti per traslazione	76
23. Noise reduction	81
24. Edge detection	87
25. Dominio spaziale e dominio delle frequenze	92
26. La trasformata di Fourier e la convoluzione	95
27. Compressione, entropia e teorema di Shannon	103
28. La codifica di Huffman	110
29. Lo standard JPEG	115

Introduzione alle immagini digitali

Che cos'è un'immagine digitale, perché la si elabora e quale modello matematico useremo per descriverla in tutto il corso.

Il ruolo culturale delle immagini

Siamo circondati da immagini. Il linguaggio delle immagini accompagna l'uomo sin dalle sue origini, mentre quello scritto esiste solo da pochi millenni: la comunicazione visuale è per questo la forma di comunicazione più immediata ed efficace. Si dice che «un'immagine vale più di mille parole», e si può aggiungere: «solo che occupa più spazio». Il conto è semplice. Mille parole di testo, a circa sei caratteri l'una, sono circa 6000 byte. Una fotografia da 1600×1200 pixel a colori, memorizzata senza compressione con 3 byte per pixel, ne occupa $1600 \times 1200 \times 3 = 5\,760\,000$, cioè circa 960 volte di più. Una gran parte di questo corso nasce da questa osservazione: come rappresentare, elaborare e comprimere grandi quantità di dati visivi.

L'occhio «vede», ma è il cervello che elabora le immagini. Colore, movimento e profondità diventano dimensioni aggiuntive rispetto all'informazione luminosa che arriva alla retina, e il nostro modo di reagire agli stimoli visivi è in parte soggettivo: per esempio non tutti hanno la stessa sensibilità alle differenze cromatiche, e le persone daltoniche non distinguono alcune combinazioni di colori. Torneremo sulla percezione nei capitoli 3 e 5.

Un po' di storia

La prima fotografia della storia è del 1827: Joseph Nicéphore Niépce (nato nel 1765, morto nel 1833) la ottenne da una finestra della sua casa di Le Gras, in Francia, esponendo per circa otto ore una lastra eliografica. La storia delle immagini *digitali* comincia però con la stampa dei quotidiani.

ANNO	EVENTO
1920	Un'immagine viene trasmessa via cavo tra New York e Londra per comparire su un quotidiano. Il protocollo di trasmissione è specifico per le immagini e il risultato è stampato in <i>halftoning</i> .
1922	Cambia il tipo di stampa: si possono ottenere fino a 5 livelli di grigio.
1929	I livelli di grigio disponibili diventano 15.
1964	Un computer della NASA riceve e elabora un'immagine della Luna, correggendone alcune distorsioni ottiche: è la prima elaborazione al calcolatore di un'immagine digitale.
1975	Steven Sasson (Kodak) costruisce il primo prototipo di fotocamera digitale (capitolo 6).

Tabella 1. Alcune tappe della storia delle immagini digitali.

HALFTONING

È una tecnica di quantizzazione spaziale: permette di riprodurre immagini a toni continui con un dispositivo che sa rappresentare soltanto pochi livelli di intensità (per esempio solo inchiostro o carta bianca), sfruttando il fatto che l'occhio, a distanza, fonde punti piccoli e ravvicinati in una tonalità media.

A cosa serve elaborare un'immagine digitale

Le applicazioni sono moltissime. Alcuni esempi tipici:

- **Dal satellite:** rilevamento del territorio, meteorologia, monitoraggio ambientale, con immagini acquisite in bande dello spettro anche non visibili.
- **In medicina:** radiografie, tomografia assiale computerizzata (TAC), risonanza magnetica, microscopia; l'elaborazione migliora il contrasto e aiuta la diagnosi.
- **Per le forze dell'ordine:** confronto di impronte digitali, lettura di targhe, miglioramento di riprese di videosorveglianza.
- **Per i beni culturali:** restauro virtuale di affreschi e testi antichi, lettura di documenti danneggiati, documentazione e archiviazione.
- **Nella vita quotidiana:** ogni fotocamera di smartphone, ogni fotografia condivisa in rete, ogni video in streaming usa le tecniche di questo corso (interpolazione del colore, filtraggio, compressione JPEG).

In generale gli scopi dell'elaborazione sono quattro: **migliorare** l'aspetto di un'immagine per l'osservatore (*enhancement*), **restaurare** un'immagine degradata, **comprimerla** per ridurne lo spazio e **estrarne informazioni** (per esempio i contorni degli oggetti).

Un'immagine come funzione

Dal punto di vista matematico un'immagine monocromatica è una funzione bidimensionale $f(x, y)$, dove x e y sono le **coordinate spaziali** e il valore $f(x, y)$ è proporzionale alla luce che dal punto raggiunge il sensore. Questa luce dipende da due fattori: quanta luce *incide* sulla scena e quanta ne viene *riflessa* dagli oggetti. Si scrive

$$f(x, y) = i(x, y) \cdot r(x, y)$$

dove i è l'**illuminazione** (luce incidente) e r è la **riflettanza** (frazione di luce riflessa). I due fattori hanno intervalli diversi: $0 < i(x, y) < \infty$, perché la luce incidente può essere arbitrariamente intensa, mentre $0 < r(x, y) < 1$, perché un oggetto non può riflettere più luce di quella che riceve. Un metallo lucido riflette molto, il velluto nero, che assorbe quasi tutta la luce, riflette pochissimo.

GRANDEZZA	ESEMPIO	VALORE TIPICO
Illuminazione	Sole a mezzogiorno	circa 90 000 lm/m ²
Illuminazione	Giornata nuvolosa	circa 10 000 lm/m ²
Illuminazione	Ufficio ben illuminato	circa 1000 lm/m ²
Illuminazione	Luna piena di notte	circa 0,1 lm/m ²
Riflettanza	Velluto nero	0,01
Riflettanza	Acciaio inossidabile	0,65
Riflettanza	Vernice bianca opaca	0,80
Riflettanza	Metallo argentato	0,90
Riflettanza	Neve	0,93

Tabella 2. Ordini di grandezza di illuminazione (in lumen per metro quadrato) e riflettanza (numero puro, senza unità).

Attenzione alle unità: l'illuminazione si misura in lm/m², la riflettanza è un rapporto adimensionale. Quando si dice che «l'argento riflette 0,90» si intende il 90 per cento della luce incidente, non 0,90 lm/m².

Tre modi equivalenti di guardare un'immagine

Un'immagine digitale in toni di grigio si può guardare in tre modi, tutti utili. È una **immagine** da osservare; è una **matrice di numeri** di dimensioni $M \times N$, dove ogni numero è il livello di grigio del pixel corrispondente; è una **superficie** in tre dimensioni, in cui x e y sono le coordinate del pixel e l'altezza z è il valore del grigio (più basso è il valore, più scuro è il pixel). Il passaggio da un'immagine a una matrice è ciò che permette di applicare all'immagine tutta l'algebra lineare.

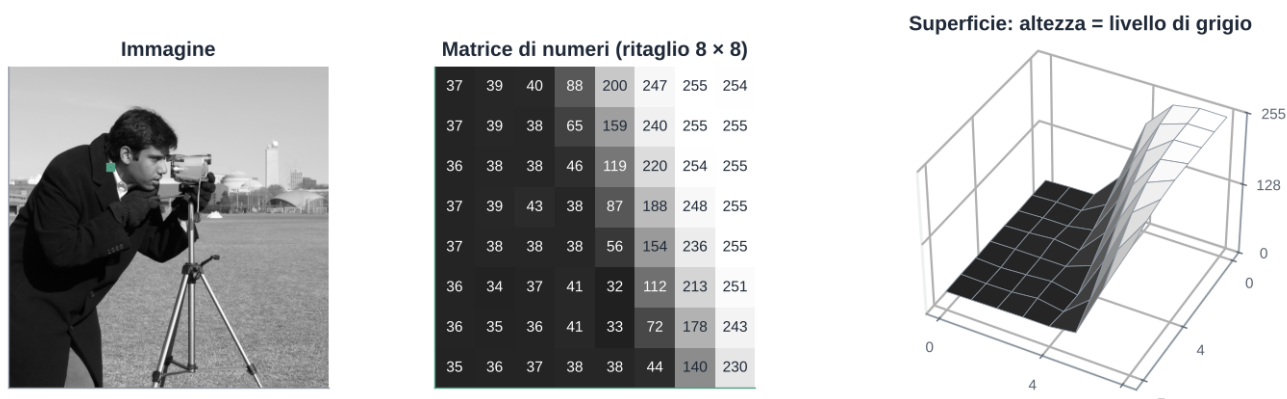


Figura 1. Lo stesso ritaglio 8×8 dell'immagine «cameraman» visto come immagine, come matrice e come superficie.

Dal continuo al discreto

In teoria $f(x, y)$ è una funzione di variabili reali a valori reali. Per rappresentarla in un calcolatore occorrono invece valori discreti. Il passaggio dal continuo al discreto avviene con due operazioni distinte: il **campionamento**, che sceglie un numero finito di punti x, y in cui misurare la funzione (capitolo 8), e la **quantizzazione**, che arrotonda ciascun valore misurato a uno fra un numero finito di livelli (capitolo 9).

Il piano in cui stanno le coordinate dell'immagine si chiama **dominio spaziale**, e le variabili x e y sono dette variabili spaziali o coordinate spaziali. Più avanti vedremo che esiste anche un secondo modo di descrivere la stessa immagine, il **dominio delle frequenze** (capitolo 25), in cui l'immagine è scritta come somma di onde.

UN'IMMAGINE DI PROVA

Per decenni la fotografia di una modella svedese, «Lena» (o «Lenna»), è stata l'immagine di prova più usata dai ricercatori. In questa dispensa, per evitare questioni di diritti, tutti gli esempi sono calcolati su immagini libere o di pubblico dominio: la fotografia «cameraman», la fotografia dell'astronauta della NASA, una veduta della Luna e un'immagine di monete.

DA RICORDARE

Un'immagine digitale è una matrice di numeri ottenuta campionando e quantizzando la funzione $f(x, y) = i(x, y) \cdot r(x, y)$. Elaborare un'immagine significa applicare operazioni a questa matrice: puntuali, locali o globali.

Immagini raster e vettoriali, rappresentazione delle immagini raster

Due modi molto diversi di descrivere un'immagine al calcolatore, e come si memorizza concretamente un'immagine raster in toni di grigio, in bianco e nero e a colori.

Due famiglie di immagini

Un calcolatore può memorizzare un'immagine in due modi. Nella **grafica raster** (o bitmap) l'immagine è una griglia di punti colorati, i **pixel**, e la memoria contiene il colore di ciascun punto: per disegnare una casa si dice, per esempio, «primo pixel bianco, secondo pixel bianco, terzo pixel giallo, ...». Nella **grafica vettoriale** l'immagine è descritta come una serie di forme geometriche: per la stessa casa si dice «dal punto di coordinate (2, 2) al punto (4, 2) c'è una linea gialla; dal punto (4, 2) al punto (4, 4) c'è una linea gialla; ...», e tutto ciò che non è coperto da una forma è lo sfondo, per esempio bianco.

	RASTER	VETTORIALE
Domanda a cui risponde	Quali punti devo colorare?	Quali linee e quali forme devo tracciare?
Che cosa si memorizza	Una matrice di valori, uno per pixel	Un elenco di primitive: segmenti, curve, poligoni, testo, colori
Uso tipico	Fotografie, disegno a mano libera	Disegno tecnico, loghi, caratteri tipografici
Ingrandimento	Compaiono i pixel (scalettatura)	Nessuna perdita: si ricalcolano le forme
Dimensione del file	Cresce con il numero di pixel	Cresce con il numero di forme

Tabella 1. Confronto tra grafica raster e grafica vettoriale.

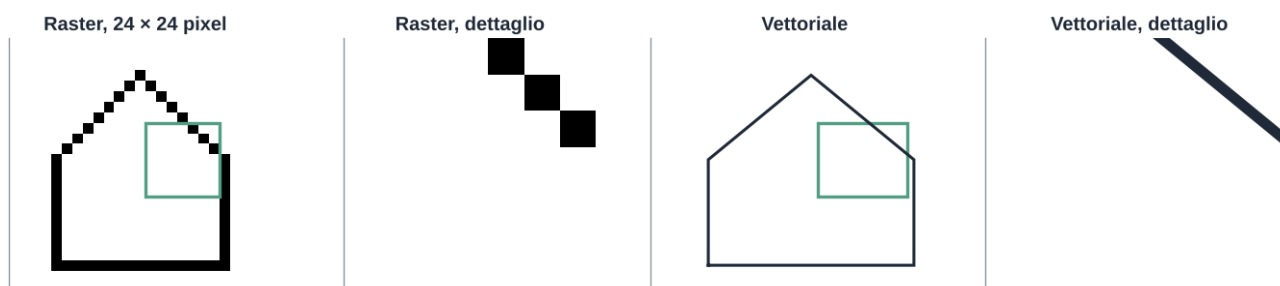


Figura 1. Lo stesso disegno in versione raster (24 × 24 pixel) e vettoriale. Ingrandendo il dettaglio, il raster mostra i pixel e le scalette lungo le diagonali; il vettoriale resta una linea.

	VANTAGGI	SVANTAGGI
Raster	È fotorealistico; è lo standard sul web.	Non ha alcuna descrizione semantica (non «sa» che cosa rappresenta); i file sono grandi; l'ingrandimento mostra i pixel.
Vettoriale	Le trasformazioni sul piano (zoom, scalatura, rotazione) sono semplici e senza perdita.	Non è fotorealistico; molti formati sono proprietari.

Tabella 2. Pro e contro dei due approcci.

La grafica vettoriale è usata dove servono misure precise o scalabilità: nel CAD (Computer Aided Design, per esempio AutoCAD), nell'editoria e nel design (Adobe Illustrator), nel linguaggio di stampa PostScript, nelle animazioni per il web e nei sistemi informativi geografici (GIS). Un'immagine vettoriale può essere stampata con strumenti appositi, i plotter. Alcuni formati vettoriali:

FORMATO	DESCRIZIONE
PS (PostScript)	Sviluppato da Adobe per la stampa di documenti su stampanti laser; usato anche per memorizzare immagini vettoriali.
EPS (Encapsulated PostScript)	Estensione di PostScript che consente di incapsulare anche immagini bitmap.
DCS (Desktop Color Separation)	Caso speciale di EPS sviluppato da Quark per tenere separati i dati ad alta risoluzione dall'anteprima a bassa risoluzione.
PDF (Portable Document Format)	Sviluppato da Adobe, è il formato più diffuso per condividere documenti di testo e immagini indipendentemente dalla piattaforma.
PICT	Formato grafico di Apple per la piattaforma Macintosh, capace di memorizzare immagini vettoriali e raster.

Tabella 3. Un elenco parziale di formati grafici vettoriali.

DAL RASTER AL VETTORIALE

È possibile anche convertire un'immagine raster in vettoriale (*vettorializzazione*) con un parametro di qualità Q : più Q è grande, più l'immagine ricostruita è fedele e più bit per pixel (bpp) servono. Nell'esempio delle slide del corso, il PSNR (capitolo 13) sale da 19,33 dB con $Q = 1$ (0,976 bpp) a 29,32 dB con $Q = 8192$ (53,429 bpp). In questo corso lavoriamo però sempre con immagini **raster**.

La matrice dei pixel

Un'immagine raster è una **matrice** di M righe e N colonne; ogni elemento è un **pixel** (*picture element*) e contiene il valore misurato in quel punto dal sensore, già quantizzato. Valgono queste convenzioni:

- Il primo elemento della matrice è quello **in alto a sinistra**.
- L'asse verticale (le righe) è orientato **dall'alto verso il basso**; l'asse orizzontale (le colonne) da sinistra verso destra. È come il quarto quadrante del piano cartesiano, ma con le coordinate tutte positive.
- Il pixel in riga i e colonna j si indica con $f(i, j)$. Nei testi di matematica e in MATLAB gli indici partono da 1; in Java e in Processing partono da **0** (capitolo 1 della dispensa di laboratorio): attenzione a non confonderli.

Tipologie di immagini

Il numero di bit usati per ogni pixel determina quanti valori diversi esso può assumere: con b bit si hanno 2^b livelli.

TIPO	BIT PER PIXEL	LIVELLI	IL PIXEL CONTIENE
Binaria (bianco e nero)	1	2	0 oppure 1
Scala di grigio	8	256	un valore da 0 (nero) a 255 (bianco)
A colori (RGB)	24 (8 per canale)	$256 \times 256 \times 256 = 16\,777\,216$ colori	una terna (R, G, B), ciascun valore da 0 a 255

Tabella 4. Le tre tipologie fondamentali di immagini raster.

ATTENZIONE AI LIVELLI

Con 1 bit si hanno 2 livelli, non 8. Con 3 bit si hanno $2^3 = 8$ livelli (da 0 a 7). La regola è sempre: livelli = 2 elevato al numero di bit.

Immagini a colori e canali

Un'immagine a colori è costituita da **tre matrici** delle stesse dimensioni, dette **canali**: una per il rosso (R), una per il verde (G) e una per il blu (B). Ogni canale è una matrice di valori da 0 a 255 esattamente come un'immagine in toni di grigio: preso da solo, un canale non è «rosso» o «verde» ma una scala di grigi che dice quanto di quel colore è presente. Quando si visualizza il canale con il suo colore (e gli altri due canali a zero) si parla di *falsi colori*. Il pixel in posizione (i, j) contiene dunque una terna di valori, ciascuno compreso tra 0 e 255.



Figura 2. Un'immagine RGB scomposta nei tre canali: in alto come scale di grigi, in basso ciascun canale con il proprio colore.

Quanta memoria occupa un'immagine

Un'immagine non compressa di $M \times N$ pixel con b bit per pixel occupa

$$M \cdot N \cdot b \text{ bit} = M \cdot N \cdot b / 8 \text{ byte}$$

DIMENSIONI	PIXEL	1 BIT	8 BIT (GRIGI)	24 BIT (RGB)
800 × 600	480 000	60 000 byte	480 000 byte	1 440 000 byte
1600 × 1200	1 920 000	240 000 byte	1 920 000 byte	5 760 000 byte
2400 × 1800	4 320 000	540 000 byte	4 320 000 byte	12 960 000 byte

Tabella 5. Occupazione di memoria di immagini non compresse (1 megabyte = un milione di byte).

Un'osservazione sul conto dei pixel: le slide del corso scrivono che un'immagine «grande» di 2400×1800 pixel ha 4 520 000 pixel; il prodotto esatto è **4 320 000**. Nei calcoli d'esame conviene sempre ripetere la moltiplicazione.

DA RICORDARE

Un'immagine raster è una matrice $M \times N$ di pixel con origine in alto a sinistra. Con b bit per pixel si hanno 2^b livelli; una RGB a 8 bit per canale usa 24 bit per pixel, cioè 3 byte, e distingue circa 16,7 milioni di colori. Nel resto del corso lavoriamo solo con immagini raster.

L'occhio umano e la formazione dell'immagine

Come è fatto l'occhio, quali sono i suoi fotorecettori e come si forma l'immagine sulla retina, a partire dal modello più semplice, il foro stenopeico.

L'occhio e la retina

Dell'occhio ci interessa soprattutto la **retina**, la membrana che ricopre la parte posteriore del bulbo oculare. La luce entra attraverso la cornea, viene focalizzata dal cristallino e forma l'immagine sulla retina, dove è raccolta da due tipi di fotorecettori: i **coni** e i **bastoncelli**.

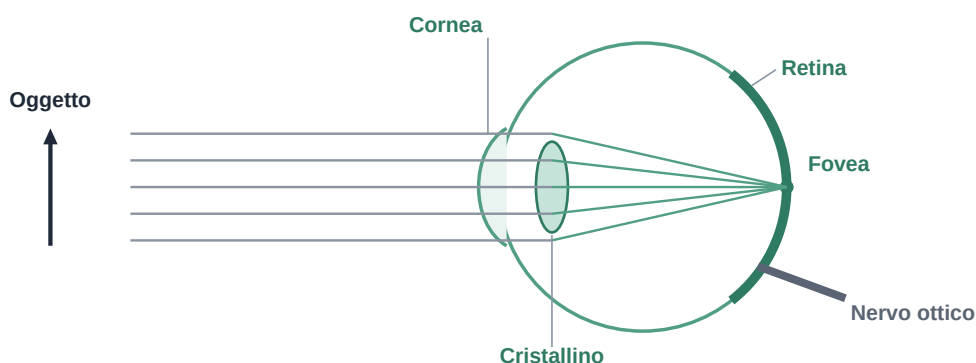


Figura 1. Schema semplificato dell'occhio: la luce attraversa cornea e cristallino e converge sulla fovea.

	CONI	BASTONCELLI
Numero	da 6 a 7 milioni circa	da 75 a 150 milioni circa
Posizione	Concentrati nella parte centrale della retina, la fovea	Distribuiti su tutta la retina
Sensibilità al colore	Fortemente sensibili al colore	Poco sensibili al colore
Collegamento ai nervi	Ogni cono è collegato a un proprio nervo ottico	Più bastoncelli sono collegati in gruppo a un nervo ottico
Tipo di visione	Fotopica (o policroma): luce intensa, colori e dettagli	Scotopica (o monocroma): luce debole, solo chiaro e scuro

Tabella 1. Coni e bastoncelli. Le cifre sono ordini di grandezza.

I coni sono di **tre tipi**, secondo la lunghezza d'onda a cui sono più sensibili: il tipo **S** (*short*, onde corte, colori bluastri), il tipo **M** (*middle*, onde medie, colori verdastri) e il tipo **L** (*long*, onde lunghe, colori rossastri). Tutti i primati hanno questi tre tipi di cellule retiniche per il colore, i mammiferi non primati ne hanno solo due, mentre gli uccelli ne hanno cinque. È la base biologica della **teoria tricromatica** che tratteremo nel capitolo 14.

DA RICORDARE: LA FOVEA

La fovea è una regione di circa $1,5 \text{ mm} \times 1,5 \text{ mm}$ con una densità di circa 150 000 coni per mm^2 . Il numero di coni nella fovea è quindi circa $1,5 \times 1,5 \times 150\,000 = 337\,500$. Un sensore CCD di una fotocamera (capitolo 6) può contenere lo stesso numero di recettori in un'area di non meno di $5 \text{ mm} \times 5 \text{ mm}$.

Il modello del foro stenopeico

Per capire come si forma l'immagine nell'occhio conviene astrarre il problema con il modello del **pinhole** (foro stenopeico). È un modello teorico in cui l'occhio viene approssimato da una scatola: su una parete interna c'è una pellicola sensibile alla luce, la parete opposta è forata da uno spillo. Ogni punto della scena emette (o riflette) raggi in tutte le direzioni, ma solo il raggio che passa per il foro raggiunge la pellicola: a ogni punto della scena corrisponde così un solo punto del piano dei sensori, e l'immagine risulta **capovolta**.

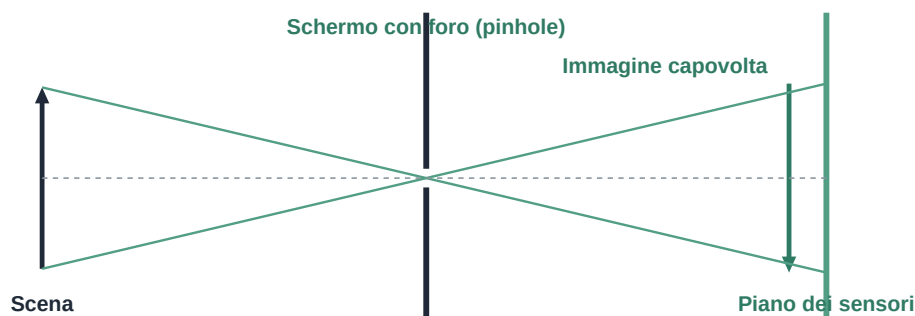


Figura 2. Il modello pinhole: i raggi che passano dal foro formano sul piano dei sensori un'immagine capovolta della scena.

Il raggio del foro più adatto, quello che bilancia quantità di luce raccolta e diffrazione, è proporzionale alla radice quadrata del prodotto tra la distanza d tra foro e piano dei sensori e la lunghezza d'onda λ della luce: $r \propto \sqrt{\lambda \cdot d}$. Per $d = 17 \text{ mm}$ e $\lambda = 550 \text{ nm}$ si ottiene $\sqrt{\lambda \cdot d} = \sqrt{5,5 \cdot 10^{-7} \text{ m} \cdot 1,7 \cdot 10^{-2} \text{ m}} \approx 0,1 \text{ mm}$: un foro adatto ha dimensioni dell'ordine del decimo di millimetro.

Pinhole ideale e pinhole reale

Un foro puntiforme di estensione infinitesima, pinhole **ideale**, farebbe passare troppo pochi fotoni per attivare i sensori. Un foro troppo piccolo, inoltre, genera fenomeni di **diffrazione**. Un pinhole **reale** ha un foro con una precisa forma geometrica, e per questo «spalma» la luce proveniente da un punto infinitesimo (detto *impulso*) su un'area finita. Questa macchia, generata dallo spargersi di un punto luminoso, si chiama **Point Spread Function** (PSF, funzione di diffusione del punto) del sistema di acquisizione.

Vale poi il **principio di sovrapposizione**: l'effetto di più sorgenti puntiformi vicine è la somma degli effetti di ciascuna. L'immagine prodotta da un pinhole reale è quindi la somma delle macchie prodotte da tutti i punti della scena. Questo semplice fatto avrà conseguenze importanti nel filtraggio (capitoli 22 e 26): un sistema che si comporta così si dice *lineare e invariante per traslazione* e il suo effetto sull'immagine è una convoluzione con la PSF.



Figura 3. Quattro sorgenti puntiformi (a sinistra) attraversano un sistema la cui PSF è un piccolo disco (al centro): l'immagine finale è la somma delle quattro macchie (a destra). Calcolata con una convoluzione.

Nella pratica i forellini sono inadeguati: raccolgono troppa poca luce per misure precise. Fin dal Rinascimento, probabilmente osservando la rifrazione in gocce d'acqua o particelle di ghiaccio, si diffuse l'uso di lenti di vetro dalla forma lenticolare, il cui diametro è molto maggiore dello spessore (*lenti sottili*). Hanno proprietà simili al pinhole ma raccolgono molta più luce; le studiamo nel prossimo capitolo.

DA RICORDARE

Il pinhole forma un'immagine capovolta; un foro reale ha una PSF non puntiforme, perciò l'immagine è una somma di macchie sovrapposte (sovrapposizione). Nella retina i coni (S, M, L) servono per il colore e per i dettagli, i bastoncelli per la visione con poca luce.

L'equazione della lente sottile

La relazione che lega la distanza dell'oggetto, quella dell'immagine e la lunghezza focale, con la magnificazione, la messa a fuoco e il comportamento dell'occhio.

Definizione geometrica

Una **lente sottile** è un modello ideale di lente il cui spessore è trascurabile rispetto alla lunghezza focale e alle distanze in gioco: tutta la rifrazione avviene in un unico piano, il *piano della lente*, e un raggio che passa per il centro ottico non viene deviato. Si può definire con tre proprietà geometriche, che sono due parti «speculari» l'una dell'altra:

1. i raggi paralleli all'asse della lente vengono concentrati in un unico punto, il **fuoco**, posto a distanza f dalla lente (la **lunghezza focale**);
2. i raggi che partono dal fuoco vengono ritrasmessi tutti paralleli all'asse della lente;
3. una lente sottile ha due fuochi, equidistanti dalla lente, uno per lato.

Una lente **convergente** (convessa) fa convergere la luce in un punto: è quella delle lenti di ingrandimento e degli occhiali per ipermetropi. Una lente **divergente** (concava) fa divergere i raggi ed è usata, per esempio, negli occhiali per miopi. Qui trattiamo la lente convergente, che è quella dell'obiettivo e del cristallino.

Un oggetto puntiforme emette raggi in tutte le direzioni. Di questi, uno è parallelo all'asse ottico e la lente lo fa passare per il fuoco; un altro passa per il fuoco davanti alla lente e ne esce parallelo all'asse; un terzo passa per il centro e non è deviato. Il punto in cui i raggi si incontrano è il punto di formazione dell'**immagine**: si può dimostrare che, per una lente ideale, vi si incontrano anche tutti gli altri raggi. Se il piano dei sensori è posto più avanti o più indietro del piano che contiene l'immagine, si ottiene un'immagine **sfocata** dell'oggetto.

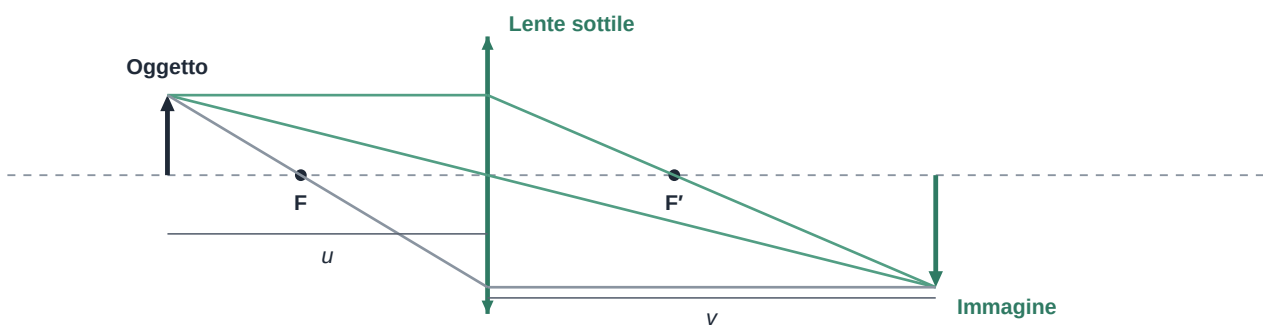


Figura 1. Formazione dell'immagine in una lente sottile con $f = 70$, $u = 120$: l'immagine si forma a $v = 168$ ed è capovolta e rimpicciolita. I tre raggi principali si incontrano nello stesso punto.

Ricavare l'equazione

Sia u la distanza dell'oggetto dalla lente, v la distanza dell'immagine dalla lente, f la lunghezza focale. Indichiamo con a l'altezza dell'oggetto e con b l'altezza dell'immagine (positiva se capovolta). Il raggio che passa per il centro non viene deviato: i due triangoli rettangoli con vertice nel centro della lente, uno con base u e altezza a , l'altro con base v e altezza b , sono simili, perciò

$$b / a = v / u$$

Il raggio parallelo all'asse, uscendo dalla lente, passa per il fuoco di destra e arriva nel punto dell'immagine. Il triangolo con vertice nel fuoco, di base f e altezza a (l'altezza a a cui il raggio parallelo incontra la lente), è simile al triangolo di base $v - f$ e altezza b :

$$b/a = (v - f) / f$$

Uguagliando le due espressioni di b/a si ha $v/u = (v - f)/f$, cioè $vf = uv - uf$. Dividendo ogni termine per uvf si ottiene

$$1/u + 1/v = 1/f$$

che è l'**equazione della lente sottile**. Se f si misura in metri, la grandezza $1/f$ si misura in **diottrie** (1 diottria = 1 m^{-1}).

CONSEGUENZE IMPORTANTI

- a) In una lente fissa f è costante: se la distanza u dell'oggetto cresce, v deve diminuire. Per questo la messa a fuoco di una fotocamera richiede di avvicinare o allontanare il piano dei sensori dalla lente.
- b) Se il piano dei sensori non può essere mosso, come la retina umana, l'unica possibilità è variare la lunghezza focale: lo fanno i muscoli che mantengono in tensione il cristallino, cambiandone la curvatura. Questo processo si chiama **accomodazione** e si misura in diottrie.
- c) Due oggetti a distanze u_1 e u_2 molto maggiori di f formano le loro immagini praticamente sullo stesso piano e sono a fuoco insieme. Se le distanze sono invece diverse e confrontabili (entro circa 30 volte la distanza della lente), non possono essere messi a fuoco contemporaneamente: si parla di **profondità di campo**, che è tanto più ridotta quanto più f è grande.

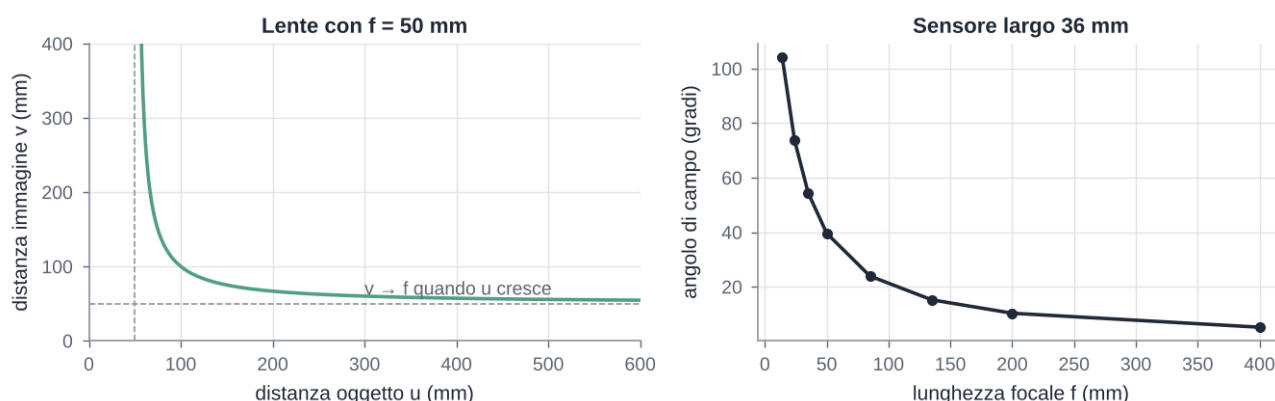


Figura 2. A sinistra: la distanza dell'immagine v in funzione di u per $f = 50 \text{ mm}$; per u grande v si avvicina a f . A destra: angolo di campo orizzontale per un sensore largo 36 mm , $2 \cdot \arctan(36/(2f))$.

L'occhio come lente

Nell'occhio il piano dei sensori, la retina, è a distanza fissa dal cristallino, circa 17 mm . Per un oggetto lontanissimo ($u \rightarrow \infty$) l'equazione dà $f = v = 17 \text{ mm}$, cioè $1/0,017 \approx 58,8$ diottrie. Per mettere a fuoco un oggetto a 25 cm , $1/f = 1/0,25 + 1/0,017 \approx 4 + 58,8 = 62,8$ diottrie: il cristallino deve quindi aumentare il proprio potere di circa 4 diottrie. È un modello semplificato (l'occhio reale ha anche la cornea e mezzi con indice di rifrazione diverso dall'aria), ma spiega l'ordine di grandezza.

Esempio: dimensione dell'immagine sulla retina. Un albero alto 15 m a 100 m di distanza, con la retina a 17 mm dal cristallino, forma sulla retina un'immagine di altezza h tale che $15/100 = h/17$, da cui $h = 2,55 \text{ mm}$.

Magnificazione

Il rapporto tra l'altezza dell'immagine e quella dell'oggetto è il **fattore di magnificazione** $m = h/k = v/u$ (in modulo). Dalla relazione $1/u + 1/v = 1/f$, sostituendo $v = mu$, si ha $1/u + 1/(mu) = (m + 1)/(mu) = 1/f$, cioè

$$f = u \cdot m / (m + 1)$$

Questa relazione serve quando si vuole fissare la focale per garantire una certa magnificazione.

Esempio 1. Un ragno a $u = 1$ cm è fotografato da una macchina digitale con il CCD a $v = 0,5$ cm: $m = 0,5/1 = 0,5$. A quale focale occorre fissare l'obiettivo per ottenere lo stesso effetto su una farfalla a $u' = 500$ cm? $f = 500 \cdot 0,5/1,5 \approx 166,7$ cm. Un teleobiettivo con focale di oltre un metro e mezzo non è comodo da portare con sé.

Esempio 2. Stesso oggetto a $u = 1$ cm, ma con una macchina analogica con $v = 2$ cm; la farfalla è a $u' = 5$ cm. Allora $m = 2/1 = 2$ e $f = 5 \cdot 2/3 \approx 3,3$ cm.

Esempio 3. Oggetto a $u = 1$ cm, macchina digitale con CCD a $v = 1$ cm; la farfalla è a $u' = 5$ cm. Allora $m = 1$ e $f = 5 \cdot 1/2 = 2,5$ cm.

Ampiezza di campo e focale

La focale determina anche quanta scena entra nell'immagine. Con la stessa area di sensori, una focale piccola (grandangolo) inquadra un campo ampio, una focale grande (teleobiettivo) inquadra un campo ristretto e ingrandisce i dettagli. Per un sensore largo s l'angolo di campo orizzontale è $2 \cdot \arctan(s/(2f))$. Con $s = 36$ mm: $f = 24$ mm dà circa $73,7^\circ$, $f = 50$ mm circa $39,6^\circ$, $f = 200$ mm circa $10,3^\circ$. A parità di focale, un sensore più piccolo riduce il campo, come mostrano le due figure delle slide in cui a focale piccola o grande si affianca un'area dei sensori piccola o grande.

Gli obiettivi reali combinano più lenti per correggere i difetti di una lente singola; ai fini del corso basta il modello della lente sottile.

DA RICORDARE

$1/u + 1/v = 1/f$; magnificazione $m = v/u$; $f = um/(m + 1)$. Nell'occhio v è fisso e si adatta f (accomodazione); in una fotocamera f è fissa e si muove il sensore (messa a fuoco).

Percezione visiva e illusioni ottiche

L'occhio misura, ma è il cervello a interpretare: leggi della percezione, intensità percepita, intervallo dinamico e le illusioni ottiche più note.

Le leggi della percezione visiva

La psicologia della Gestalt ha descritto alcune regole con cui il sistema visivo raggruppa e completa gli stimoli. Le principali sono cinque.

LEGGE	ENUNCIATO
Prossimità	Gli elementi più vicini tra loro tendono a essere percepiti come un gruppo unico.
Chiusura	Le figure chiuse si vedono più facilmente di quelle aperte e tendono a prevalere su di esse; si completano mentalmente i contorni interrotti.
Somiglianza (uguaglianza)	Elementi uguali o simili vengono percepiti insieme: punti neri e bianchi alternati sono visti come file orizzontali di punti dello stesso colore.
Continuità	Punti disposti uno di seguito all'altro in una direzione sono percepiti come una struttura unitaria: due curve che si incrociano sono viste come due linee complete, non come quattro pezzi.
Buona forma	La mente tende a vedere forme semplici, regolari e complete, anche dove il contorno non è disegnato.

Tabella 1. Le leggi principali della percezione visiva.



Figura 1. Dimostrazioni delle cinque leggi. Nell'ultima si percepisce un triangolo (illusorio) anche se i suoi lati non sono disegnati.

Completare l'informazione mancante: inpainting

Il cervello «integra» l'informazione mancante o corrotta. Questa capacità ha un corrispondente in elaborazione di immagini: l'**inpainting**, cioè il riempimento di una *lacuna* (un graffio, un oggetto da rimuovere) usando l'informazione che sta intorno. L'algoritmo riconosce la zona da ricostruire e la riempie in modo statistico o per propagazione dei bordi circostanti; il risultato è plausibile ma, ovviamente, **non** è il contenuto originale perduto.



Figura 2. Due graffi bianchi su una fotografia (al centro) rimossi con un algoritmo di inpainting (a destra).

Intensità percepita

Le immagini digitali hanno un numero finito di livelli di intensità, quindi è importante sapere con quale finezza l'occhio distingue tra livelli diversi. Sperimentalmente si trova che la luminosità percepita è funzione **logaritmica** dell'intensità che arriva all'occhio (legge di Weber e Fechner): ciò che l'occhio nota è soprattutto il *rapporto* tra due intensità, non la differenza. Di conseguenza due toni scuri vicini sono distinti con più facilità di due toni chiari separati dalla stessa differenza assoluta. Questo spiega due scelte tecniche che incontreremo: la quantizzazione logaritmica, che dedica più livelli ai toni scuri (capitolo 9), e la correzione gamma (capitolo 20).

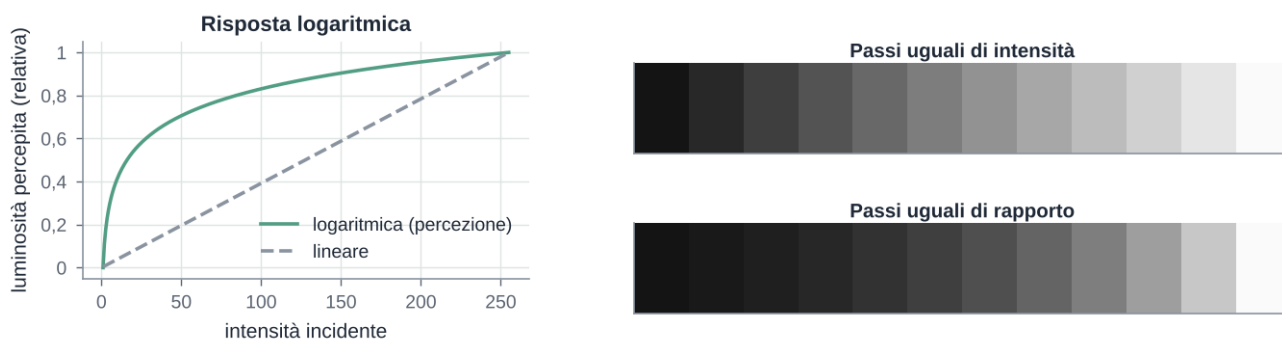


Figura 3. A sinistra la risposta logaritmica a confronto con quella lineare; a destra due rampe di grigio: nella seconda, con rapporti costanti tra toni successivi, i passi appaiono più regolari.

Intervallo dinamico e adattamento

Il sistema visivo non lavora contemporaneamente su tutto l'intervallo di intensità che sa percepire, che si estende per molti ordini di grandezza (dell'ordine di 10^{10} tra la soglia della visione scotopica e il limite dell'abbagliamento), ma solo su una porzione di esso, centrata sul **livello di adattamento** corrente. Se la stanza è buia e si apre una finestra, il paesaggio esterno abbaglia e non si vede nulla; se invece nella stanza c'è molta luce, l'occhio si adatta al chiaro e riesce a distinguere i dettagli all'esterno. Per questo le fotocamere compensano facendo più scatti con esposizioni diverse e combinandoli (tecnica *HDR*, High Dynamic Range). Inoltre l'occhio distingue in modo diverso nelle zone chiare e in quelle scure.

Illusioni ottiche

Esistono fenomeni, tipici del sistema visivo umano, in cui la luminosità percepita **non** dipende soltanto dall'intensità emessa. Due sono classici.

Bande di Mach (Ernst Mach, 1865). Una scala di bande, ciascuna di intensità costante, viene percepita in modo non uniforme in prossimità dei bordi: ogni banda appare un po' più scura dal lato verso la banda più chiara, e un po' più chiara dal lato verso la banda più scura, così che i bordi appaiono accentuati. La spiegazione oggi più accettata è l'*inibizione laterale* nella retina: ogni cellula è eccitata dalla luce nel proprio punto e inibita dalla luce nei punti vicini, quindi risponde alla differenza tra centro e contorno. La curva verde della figura è calcolata con questo

modello (centro meno contorno) e riproduce gli «sbalzi» ai bordi.

Contrasto simultaneo. Quadrati centrali con la stessa intensità vengono percepiti diversamente a seconda dello sfondo: se lo sfondo è scuro il centro appare più chiaro, se lo sfondo è chiaro il centro appare più scuro.

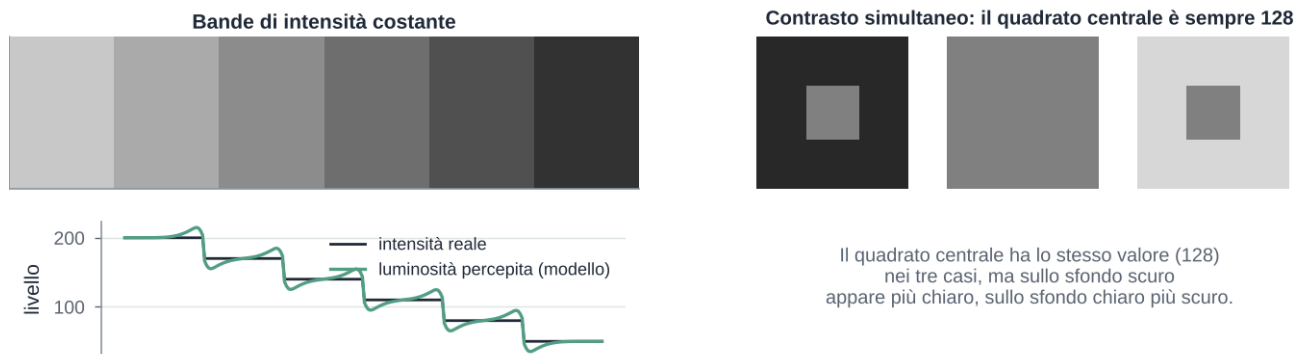


Figura 4. A sinistra le bande di Mach e il profilo percepito calcolato con un modello a inibizione laterale. A destra il contrasto simultaneo.

PERCHÉ CI INTERESSA

Le illusioni mostrano che misure oggettive (come il PSNR, capitolo 13) e percezione non coincidono. Mostrano anche che i bordi sono molto importanti per la visione, tema che ritroveremo con l'estrazione dei contorni (capitolo 24) e con la compressione, che elimina soprattutto ciò che l'occhio non coglie (capitolo 27).

DA RICORDARE

La percezione dell'intensità è logaritmica; l'occhio opera su una finestra dell'intervallo dinamico attorno al livello di adattamento; la luminosità percepita dipende dal contesto (bande di Mach, contrasto simultaneo); il cervello completa le informazioni mancanti (leggi della Gestalt, inpainting).

I sensori digitali

Come la luce riflessa dalla scena diventa un impulso elettrico e poi un numero: sensori singoli, in linea e a matrice, il CCD e la nascita della fotocamera digitale.

Dalla luce al numero

Quando la luce colpisce un oggetto, una parte viene assorbita e una parte viene riflessa. Quella che viene riflessa dà origine al colore percepito. Per creare un'immagine digitale è essenziale che la luce riflessa sia catturata da un **sensore** ed elaborata. Il sensore è fatto di un materiale particolarmente sensibile alla luce: l'energia che lo colpisce viene trasformata in un impulso elettrico, che successivamente viene **digitalizzato**.

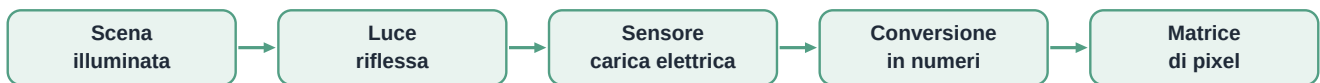


Figura 1. La catena di acquisizione: dalla scena alla matrice di numeri.

Nei due passaggi finali si ritrovano le due operazioni già incontrate nel capitolo 1: la disposizione geometrica delle celle del sensore realizza il **campionamento** (capitolo 8), mentre la conversione della carica in un numero intero realizza la **quantizzazione** (capitolo 9).

Tre geometrie di sensore

Il modo in cui i sensori sono disposti determina come si acquisisce l'immagine e quanto meccanica è la scansione.

- **Singolo sensore.** Gli scanner più semplici usano un solo sensore che viene spostato lungo la sorgente da digitalizzare, punto per punto e riga per riga.
- **Sensori in linea.** Le macchine per la tomografia assiale computerizzata (TAC) usano sensori disposti in un'unica linea. Lo spostamento avviene in una sola direzione: ad ogni passo si acquisisce una riga intera.
- **Sensori in matrice (2D array).** Nelle macchine fotografiche digitali i sensori sono disposti su una matrice. Non serve spostare il sensore per effettuare una scansione: l'intera immagine è acquisita in un'unica esposizione.

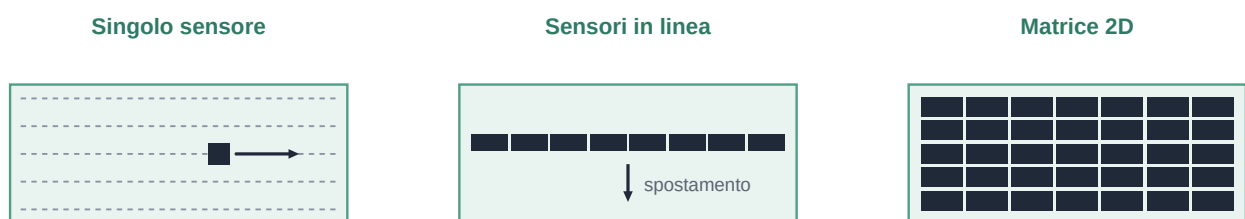


Figura 2. Le tre geometrie di sensore. La freccia indica il movimento meccanico necessario per coprire la scena.

I più diffusi sensori a matrice sono i **CCD**, oggi affiancati e in gran parte sostituiti dai sensori CMOS.

Il CCD

Il CCD (*Charge Coupled Device*, dispositivo ad accoppiamento di carica) è un dispositivo elettronico formato da una matrice di celle. Quando i fotoni colpiscono una cella, per effetto fotoelettrico generano al suo interno cariche elettriche: più la luce è intensa e più a lungo dura l'esposizione, maggiore è la carica accumulata. Le slide del corso parlano di cella che «assume una carica positiva»; il segno è una questione di convenzione, ciò che conta è che la carica sia **proporzionale alla luce ricevuta**.

Le celle non possono però caricarsi oltre un certo limite: sono come **secchi d'acqua** che non possono riempirsi oltre la loro capacità. Se la luce è troppa, il secchio trabocca e la cella registra comunque il valore massimo: è il fenomeno della **sovrassaturazione**. Nell'immagine finale le zone troppo illuminate diventano bianco pieno e il dettaglio al loro interno è perso in modo irreversibile.



Figura 3. A sinistra la risposta di una cella, lineare fino alla capacità e poi costante. A destra la stessa scena con esposizione crescente: nelle zone chiare la cella satura e il dettaglio scompare.

MEGAPIXEL

Il numero di celle per area di esposizione è un parametro di qualità della fotocamera, e si misura in **megapixel** (milioni di pixel). Un sensore di 4000×3000 celle contiene $4000 \times 3000 = 12\,000\,000$ di pixel, cioè 12 megapixel. Attenzione: più celle non significano automaticamente immagini migliori, perché a parità di area le celle diventano più piccole e raccolgono meno luce ciascuna.

Lo schema di misurazione del CCD

Dopo che le cariche sono state acquisite da una matrice di celle, devono essere trasferite in una memoria digitale. Secondo lo schema descritto a lezione, la scansione avviene in C fasi, una per ciascuna colonna della matrice. In ogni fase:

- la **prima colonna** viene trasferita in memoria, con un trasferimento verticale verso la memoria;
- nello stesso tempo tutti gli elementi dalla seconda colonna in poi vengono trasferiti dalla propria colonna a quella precedente, con un trasferimento orizzontale verso la prima colonna.

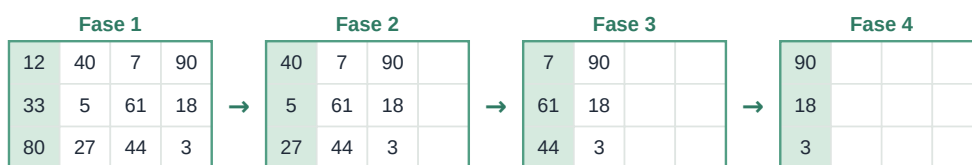


Figura 4. Lettura di un CCD con 3 righe e 4 colonne: a ogni fase la colonna evidenziata viene portata in memoria e le altre scorrono verso sinistra. Dopo 4 fasi tutte le 12 celle sono state lette.

In una matrice con C colonne servono dunque C fasi. Nei testi si trova spesso la stessa descrizione scambiando il ruolo di righe e colonne: il principio, cioè far scorrere le cariche cella per cella fino a un punto di uscita, non cambia.

CCD E CMOS

Nei sensori CMOS, oggi i più diffusi, ogni cella ha un proprio amplificatore e può essere letta direttamente indirizzandola per riga e colonna, senza far scorrere le cariche. Il principio di acquisizione (luce, carica, conversione in numero) è lo stesso.

La prima fotocamera digitale

Nel dicembre 1975 Steven Sasson, ingegnere elettrico alla Eastman Kodak di Rochester (New York), fu la prima persona a scattare una fotografia con una fotocamera digitale. Il prototipo aveva queste caratteristiche:

CARATTERISTICA	VALORE
Peso	4 kg
Risoluzione	$100 \times 100 \text{ pixel} = 10\,000 \text{ pixel} = 0,01 \text{ megapixel}$
Memorizzazione	su audiocassetta
Tempo per registrare uno scatto	23 secondi
Tempo per visualizzarlo su un televisore	altri 23 secondi

Tabella 1. Il prototipo di Sasson del 1975.

Il test finale fu fatto facendo posare un assistente. Il primo risultato lasciava intravedere solo la silhouette dei capelli; dopo una piccola taratura dell'elettronica il viso fu subito visibile. Un confronto dà l'idea dei progressi: una fotocamera da 12 megapixel contiene $12 / 0,01 = 1200$ volte più pixel del prototipo del 1975.

DA RICORDARE

Un sensore trasforma la luce in carica elettrica proporzionale all'esposizione; la disposizione delle celle campiona la scena, la conversione in numeri la quantizza. Singolo sensore e sensori in linea richiedono uno spostamento meccanico, i sensori a matrice (CCD) no. Le celle hanno una capacità massima: oltre di essa il valore satura.

Il Bayer pattern e la color interpolation

Perché ogni cella del sensore registra un solo colore, come è fatta la griglia di Bayer e come si ricostruiscono i due colori mancanti in ogni pixel.

Un solo colore per cella

Una cella del sensore misura soltanto la quantità di luce che la raggiunge: non sa di che colore sia. Per ottenere un'immagine a colori, davanti a ogni cella si pone un minuscolo **filtro colorato** che lascia passare solo una banda dello spettro, per esempio il rosso. L'insieme dei filtri si chiama **CFA** (*Color Filter Array*, matrice di filtri colorati). Poiché ogni cella memorizza un solo colore alla volta e non una terna, occorre scegliere qual è il modello di memorizzazione ottimale. I due colori mancanti per completare la terna di ciascun pixel saranno ottenuti per **interpolazione** dai pixel vicini: questa fase si chiama *color interpolation*. Il grado di accuratezza del risultato dipende da quanto è sofisticato il metodo di interpolazione.

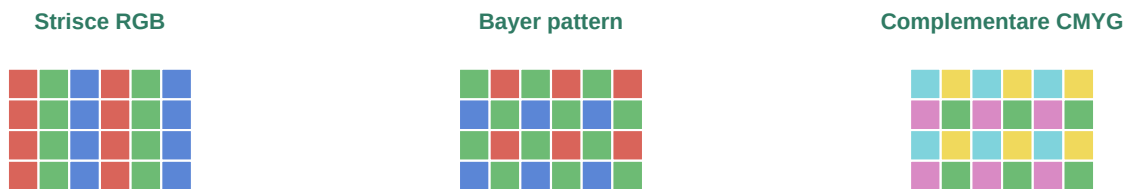


Figura 1. Tre possibili modelli di CFA: strisce RGB, Bayer pattern e un modello a colori complementari (ciano, giallo, magenta, verde) usato in alcuni sensori.

Il Bayer pattern

Tra i modelli possibili il più usato è il **Bayer pattern**, proposto nel 1976 da Bryce Bayer, ricercatore della Kodak (nelle slide il nome è scritto «Byce»). Da allora è la scelta di quasi tutte le fotocamere e i dispositivi elettronici che acquisiscono immagini a colori. È una griglia di filtri in cui il motivo si ripete in blocchi 2×2 :

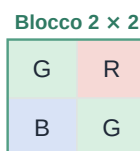


Figura 2. Il blocco elementare del Bayer pattern: due filtri verdi sulla diagonale, uno rosso e uno blu.

- Il rapporto tra i filtri è **1:2:1** per R:G:B: in ogni blocco ci sono un rosso, due verdi e un blu, quindi il 25 per cento dei pixel è rosso, il 50 per cento verde e il 25 per cento blu.
- I pixel verdi sono disposti sulle **diagonali** del blocco.
- Si privilegiano le misure nel canale verde perché è quello più importante per la percezione umana: la luminosità, che è la componente che l'occhio risolve meglio (capitolo 3), dipende soprattutto dal verde.

Una immagine conservata come matrice di valori del sensore, cioè con un solo valore per pixel secondo il Bayer pattern, è nel formato **raw**. Ogni produttore ha un proprio formato raw proprietario.

PRODUTTORE	FORMATO	PRODUTTORE	FORMATO
Canon	CRW, CR2	Olympus	ORF
Epson	ERW	Pentax	PEF
Foveon	X3F	Sony	ARW
Fuji	RAF	Samsung	SRW
Hasselblad	3FR	Nikon	NEF
Kodak	DCR	Minolta	MRW

Tabella 1. Alcuni formati raw proprietari.

Che cosa manca in ogni pixel

Con il blocco G R / B G ogni pixel ha un solo valore misurato e due da ricostruire. Se un pixel è verde, mancano il rosso e il blu; se è rosso, mancano il verde e il blu; se è blu, mancano il rosso e il verde.

PIXEL MISURATO	VALORE NOTO	DA RICOSTRUIRE	VICINI DISPONIBILI
Rosso	R	G e B	4 verdi in croce (su, giù, sinistra, destra), 4 blu in diagonale
Verde, su riga di rossi	G	R e B	2 rossi (sinistra e destra), 2 blu (sopra e sotto)
Verde, su riga di blu	G	R e B	2 rossi (sopra e sotto), 2 blu (sinistra e destra)
Blu	B	R e G	4 rossi in diagonale, 4 verdi in croce

Tabella 2. Valori mancanti e vicini utilizzabili per ciascun tipo di pixel.

Dal mosaico all'immagine: il demosaicking

La matrice di dati rilasciata dal sensore può essere visualizzata come immagine: in questo caso si vede una immagine in scala di grigi. Poiché si è conservata una sola componente della terna, se si colora ogni pixel con il solo colore del suo filtro (immagine a *falsi colori*) l'immagine appare composta da un **mosaico**. Per ottenere un'immagine a colori occorre un algoritmo che ricavi i valori mancanti di ogni terna dai dati dell'intorno. Poiché dall'immagine a colori sparisce l'effetto a mosaico, l'algoritmo di color interpolation viene anche chiamato **demosaicking**.

Color interpolation per replication

Nella *replication*, detta anche interpolazione al **vicino più prossimo** (*nearest neighbor*), per ogni pixel i valori mancanti della terna vengono semplicemente **copiati** dall'intorno. È il metodo più rapido ma anche il più grossolano, perché copia valori che si riferiscono a punti diversi della scena.

Esempio. Dato il blocco di valori del sensore

Dati del sensore		Filtri	
120	240	G	R
0	255	B	G

Figura 3. Valori del sensore e relativi filtri (indici di riga e colonna a partire da 1).

si vuole la terna (R, G, B) nel punto (2, 2) con la replication. Il punto (2, 2) corrisponde a un pixel verde, quindi il verde è già noto e vale 255. Per il rosso e il blu si usano i valori più vicini nel blocco: il rosso del pixel (1, 2) vale 240 e il blu del pixel (2, 1) vale 0. La terna cercata è dunque **(R, G, B) = (240, 255, 0)**.

Nel pixel (1, 1), anch'esso verde, la terna è (240, 120, 0) con lo stesso procedimento. Per un pixel rosso o blu manca invece un verde e nel blocco ce ne sono due: occorre fissare una convenzione. Qui si copia il verde in alto a sinistra del blocco; con un'altra convenzione i risultati cambiano leggermente, ma il livello di qualità è lo stesso.

Color interpolation bilineare

Nell'interpolazione **bilineare** i valori mancanti si calcolano come **media** dei valori misurati più vicini dello stesso colore. Nella matrice del colore che il pixel già possiede non si fa nulla. Per gli altri due colori:

- pixel **rosso**: il verde è la media dei 4 verdi in croce, il blu è la media dei 4 blu in diagonale;
- pixel **blu**: il rosso è la media dei 4 rossi in diagonale, il verde è la media dei 4 verdi in croce;
- pixel **verde**: il rosso e il blu sono la media di 2 valori ciascuno, i due vicini orizzontali per un colore e i due verticali per l'altro.

Esempio. Un ritaglio 5×5 di dati del sensore, con lo stesso blocco G R / B G che ricomincia in alto a sinistra:

Dati del sensore				
90	200	94	210	98
40	96	44	100	48
100	220	104	230	108
50	104	54	108	58
110	240	114	250	118

Figura 4. Ritaglio 5×5 del sensore. In evidenza i quattro pixel centrali, uno per tipo, con indici di riga e colonna a partire da 0.

Pixel rosso in (2, 1), valore 220. Il verde è la media dei quattro vicini in croce: $(100 + 104 + 96 + 104) / 4 = 101$. Il blu è la media dei quattro vicini diagonali: $(40 + 44 + 50 + 54) / 4 = 47$. Terna **(220, 101, 47)**.

Pixel blu in (3, 2), valore 54. Il rosso è la media dei quattro vicini diagonali: $(220 + 230 + 240 + 250) / 4 = 235$. Il verde è la media dei quattro vicini in croce: $(104 + 114 + 104 + 108) / 4 = 107,5$. Terna **(235, 107,5, 54)**.

Pixel verde in (2, 2), valore 104, su una riga di rossi. Il rosso è la media dei vicini a sinistra e a destra, $(220 + 230) / 2 = 225$; il blu è la media dei vicini sopra e sotto, $(44 + 54) / 2 = 49$. Terna **(225, 104, 49)**. Nel pixel verde in (3, 3), su una riga di blu, i ruoli si scambiano: rosso $(230 + 250) / 2 = 240$ e blu $(54 + 58) / 2 = 56$, terna **(240, 108, 56)**.

Come si vede, in ciascun caso si calcola la media di soli 2 o 4 valori. L'interpolazione bilineare produce risultati migliori della replication con un incremento modesto della complessità di calcolo. Metodi più sofisticati usano 16 pixel (interpolazione bicubica, capitolo 12) oppure tengono conto della direzione dei bordi.

Confronto dei risultati

Per confrontare i metodi si parte da una fotografia a colori, si simula il sensore tenendo di ogni pixel un solo valore secondo il Bayer pattern e si ricostruisce l'immagine con i due metodi. Sull'intera fotografia dell'astronauta (512×512 pixel) il confronto con l'originale, misurato con il PSNR (capitolo 13), dà 25,69 dB per la replication e 30,64 dB per l'interpolazione bilineare. Un valore più alto indica una ricostruzione più fedele. La replication introduce sui bordi evidenti frange colorate, perché i canali sono ricostruiti con valori spostati di un pixel; la bilineare le attenua ma non le elimina del tutto.

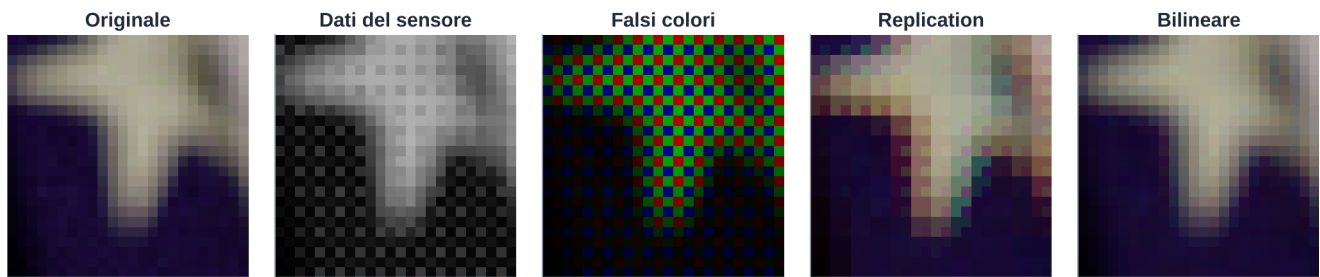


Figura 5. Un ritaglio di 24×24 pixel (una stella della bandiera) della fotografia dell'astronauta: l'originale, la matrice del sensore vista come immagine in grigi, la stessa matrice a falsi colori e le due ricostruzioni.

DA RICORDARE

Ogni pixel del sensore registra un solo colore. Il Bayer pattern ha rapporto R:G:B = 1:2:1, con i verdi sulle diagonali, perché l'occhio è più sensibile al verde. Il demosaicking ricostruisce i due colori mancanti: per replication si copiano i valori vicini, per interpolazione bilineare se ne fa la media.

Campionamento e teorema di Shannon

Come si passa da un segnale continuo a un numero finito di campioni, quanti ne servono e perché esiste un limite sotto il quale il segnale non si può più ricostruire.

Dal segnale continuo ai campioni

Il **campionamento** è il processo con cui un segnale continuo (un suono, una funzione matematica, la luce che arriva sul sensore) viene misurato a intervalli regolari per trasformarlo in un segnale **discreto**, cioè definito solo in alcuni istanti, o punti, detti *campioni*. Tra un campione e l'altro il segnale discreto non esiste: è rappresentato soltanto da una sequenza di numeri. Il calcolatore lavora soltanto con dati discreti, ed è per questo che il campionamento è fondamentale in ogni sistema audio, video e di immagini digitali.

Un esempio quotidiano: prendere fotografie a intervalli regolari di un movimento continuo. Più fotografie si fanno ogni secondo, più accuratamente si rappresenta il movimento.

Per un'immagine il campionamento è **spaziale**: la scena continua $f(x, y)$ viene misurata in una griglia regolare di $M \times N$ punti, e ogni misura diventa un pixel. L'intervallo tra due punti vicini, il *passo di campionamento*, fissa la risoluzione dell'immagine (capitolo 11). Nei sensori a matrice visti nel capitolo 6 è la geometria stessa delle celle a fissare questo passo.

Che cosa succede se il campionamento è troppo rado

Un errore nella frequenza di campionamento può stravolgere il segnale in due modi.

- **Si perdono dettagli e informazioni.** Sebbene grave, è spesso una necessità: non si possono conservare milioni di campioni e ci si accontenta di perdere informazione pur di tenere i dati in dimensioni maneggevoli.
- **Compaiono dettagli che nell'originale non c'erano.** Il segnale viene «alterato» e cambiato in qualcosa di altro. Questo fenomeno si chiama **aliasing** (capitolo 10): è subdolo perché imprevedibile e richiede attenzione.

Un campionamento si dice **corretto** quando la frequenza con cui si prendono i campioni è abbastanza alta rispetto alla rapidità con cui varia il segnale. Per sapere quanto è «abbastanza» serve un teorema.

Il Nyquist rate

Il **Nyquist rate** (Harry Nyquist, 1928) è il **doppio** della più alta frequenza presente in un segnale continuo e limitato. Per stimare la frequenza più alta in pratica si può usare una definizione operativa in una dimensione, che si generalizza subito a due:

1. Si osserva il fenomeno su un intervallo, per esempio da a a b .
2. Se il fenomeno è circa costante in tutto l'intervallo, la frequenza più alta è 1: si ha un solo ciclo.
3. Altrimenti si divide l'intervallo in 2, poi in 3, in 4 parti e così via, finché in ciascuna parte il fenomeno è in pratica costante (può variare da una parte all'altra).
4. Se la suddivisione trovata ha N parti, la frequenza più alta è N e il Nyquist rate è **$2N$** .

Per un'immagine una «parte» è un tratto: si cerca il dettaglio più fine che si vuole conservare e si conta quanti di questi dettagli stanno nel lato dell'immagine.

Il teorema del campionamento di Shannon

TEOREMA DI SHANNON (CLAUDE E. SHANNON, 1949)

Se si raccolgono campioni con frequenza **più alta** del Nyquist rate, il segnale può essere ricostruito **fedelmente** in ogni suo punto.

$$f_c > 2 \cdot f_{\max}$$

dove f_c è la frequenza di campionamento e $f_{\max}$ la più alta frequenza contenuta nel segnale. Nella pratica si legge spesso «almeno il doppio»: nel caso limite di campionamento esattamente al doppio il risultato dipende dalla fase del segnale (una sinusoide campionata proprio sui suoi zeri sembrerebbe nulla), quindi è prudente rimanere strettamente sopra il limite. Dal punto di vista matematico la ricostruzione perfetta si ottiene sommando tante funzioni sinc, una per campione; i metodi di interpolazione usati nelle immagini (capitolo 12) ne sono approssimazioni più semplici.

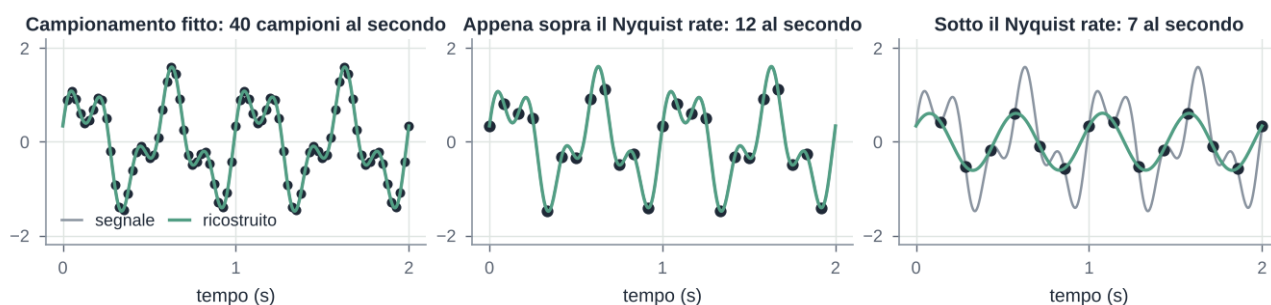


Figura 1. Un segnale con frequenze di 2 e 5 cicli al secondo (Nyquist rate: 10 campioni al secondo). Con 40 e con 12 campioni al secondo la ricostruzione sovrappone il segnale originale. Con 7 campioni al secondo, sotto il Nyquist rate, la curva ricostruita è un'altra: è l'aliasing.

COME SI LEGGE LA FIGURA

Il segnale ha frequenza massima di 5 cicli al secondo, quindi il Nyquist rate è $2 \times 5 = 10$ campioni al secondo. Nel primo e nel secondo caso (40 e 12 campioni al secondo) si è sopra il limite, e la ricostruzione con funzioni sinc coincide con il segnale. Nel terzo (7 campioni al secondo) la componente a 5 cicli è «scambiata» per una a 2 cicli: si ottiene un segnale diverso, con la stessa sequenza di campioni.

Due esempi numerici

Righe su uno schermo. Su un centimetro di schermo si hanno 10 linee, alternativamente bianche e nere. Un ciclo è formato da una linea bianca e una nera, quindi 10 linee sono 5 cicli per centimetro. Servono almeno 2 pixel per ciclo, cioè $5 \times 2 = 10$ pixel per centimetro: per rappresentare correttamente una trama di 10 linee in un centimetro servono almeno 10 pixel in quel centimetro.

Audio. L'orecchio umano arriva a circa 20 kHz, quindi la frequenza minima di campionamento di un suono è circa 40 kHz. Gli standard reali tengono un margine: 44,1 kHz per il CD, 48 kHz per l'audio professionale, 96 kHz o più per le registrazioni di alta qualità.

Applicazione alle immagini: i tratti fini

Si consideri un disegno al tratto di 720×720 pixel con tratti da 4 e da 6 pixel. Il ragionamento è questo: se si preservano i tratti più fini, si preservano anche tutti gli altri.

Il dettaglio più fine ha tratti da 4 pixel. Su 720 pixel stanno $720 / 4 = 180$ tratti, quindi la frequenza più alta è 180 e il Nyquist rate è $2 \times 180 = 360$: bastano **360 campioni per lato**, cioè uno ogni due pixel. Qui ogni tratto è contato

come un intero ciclo, una stima prudente. Se invece si decidesse di trascurare i tratti fini e di conservare solo quelli da 6 pixel, si avrebbero $720 / 6 = 120$ tratti e un Nyquist rate di 240.

CAMPIONI PER LATO	UN CAMPIONE OGNI	RISPETTO AL NYQUIST RATE (360)	ESITO
360 × 360	2 pixel	uguale alla stima	corretto
240 × 240	3 pixel	sotto	i tratti fini si deteriorano
120 × 120	6 pixel	molto sotto	aliasing evidente

Tabella 1. Tre scelte di campionamento per il disegno con tratti da 4 e 6 pixel.

Per verificarlo si tiene un pixel ogni 2, 3 o 6 (righe e colonne a passo fisso) e poi si riporta l'immagine a 720×720 con interpolazione bilineare, per confrontarla con l'originale a parità di dimensioni.

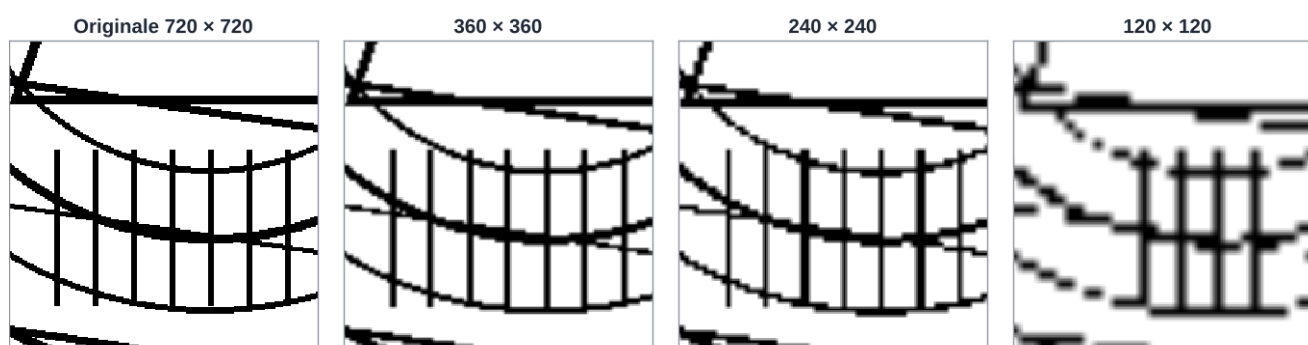


Figura 2. Ritaglio di 200×200 pixel dello stesso disegno campionato con 360, 240 e 120 campioni per lato e riportato a 720×720 con interpolazione bilineare.

Con 360 campioni per lato (un campione ogni 2 pixel) il disegno resta quasi identico, con solo una leggera sfocatura dovuta all'interpolazione. Con 240 (uno ogni 3) i tratti da 4 pixel diventano irregolari e si assottigliano. Con 120 (uno ogni 6) alcuni tratti sottili si spezzano in trattini e le barre verticali in basso, distanti 25 pixel, si fondono in un gruppo con un numero di barre diverso da quello reale: sono comparsi dettagli che non esistevano. Questo è il campionamento sotto il Nyquist rate.

DA RICORDARE

Campionare significa misurare un segnale a intervalli regolari. Per poterlo ricostruire senza errori la frequenza di campionamento deve essere maggiore del doppio della frequenza più alta presente nel segnale (Nyquist rate). Sotto questo limite si perdono dettagli e si creano dettagli falsi (aliasing).

La quantizzazione

Come si arrotondano i valori reali misurati dal sensore a un numero finito di livelli: schema generale, quantizzazione uniforme e logaritmica, errore e numero di bit.

Perché quantizzare

I sensori sono apparecchiature analogiche: forniscono misure di luminosità come numeri **reali**. Un calcolatore non può però rappresentare infiniti valori possibili: può usare solo un numero finito di **livelli**. Occorre quindi arrotondare ciascun valore misurato a uno di questi livelli, e mantenerlo in un certo intervallo. Questo processo si chiama **quantizzazione**. Insieme al campionamento (capitolo 8) completa il passaggio dal continuo al digitale: il campionamento discretizza le coordinate, la quantizzazione discretizza i valori.

Le misure sono inoltre sempre soggette a **errore**, a causa di difetti del sensore e di perturbazioni termiche (rumore). Nei CCD, anche con l'obiettivo chiuso, ci sono correnti parassite che inducono rumore nel dispositivo: sono dette *dark current* (corrente di buio).

Un esempio con soli otto livelli, che sono gli interi da -3 a 4: il campione reale 2,7 viene quantizzato a 3, il campione -1,4 a -1 e il campione 0,1 a 0. Ogni campione è arrotondato al livello più vicino.

La procedura generale

Se i valori da quantizzare sono numeri reali nell'intervallo $[a, b]$ e si vuole quantizzarli su n livelli, si fissano $n + 1$ numeri, detti **soglie**:

$$t_0 = a < t_1 < t_2 < \dots < t_n = b$$

Il valore x viene assegnato al livello k se $t_k \leq x < t_{k+1}$ (il valore b è assegnato all'ultimo livello). A ogni livello k si associa poi un **valore rappresentativo** Q_k , per esempio il centro dell'intervallo.

Fissato il numero di livelli N , bisogna decidere come rappresentarli in memoria: si usano etichette numeriche, e servono

$$B = \log_2(N) \text{ bit}$$

Con 4 livelli bastano 2 bit, con 8 livelli 3 bit, con 256 livelli 8 bit. Viceversa B bit danno 2^B livelli.

BIT	LIVELLI	USO TIPICO
1	2	immagine binaria (bianco e nero)
2	4	esempio didattico
3	8	esempio didattico
8	256	una immagine in toni di grigio, un canale di colore
16	65 536	immagini mediche e fotografie raw ad alta qualità
24 (3 × 8)	16 777 216	immagine a colori RGB

Tabella 1. Numero di livelli in funzione dei bit. Attenzione a un errore frequente: 1 bit dà 2 livelli, non 8 (gli 8 livelli, da 0 a 7, richiedono 3 bit).

Quantizzazione uniforme

Nella quantizzazione **uniforme** i livelli sono equidistanti. Se il livello di ingresso L è un intero in $0 \dots N - 1$ e si vuole una uscita in $0 \dots K - 1$ con $K \leq N$, il livello dopo la quantizzazione è la parte intera di

$$L' = \lfloor L \cdot K / N \rfloor$$

Equivalentemente si calcola il **passo** (*step*) $= N / K$ e si divide il valore per il passo, prendendo la parte intera.

Esempio. Si porta una immagine da 256 livelli a 8 livelli. Il passo è $256 / 8 = 32$, quindi i valori da 0 a 31 diventano il livello 0, da 32 a 63 il livello 1, da 64 a 95 il livello 2, e così via fino ai valori da 224 a 255 che diventano il livello 7. Con la formula:

VALORE	CALCOLO	LIVELLO
10	$10 \cdot 8 / 256 = 0,31$	0
20	$20 \cdot 8 / 256 = 0,63$	0
32	$32 \cdot 8 / 256 = 1$	1
12	$12 / 32 = 0,375$	0
77	$77 / 32 = 2,41$	2
9	$9 / 32 = 0,28$	0
250	$250 / 32 = 7,81$	7

Tabella 2. Riquantizzazione da 256 a 8 livelli. La sequenza 12, 77, 9, 250 diventa 0, 2, 0, 7.

Il caso estremo è la **binarizzazione**, cioè la quantizzazione a due livelli: si fissa una soglia, e i valori sopra la soglia diventano bianco, gli altri nero. Si ottiene l'immagine in bianco e nero.

Per visualizzare un'immagine a K livelli su uno schermo a 256 livelli, i livelli $0 \dots K - 1$ vanno riportati sulla scala $0 \dots 255$, per esempio al centro di ciascun intervallo.

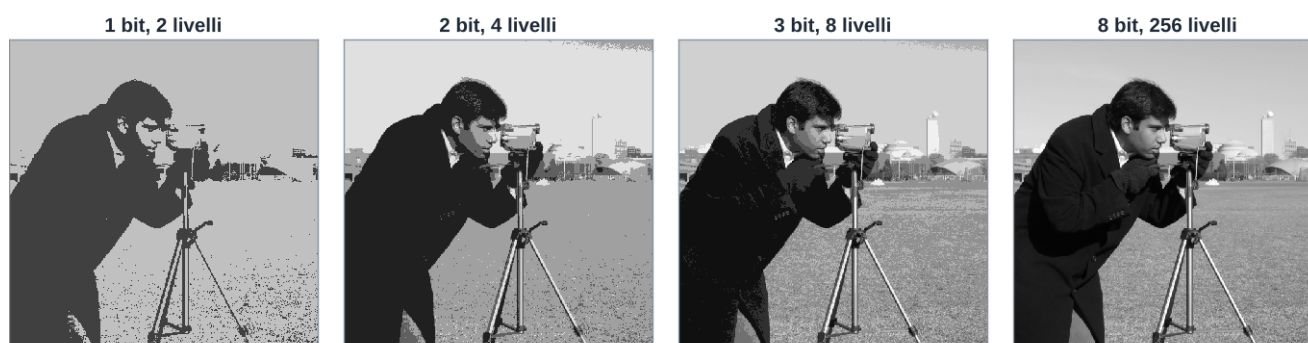


Figura 1. Effetto della quantizzazione uniforme. Con pochi livelli compaiono falsi contorni (posterizzazione) nelle zone di tono continuo.

L'errore di quantizzazione

La quantizzazione introduce sempre un piccolo errore: la differenza tra il valore reale e il valore arrotondato, detta **errore di quantizzazione**. Più livelli si usano, più l'errore è piccolo e migliore è la qualità. Con passo s e valore rappresentativo al centro dell'intervallo l'errore massimo è $s / 2$, e per valori distribuiti in modo uniforme l'errore quadratico medio vale circa $s^2 / 12$. Con 8 livelli su 256 il passo è 32 e si prevede un errore quadratico medio di $32^2 / 12 \approx 85,3$, a fronte di 87,7 misurato sull'immagine di prova.

LIVELLI K	BIT	ERRORE QUADRATICO MEDIO	PSNR (DB)
2	1	1229,2	17,2
4	2	282,0	23,6
8	3	87,7	28,7
16	4	20,8	35,0
32	5	5,4	40,8
64	6	1,5	46,3
128	7	0,5	51,1

Tabella 3. Qualità della fotografia «cameraman» quantizzata uniformemente da 256 a K livelli (il PSNR è definito nel capitolo 13).

Ogni bit in più riduce il passo a metà e quindi l'errore quadratico medio di un fattore 4: il PSNR cresce di circa 6 dB per bit, come si vede dalla tabella.

Quantizzazione non uniforme e logaritmica

Nella quantizzazione **non uniforme** il livello dopo la riquantizzazione è $L' = f(L, N, K)$, dove la funzione f definisce lo schema e può avere le forme più varie. La più comune è la quantizzazione **logaritmica**:

$$L' = \lfloor \log_2(L) \cdot K / \log_2(N) \rfloor$$

Nel caso più comune $N = 256$ e $K = 8$, perciò $\log_2(N) = 8$ e la formula diventa semplicemente $L' = \lfloor \log_2(L) \rfloor$ (per $L = 0$, dove il logaritmo non è definito, si pone per convenzione $L' = 0$). Si può anche portare 0 ... 255 in 0 ... 15 e allora la formula generale è necessaria. Gli intervalli che ne risultano per $K = 8$ sono:

LIVELLO	0	1	2	3	4	5	6	7
Da (valore minimo)	0	2	4	8	16	32	64	128
A (valore massimo)	1	3	7	15	31	63	127	255
Ampiezza	2	2	4	8	16	32	64	128

Tabella 4. Intervalli della quantizzazione logaritmica da 256 a 8 livelli.

Gli intervalli sono stretti per i toni scuri e larghissimi per i toni chiari: la quantizzazione logaritmica assegna **più livelli alla zona dei toni scuri** e meno a quella dei toni chiari. Il motivo è la percezione (capitolo 5): l'occhio distingue meglio i rapporti tra intensità che le differenze assolute. Per questo la quantizzazione effettuata da scanner e fotocamere è non uniforme, ed è particolarmente importante quando si elaborano dati medici come le radiografie.

Un'altra forma è la **compressione non lineare seguita da quantizzazione uniforme**, nota come legge μ . Con V_{in} e V_{out} valori normalizzati, V_{max} il livello massimo e μ il parametro di compressione (valore standard 255):

$$V_{out} = V_{max} \cdot \ln(1 + \mu \cdot |V_{in}| / V_{max}) / \ln(1 + \mu) \cdot \text{sgn}(V_{in})$$

La funzione V_{out} sale rapidamente per i valori piccoli e poi si appiattisce: dopo averla applicata, una quantizzazione uniforme dedica più livelli ai valori piccoli.

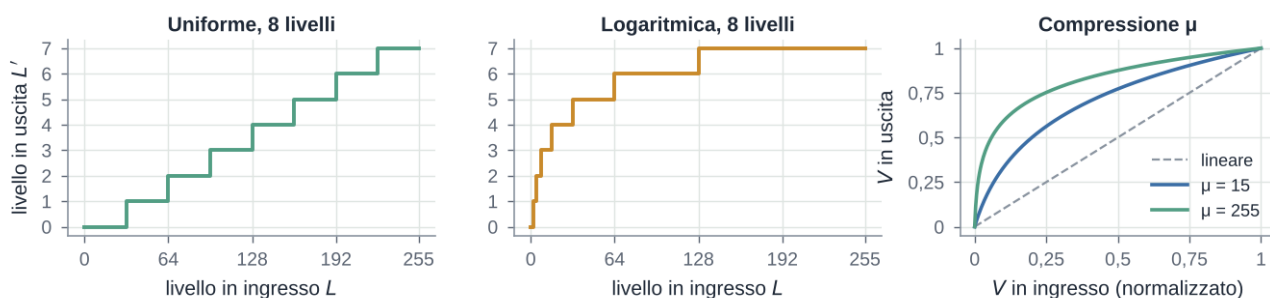


Figura 2. A sinistra e al centro la funzione di quantizzazione uniforme e logaritmica da 256 a 8 livelli. A destra la compressione μ : più μ è grande, più la curva dedica spazio ai valori piccoli.



Figura 3. Su una fotografia scura la quantizzazione uniforme a 8 livelli schiaccia il volto e il cappotto in due o tre tonalità quasi nere, quella logaritmica ne conserva meglio la struttura, a prezzo di una resa molto grossolana dei toni chiari come il cielo. Per la visualizzazione ogni livello è riportato al centro del suo intervallo di ingresso.

Immagini a colori

In una immagine a colori RGB la matrice di uscita è formata da 3 matrici delle stesse dimensioni $M \times N$, una per canale (rosso, verde, blu), e ciascuna è quantizzata a 8 bit, cioè 256 livelli. Ogni pixel può quindi assumere $256 \times 256 \times 256 = 16\,777\,216$ colori, con 24 bit per pixel. Se si riduce a 4 livelli per canale, i colori possibili sono $4 \times 4 \times 4 = 64$: l'immagine ha meno tonalità ed è meno dettagliata. In generale più bit significano più livelli e quindi una rappresentazione migliore, a prezzo di più memoria.

DA RICORDARE

La quantizzazione arrotonda i valori misurati a un numero finito di livelli. Quella uniforme usa $L' = \lfloor L \cdot K / N \rfloor$ e introduce un errore al più pari a mezzo passo; quella logaritmica dedica più livelli ai toni scuri. Servono $\log_2(N)$ bit per N livelli, e ogni bit in più migliora il PSNR di circa 6 dB.

Aliasing, antialiasing e moiré

Che cosa succede quando si campiona sotto il Nyquist rate: le alte frequenze si travestono da basse frequenze, con effetti visibili nelle immagini, nei video e nel suono.

Sottocampionamento e frequency aliasing

Che cosa succede se si campiona a una frequenza inferiore al Nyquist rate (capitolo 8)? Si perdono dettagli significativi e spesso se ne introducono di nuovi, che nella realtà non ci sono. Questo fenomeno si chiama **frequency aliasing**, o semplicemente **aliasing**: le alte frequenze vengono «mascherate» da basse frequenze e trattate come tali nella fase di campionamento. La parola viene da *alias*, falsa identità.

Il caso più semplice è una sinusoidale. Se il segnale ha frequenza f e viene campionato con frequenza f_c , i campioni sono identici a quelli di una sinusoidale di frequenza apparente

$$f_{\text{app}} = |f - k \cdot f_c|$$

dove k è l'intero che porta il risultato nell'intervallo da 0 a $f_c / 2$. Per esempio un segnale a 9 Hz campionato a 10 Hz (il Nyquist rate sarebbe 18 Hz) ha frequenza apparente $|9 - 1 \cdot 10| = 1$ Hz: i campioni coincidono con quelli di un segnale lentissimo.

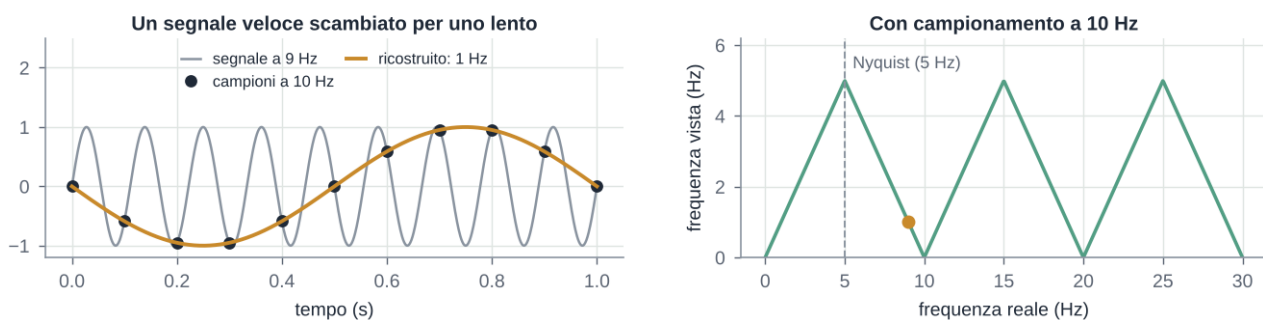


Figura 1. A sinistra: i campioni di un segnale a 9 Hz presi a 10 Hz coincidono con quelli di una sinusoidale a 1 Hz. A destra: frequenza vista in funzione della frequenza reale; fino a 5 Hz coincidono, poi la frequenza vista «si riflette».

Se si trattasse di un'onda sonora, un suono acuto verrebbe sostituito da un suono grave. Per questo la scelta della frequenza di campionamento è un passo essenziale per realizzare un CD: si deve campionare più del doppio della frequenza massima udibile.

Aliasing nelle immagini

Nelle immagini l'aliasing ha due aspetti tipici: **scalettature** sui bordi (i bordi inclinati diventano a gradini) e comparsa di **falsi dettagli**, per esempio fori o linee che nell'originale non c'erano. Se si riduce molto una fotografia di un paesaggio roccioso prendendo un pixel ogni tanto, graffi e disegni sulle rocce diventano indistinguibili, e compaiono insieme bordi scalettati e fori che nell'originale non c'erano.

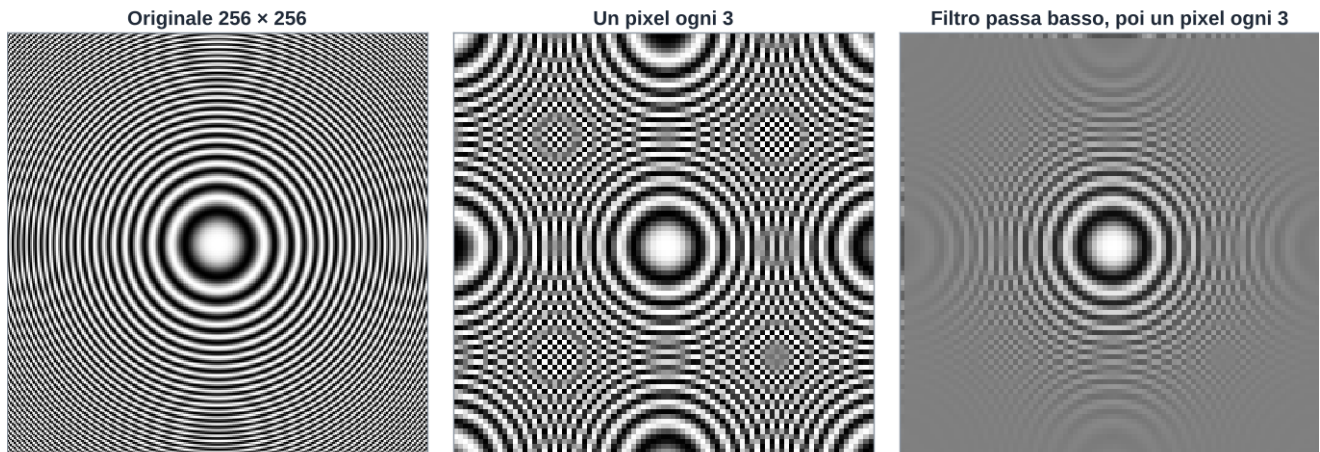


Figura 2. Un disegno con anelli sempre più fitti verso l'esterno (frequenza crescente). Prendendo un pixel ogni 3 compaiono, al posto degli anelli fitti, anelli larghi e archi che non esistono (aliasing e moiré). Se si leviga prima l'immagine con un filtro passa basso, i dettagli troppo fini scompaiono ma non si creano falsi dettagli.

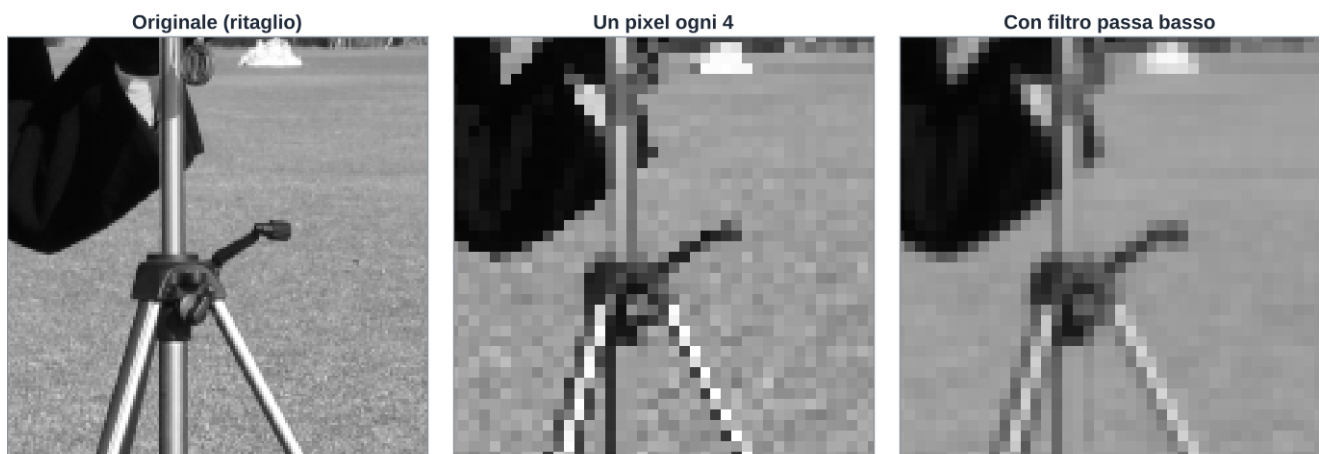


Figura 3. Il treppiede e il prato della fotografia «cameraman» ridotti di un fattore 4. Senza filtro le gambe sottili del treppiede si spezzano in punti e il prato assume una trama irregolare; con il filtro passa basso l'immagine è più sfocata ma non contiene falsi disegni.

Il moiré

Un caso molto visibile di aliasing è il **moiré**: un disegno a righe o a quadretti molto fitti, ripreso con un sensore che ha anch'esso una griglia regolare, produce una trama a onde larghe che non esiste nella scena. Si vede per esempio quando una camicia a righe fini è ripresa in televisione, o quando si fotografa uno schermo. Nasce dall'interferenza tra due strutture periodiche con passi vicini: la griglia del sensore e il disegno. Con la formula di prima: un reticolo con periodo di 5 pixel ha frequenza $1/5 = 0,2$ cicli per pixel; se si tiene un pixel ogni 4 la frequenza di campionamento è $1/4 = 0,25$ e quella apparente $|0,2 - 0,25| = 0,05$ cicli per pixel, cioè un periodo di 20 pixel dell'originale, quattro volte più largo del vero.

ATTENZIONE A UN'ASSOCIAZIONE SBAGLIATA

Il moiré non dipende dalla quantizzazione e non nasce dalla sovrapposizione di due fotogrammi: è un effetto di aliasing, cioè di campionamento spaziale troppo rado, o di sovrapposizione di due reticoli. La quantizzazione produce invece i falsi contorni visti nel capitolo 9.

Aliasing temporale: la ruota del carro

Anche il tempo si campiona: un video è una sequenza di fotogrammi presi a intervalli regolari. L'aliasing temporale è quello della **ruota del carro** (*wagon wheel*) nei film: una ruota che gira veloce appare ferma o che gira all'indietro. Se la ruota ha 8 raggi uguali, il suo aspetto si ripete ogni $360^\circ / 8 = 45^\circ$. Se tra un fotogramma e il successivo ruota di 40° , l'occhio abbina ogni raggio al raggio più vicino del fotogramma precedente, che dista $45^\circ - 40^\circ = 5^\circ$ dietro di sé: la ruota sembra girare all'indietro di 5° per fotogramma, anziché avanti di 40° . Un movimento viene sostituito da un altro.

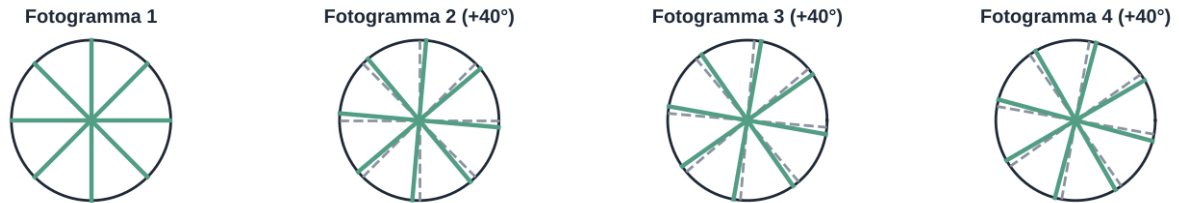


Figura 4. Una ruota a 8 raggi che ruota di 40° per fotogramma. In verde i raggi nel fotogramma, tratteggiati in grigio quelli del precedente: il raggio più vicino al tratteggiato sta 5° dietro, quindi il moto percepito è all'indietro.

Lo stesso accade con un'animazione costruita con tutti i campioni disponibili, con uno ogni 4 o uno ogni 25: nell'ultimo caso il moto viene percepito come retrogrado. Cercando su internet «wagon wheel aliasing» si trovano molti video dimostrativi.

Antialiasing

Nella realtà l'aliasing è sempre presente, almeno in misura minima, perché nessun segnale reale è davvero limitato in banda: l'aliasing viene introdotto quando si impone che il segnale sia limitato per poterlo campionare. Può però essere ridotto applicando una funzione di **smussamento** (filtro passa basso) al segnale originario *prima* del campionamento: si tolgono volontariamente le frequenze troppo alte, che comunque non potrebbero essere rappresentate. Questa tecnica si chiama **antialiasing**.

Il prezzo è una immagine un po' più sfocata, ma priva di falsi dettagli: tra un dettaglio perso e un dettaglio falso si preferisce perdere. Le fotocamere applicano questo principio con un filtro ottico passa basso davanti al sensore, e i programmi di grafica applicano un filtro prima di ridurre un'immagine. Il filtraggio è il tema dei capitoli 22 e 23.

DA RICORDARE

Campionando sotto il Nyquist rate le alte frequenze si presentano come basse frequenze (aliasing): compaiono scalettature, falsi dettagli e moiré, e nei video movimenti sbagliati. L'antialiasing consiste nel filtrare con un passa basso prima di campionare o ridurre.

La risoluzione delle immagini digitali

Quanti pixel per unità di lunghezza: dpi, megapixel, risoluzione di ripresa e di resa, formati standard e legame con la memoria occupata.

Che cos'è la risoluzione

RISOLUZIONE

Si dice **risoluzione** il numero di pixel per unità di misura. Si misura in pixel al centimetro o in **dpi** (*dots per inch*, punti per pollice). Può anche essere espressa come numero di pixel dell'intera immagine, per esempio 16 megapixel.

Il pollice (*inch*) vale 2,54 cm, quindi 1 pixel per centimetro equivale a 2,54 dpi. Il legame tra dimensione fisica, risoluzione e numero di pixel è

$$\text{pixel} = \text{lunghezza (in pollici)} \times \text{dpi}$$

Una pagina A4 (circa $8,27 \times 11,69$ pollici) acquisita a 300 dpi è quindi di circa 2480×3508 pixel, cioè poco meno di 8,7 megapixel. A 24 bit per pixel occupa $2480 \times 3508 \times 3 = 26\,099\,520$ byte, circa 26 MB senza compressione.

La risoluzione indica il **grado di qualità** di un'immagine. Lo schermo di un computer non può mostrare linee o disegni, ma soltanto punti: se questi sono abbastanza piccoli, più piccoli della risoluzione percepita dall'occhio, l'osservatore ha l'impressione di vedere linee anziché punti allineati e disegni anziché ammassi di puntini distinti.

Risoluzione di ripresa e di resa

Il concetto di risoluzione è **relativo** e va considerato con attenzione nelle diverse situazioni, perché esistono due risoluzioni diverse: quella dell'apparecchiatura di *ripresa* e quella dell'apparecchiatura di *resa*. In entrambi i casi si contano i sensori, o gli elementi, per unità lineare di misura.

APPARECCHIATURA	TIPO	RISOLUZIONE TIPICA
Scanner	ripresa	fino a 6000 dpi e oltre
Fotocamera	ripresa	numero di sensori del circuito di ripresa, in megapixel
Pellicola fotografica	ripresa (equivalente)	da 3000 a 4000 dpi equivalenti
Stampante	resa	fino a 3000 dpi e oltre
Schermo	resa	numero di elementi per unità di misura, tradizionalmente 72 dpi

Tabella 1. Risoluzione di alcune apparecchiature.

Per le stampe le slide riportano valori che dipendono dal prodotto: quotidiano 75, rivista 133, brochure 175, libro fotografico 2400. I primi tre sono in pratica le **lineature di retino** (linee per pollice) della stampa a toni di grigio, e la risoluzione dell'immagine che serve è di norma da una volta e mezza a due volte la lineatura; il 2400 è la risoluzione della macchina da stampa o da fotocomposizione, che usa molti punti dell'apparecchio per rendere un singolo pixel dell'immagine. Il messaggio è comunque chiaro: la risoluzione necessaria dipende dall'uso finale.

REGOLA PRATICA

Un'immagine nata con una certa risoluzione deve essere visualizzata con la stessa risoluzione per avere il massimo della resa: se la si ingrandisce, ogni pixel diventa più grande e i dettagli non aumentano, come si vede nell'esempio che segue.

Esempio: lo stesso soggetto a tre risoluzioni

Si deve rappresentare una scena larga 2 m. Con 200 pixel di larghezza la risoluzione «reale» è di 1 punto per centimetro (200 pixel per 200 cm); con 100 pixel è 1 punto ogni 2 cm; con 10 pixel è 1 punto ogni 20 cm. Se l'immagine è a colori con 24 bit per pixel e quadrata, la memoria necessaria è:

LARGHEZZA	RISOLUZIONE REALE	MEMORIA (24 BIT PER PIXEL)
200 pixel	1 punto per 1 cm	$200 \times 200 \times 24 = 960\,000$ bit
100 pixel	1 punto per 2 cm	$100 \times 100 \times 24 = 240\,000$ bit
10 pixel	1 punto per 20 cm	$10 \times 10 \times 24 = 2\,400$ bit

Tabella 2. Dimezzando la risoluzione la memoria si riduce a un quarto.

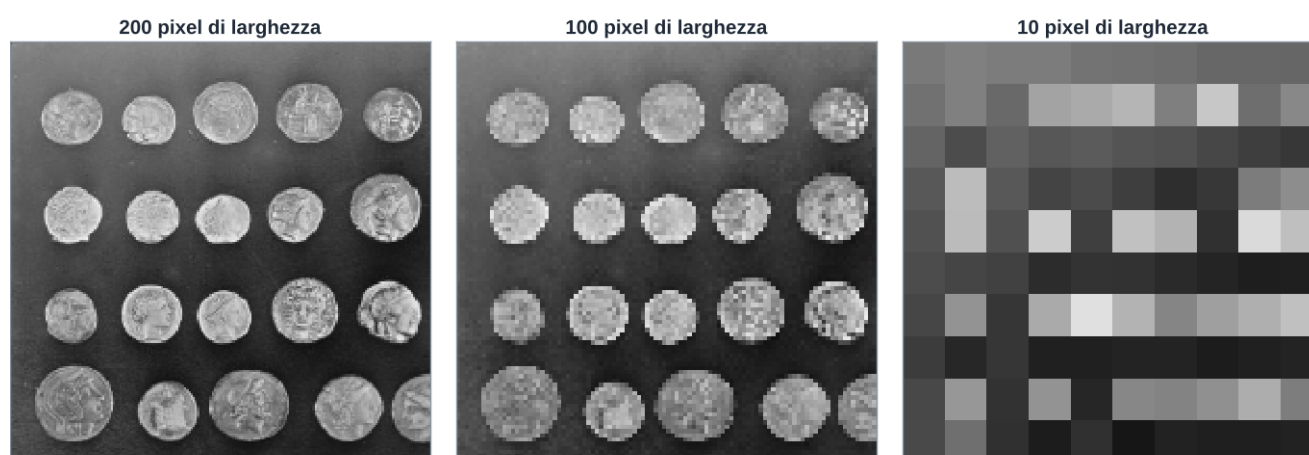


Figura 1. La stessa scena larga 2 m rappresentata con 200, 100 e 10 pixel per lato (1 punto per 1 cm, 1 per 2 cm, 1 per 20 cm). Ogni immagine è mostrata alla stessa dimensione, perciò i pixel diventano sempre più grandi.

Dimezzare il numero di pixel per lato divide per 4 il numero totale di pixel e quindi la memoria: la qualità cala, ma anche il costo di memorizzazione e di trasmissione.

Quanta risoluzione serve all'occhio

La risoluzione utile dipende dalla distanza a cui si osserva. Un occhio normale distingue due punti che sottendono un angolo di circa 1 primo d'arco, cioè $1/60$ di grado (capitolo 3). A una distanza d la dimensione minima distinguibile è $d \cdot \tan(1')$, quindi la risoluzione massima utile è l'inverso di questa dimensione.

DISTANZA DI OSSERVAZIONE	DETTAGLIO MINIMO	RISOLUZIONE MASSIMA UTILE
30 cm (smartphone)	0,087 mm	circa 290 dpi
60 cm (monitor)	0,175 mm	circa 145 dpi
3 m (televisore)	0,87 mm	circa 29 dpi

Tabella 3. Risoluzione oltre la quale l'occhio non distingue i singoli pixel (calcolata con 1 primo d'arco).

Per questo gli schermi degli smartphone, osservati da vicino, hanno risoluzioni di diverse centinaia di dpi, mentre un televisore, guardato da lontano, ne richiede molti meno.

Formati standard di schermo

Le dimensioni più comuni hanno nomi convenzionali. Il numero di pixel complessivo, in megapixel, si ottiene moltiplicando larghezza per altezza; l'ultima colonna dà la memoria a 24 bit per pixel.

NOME	PIXEL	MEGAPIXEL	MEMORIA (MB)
VGA	640 × 480	0,31	0,9
SVGA	800 × 600	0,48	1,4
XGA	1024 × 768	0,79	2,4
HD (720p)	1280 × 720	0,92	2,8
Full HD (1080p)	1920 × 1080	2,07	6,2
QHD	2560 × 1440	3,69	11,1
4K UHD	3840 × 2160	8,29	24,9
8K UHD	7680 × 4320	33,18	99,5

Tabella 4. Formati standard più comuni. La memoria è quella di una immagine non compressa a 24 bit per pixel.

DA RICORDARE

La risoluzione è il numero di pixel per unità di lunghezza (pixel per cm, dpi) oppure il numero totale di pixel (megapixel). Va distinta la risoluzione di ripresa da quella di resa. Raddoppiare la risoluzione per lato quadruplica pixel e memoria.

Interpolazione: replication, bilineare e bicubica

Come si stimano i valori che mancano quando un'immagine viene trasformata, ingrandita o ridotta: dal mapping delle trasformazioni affini ai tre metodi di interpolazione.

Perché serve interpolare

L'**interpolazione** è il processo che, partendo da dati reali, **stima** i dati non conosciuti. Non comporta un miglioramento della qualità dell'immagine, come se si «riacquisissero» i valori mancanti: effettua soltanto una stima dei valori ignoti, riempiendo i buchi. Serve in tre situazioni: nelle **trasformazioni geometriche** (rotazione, scalatura), nello **zooming** e nella **color interpolation** del Bayer pattern (capitolo 7).

Trasformazioni affini e mapping

Le trasformazioni affini cambiano la posizione dei pixel. Nel corso si usa la convenzione dei **vettori riga**: con T matrice 3×3 e coordinate omogenee, il pixel di ingresso (v, w) va nel pixel di uscita (x, y) dato da

$$[x \ y \ 1] = [v \ w \ 1] \cdot T$$

Le tre trasformazioni di base, con θ angolo di rotazione, (t_x, t_y) traslazione e (c_x, c_y) fattori di scala, hanno le matrici

Rotazione			Traslazione			Scala		
$\cos \theta$	$\sin \theta$	0	1	0	0	c_x	0	0
$-\sin \theta$	$\cos \theta$	0	0	1	0	0	c_y	0
0	0	1	t_x	t_y	1	0	0	1

Figura 1. Matrici delle trasformazioni affini con la convenzione dei vettori riga.

Le trasformazioni si **combinano** moltiplicando le matrici: con rotazione, traslazione e scala si ha $T = T_{\text{rot}} \cdot T_{\text{trasl}} \cdot T_{\text{scala}}$, applicate nell'ordine in cui sono scritte da sinistra a destra, perché il vettore riga sta a sinistra. Poiché il prodotto di matrici non è commutativo, l'ordine conta. Per ruotare attorno al centro dell'immagine si traslano prima il centro nell'origine, poi si ruota, poi si riporta indietro.

Ci sono due modi di applicare T .

- **Forward mapping**: si scorre l'immagine di *ingresso* e per ogni pixel (v, w) si calcola la posizione (x, y) nell'immagine di uscita con la formula sopra.
- **Inverse mapping**: si scorre l'immagine di *uscita* e per ogni pixel (x, y) si calcola da quale punto dell'ingresso proviene, con la formula inversa $[v \ w \ 1] = [x \ y \ 1] \cdot T^{-1}$. È il metodo usato da MATLAB.

Le posizioni calcolate sono numeri reali che vanno arrotondati: con il forward mapping alcuni pixel di uscita non sono mai individuati dalle formule e restano vuoti (basta pensare a un ingrandimento: pochi pixel di ingresso devono coprire molte posizioni), mentre due pixel di ingresso possono finire nello stesso pixel di uscita. L'inverse mapping evita il problema: ogni pixel di uscita viene visitato esattamente una volta, e il valore si ricava per **interpolazione** dai pixel di ingresso intorno al punto trovato.

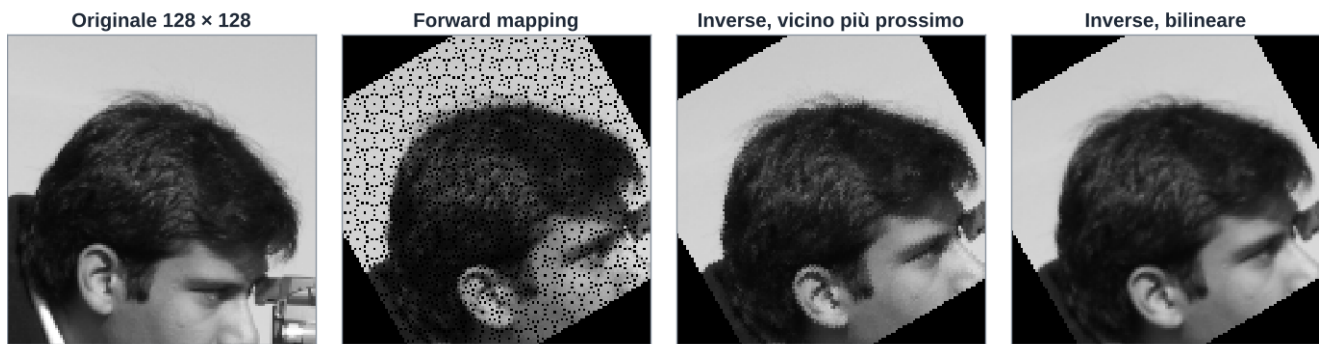


Figura 2. Rotazione di 30° attorno al centro. Nel forward mapping molti pixel di uscita non vengono mai raggiunti e restano neri; con l'inverse mapping ogni pixel di uscita riceve un valore, scelto con il vicino più prossimo o con l'interpolazione bilineare.

Zooming: ingrandire

Uno zooming 2× raddoppia le dimensioni: una immagine $m \times n$ diventa $2m \times 2n$, quindi il numero totale di pixel **quadruplica**. Si può anche scalare in modo diverso nelle due direzioni, per esempio raddoppiando le righe e triplicando le colonne: in tal caso gli algoritmi sono più complicati. Dopo aver posizionato i valori già noti, occorre stimare quelli nelle posizioni vuote. Ci sono tre metodi principali.

REPLICATION (VICINO PIÙ PROSSIMO)

Ogni nuova posizione riceve l'intensità del pixel più vicino dell'immagine originale. È molto semplice, ma introduce artefatti: distorsioni lungo i bordi degli oggetti, che appaiono a gradini.

INTERPOLAZIONE BILINEARE

Si usano i **quattro** pixel più vicini al punto (x, y) . Il valore assegnato è

$$v(x, y) = a \cdot x + b \cdot y + c \cdot x \cdot y + d$$

dove i quattro coefficienti si ricavano risolvendo le quattro equazioni, una per ciascun pixel vicino. Con i quattro vicini a $v(0, 0) = 10$, $v(1, 0) = 20$, $v(0, 1) = 30$, $v(1, 1) = 60$ si trova $d = 10$, $a = 20 - 10 = 10$, $b = 30 - 10 = 20$, $c = 60 - 20 - 30 + 10 = 20$. Nel punto di mezzo $(0,5; 0,5)$ si ottiene $5 + 10 + 5 + 10 = 30$, che è la media dei quattro valori; nel punto $(0,25; 0,75)$ si ottiene $2,5 + 15 + 3,75 + 10 = 31,25$. Il risultato è migliore della replication con un modesto aumento della complessità.

Esempio. Zoom 2× di una matrice 3 × 3: i valori noti vanno nelle posizioni di indice pari, nelle altre si fanno medie dei vicini noti.

Ingresso 3 × 3

1	2	3
4	5	6
7	8	9

→

Uscita 5 × 5 (in verde i valori noti)

1	1,5	2	2,5	3
2,5	3	3,5	4	4,5
4	4,5	5	5,5	6
5,5	6	6,5	7	7,5
7	7,5	8	8,5	9

Figura 3. Zooming bilineare: ad esempio il valore in alto al centro è $(1 + 2) / 2 = 1,5$ e quello al centro del primo quadrato è $(1 + 2 + 4 + 5) / 4 = 3$.

INTERPOLAZIONE BICUBICA

Usa i **sedici** pixel più vicini al punto. Il valore è un polinomio di terzo grado in x e in y con sedici coefficienti, determinati risolvendo sedici equazioni in sedici incognite, una per punto. In pratica si calcola come media pesata dei 16 vicini con pesi che dipendono dalla distanza (figura 4). Generalmente la bicubica preserva meglio i dettagli della bilineare ed è la tecnica standard dei programmi commerciali di editing, come Adobe Photoshop. I pesi possono essere anche *negativi*: in figura il pixel più lontano pesa $-0,0625$ e dà alla bicubica un leggero effetto di accentuazione dei bordi.

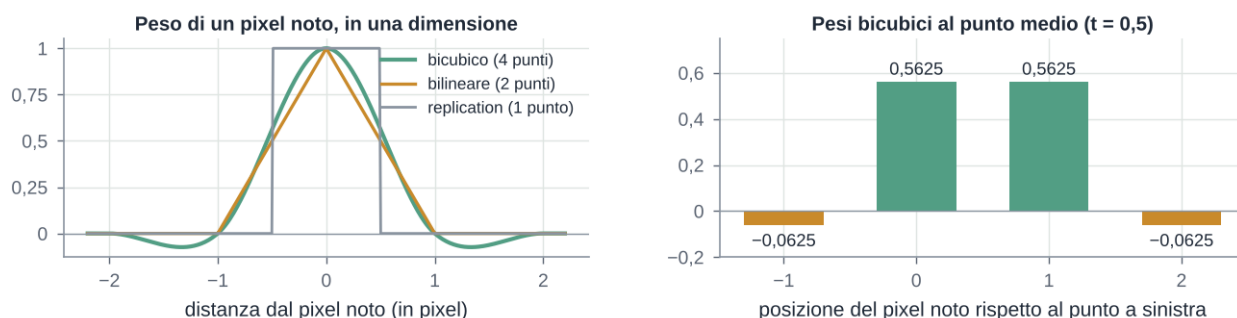


Figura 4. A sinistra i pesi dei tre metodi in una dimensione; a destra i pesi bicubici nel punto a metà tra i due pixel centrali.

Il problema dei bordi

Ai bordi dell'immagine mancano alcuni dei pixel vicini. Le soluzioni possibili sono tre.

1. Non fare nulla e **non calcolare** i valori ai bordi, considerando solo le zone centrali (si ottiene una immagine più piccola).
2. **Replicare** le ultime righe e colonne dopo aver fatto i calcoli: la matrice $(2m - 1) \times (2n - 1)$ dello zoom diventa $2m \times 2n$ copiando l'ultima riga e l'ultima colonna.
3. Interpolare con i valori **presenti**, anche se in numero minore di quelli usati per gli altri pixel.

Zoom 6×6 con ultima riga e colonna replicate

1	1,5	2	2,5	3	3
2,5	3	3,5	4	4,5	4,5
4	4,5	5	5,5	6	6
5,5	6	6,5	7	7,5	7,5
7	7,5	8	8,5	9	9
7	7,5	8	8,5	9	9

Figura 5. La seconda soluzione: la riga e la colonna aggiunte (in ambra) replicano quelle dell'ultima posizione calcolata.

Zooming out e decimazione

Se il fattore di zoom è minore di 1 si ottiene una immagine più piccola: il processo si chiama **decimazione**. Con fattore 0,5, una immagine $m \times n$ diventa $m / 2 \times n / 2$. Si divide l'immagine in blocchi 2×2 e si può:

- **metodo 1**: scegliere un solo pixel per blocco (il primo, oppure a caso);
- **metodo 2**: calcolare il **valore medio** dei quattro pixel del blocco.

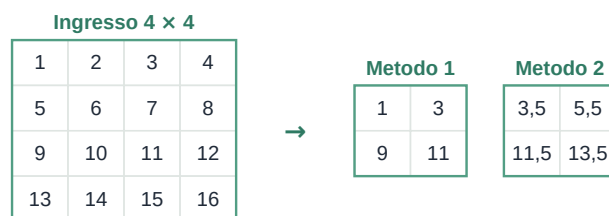


Figura 6. Decimazione di una matrice con i due metodi. Il metodo 1 prende il primo pixel di ogni blocco; il metodo 2 fa la media: ad esempio $(1 + 2 + 5 + 6) / 4 = 3,5$.

Le slide del corso riportano per il metodo 2 i valori 3, 5, 11, 13, che sono le medie 3,5, 5,5, 11,5 e 13,5 troncate all'intero. Il metodo 2 equivale a un filtro passa basso seguito da una scelta di un pixel, e riduce l'aliasing (capitolo 10); il metodo 1 no.

Confronto tra i metodi

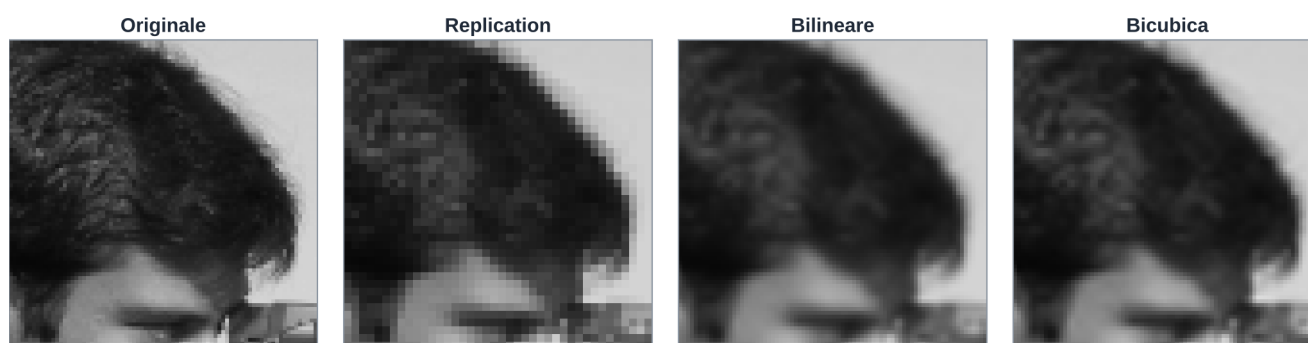


Figura 7. Ritaglio di 80 × 80 pixel (capelli e fronte). La fotografia è stata ridotta a 256 × 256 (media su blocchi 2 × 2) e riportata a 512 × 512 con i tre metodi.

Per confrontare i metodi con una misura oggettiva si riduce la fotografia «cameraman» di un fattore 2 e la si riporta alla dimensione originale, poi si calcola il PSNR (capitolo 13) rispetto all'immagine di partenza. Risultati: replication 28,68 dB, bilineare 29,12 dB, bicubica 29,99 dB. L'ordine è quello atteso. Il PSNR non è però il parametro migliore per valutare un algoritmo di interpolazione, anche se è il più diffuso: il giudizio dell'occhio, specie sui bordi, può essere diverso.

METODO	VICINI USATI	QUALITÀ	COSTO
Replication	1	bordi a gradini	minimo
Bilineare	4	buona, un po' sfocata	modesto
Bicubica	16	migliore, bordi più netti	maggiore

Tabella 1. Riepilogo dei tre metodi di interpolazione.

DA RICORDARE

L'interpolazione stima valori sconosciuti ma non aggiunge informazione. Replication copia il vicino, bilineare usa 4 pixel e un polinomio $a \cdot x + b \cdot y + c \cdot x \cdot y + d$, bicubica usa 16 pixel. Nelle trasformazioni geometriche si preferisce l'inverse mapping, perché non lascia buchi.

Il PSNR e la misura della qualità

Come si misura numericamente quanto un'immagine elaborata si discosta dall'originale: errore quadratico medio, PSNR in decibel, immagini a colori e limiti della misura.

Misurare la qualità

Quando un'immagine viene compressa, ridotta, ingrandita o modificata, serve un numero che dica quanto l'immagine risultante si discosta dall'originale. Due indici molto usati sono l'**MSE** (*Mean Square Error*, errore quadratico medio) e il **PSNR** (*Peak Signal to Noise Ratio*, rapporto tra il picco del segnale e il rumore). Sono tecniche **full reference**: per calcolarli servono sia l'immagine I da valutare sia una sua versione ottimale I' , entrambe $M \times N$.

L'errore quadratico medio

MSE

L'errore quadratico medio tra I e I' è la media dei quadrati delle differenze tra i pixel corrispondenti. Più è basso, minore è la differenza tra le immagini.

$$\text{MSE} = 1 / (M \cdot N) \cdot \sum_{x=1 \dots M} \sum_{y=1 \dots N} (I(x, y) - I'(x, y))^2$$

Il PSNR

Il PSNR si ottiene dall'MSE con una scala logaritmica e si esprime in **decibel** (dB). Maggiore è il suo valore, maggiore è la somiglianza con l'originale.

$$\text{PSNR} = 10 \cdot \log_{10}(\text{MAX}^2 / \text{MSE}) = 20 \cdot \log_{10}(\text{MAX} / \sqrt{\text{MSE}})$$

Le due scritture sono equivalenti. MAX è il massimo valore possibile di un pixel: 255 per immagini a 8 bit, in generale $2^B - 1$ per immagini a B bit. Se le immagini sono identiche l'MSE è 0 e il PSNR è infinito.

Esempio. Due immagini 3×3 a 8 bit:

I			I'			Differenza I - I'		
52	55	61	50	57	60	2	-2	1
63	59	66	63	61	64	0	-2	2
62	59	68	65	59	70	-3	0	-2

Figura 1. Due immagini 3×3 e la loro differenza pixel per pixel.

I quadrati delle differenze sono 4, 4, 1, 0, 4, 4, 9, 0, 4, con somma 30. L'MSE è $30 / 9 = 3,33$ e

$$\text{PSNR} = 10 \cdot \log_{10}(255^2 / 3,33) = 10 \cdot \log_{10}(19\,507,5) \approx 42,9 \text{ dB}$$

Come si interpreta. Dimezzare l'ampiezza dell'errore divide l'MSE per 4 e aumenta il PSNR di circa 6 dB ($10 \cdot \log_{10} 4 \approx 6,02$). Per immagini a 8 bit compresse con perdita, valori tra 30 e 50 dB sono tipici: sopra i 40 dB le differenze sono in genere poco visibili, sotto i 30 dB cominciano a essere evidenti. Nell'esempio delle slide del corso, dedicato alla compressione JPEG, il PSNR parte da 19,33 dB con il fattore di qualità $Q = 1$, cioè con una compressione molto spinta e un'immagine visibilmente degradata.

MSE	ERRORE TIPICO PER PIXEL	PSNR (DB)
1	1,0	48,1
10	3,2	38,1
100	10,0	28,1
1000	31,6	18,1

Tabella 1. Corrispondenza tra MSE, scarto quadratico (radice dell'MSE) e PSNR per immagini a 8 bit.

Il PSNR nelle operazioni di zooming

Quando si valuta un algoritmo di ingrandimento si parte da un'immagine I , la si riduce, la si riporta alla dimensione originale con l'algoritmo da valutare e si confronta il risultato I' con I . Non si può infatti confrontare direttamente un'immagine con una versione ingrandita ($2\times$ per esempio): i pixel non corrispondono più uno a uno e le dimensioni sono diverse. Ecco lo schema, che è quello usato nel capitolo 12:



Figura 2. Schema per il calcolo del PSNR di un algoritmo di interpolazione.

Il PSNR non è il migliore parametro per valutare la qualità di un algoritmo di interpolazione, ma è il più diffuso.

Immagini a colori

Per immagini RGB si usa di solito una di queste soluzioni: la semplice **media** degli MSE (o dei PSNR) sui tre canali; oppure una **combinazione lineare** che pesa maggiormente il canale verde, perché più importante per la percezione. Per esempio, con la ricostruzione di Bayer del capitolo 7 sulla fotografia dell'astronauta, i PSNR dei tre canali rosso, verde e blu con l'interpolazione bilineare sono:

CANALE	MSE	PSNR (DB)
Rosso	63,4	30,11
Verde	29,5	33,43
Blu	75,3	29,36
Media semplice degli MSE	56,1	30,64
MSE pesato (0,299 R + 0,587 G + 0,114 B)	44,9	31,61

Tabella 2. PSNR per canale e due modi di riassumerlo in un solo valore.

Andamento di MSE e PSNR

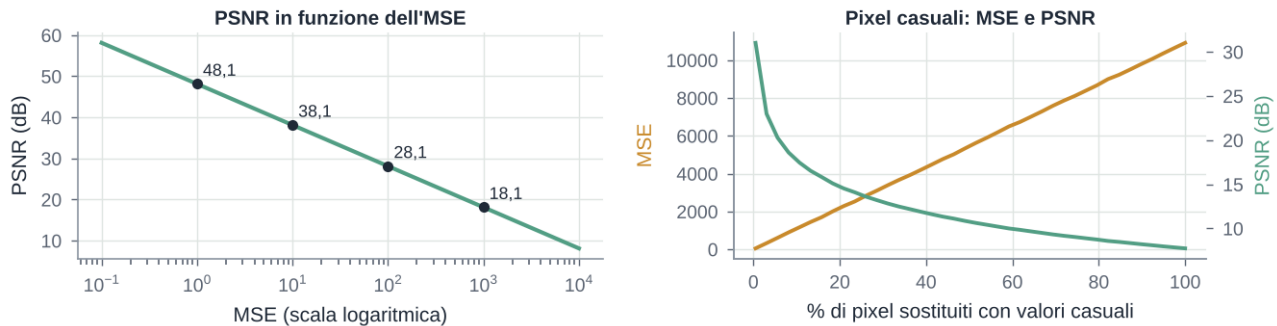


Figura 3. A sinistra il PSNR in funzione dell'MSE (8 bit): ogni fattore 10 nell'MSE vale 10 dB. A destra l'esperimento in cui una frazione crescente di pixel della fotografia viene sostituita con valori casuali: l'MSE cresce linearmente, il PSNR scende in modo logaritmico, molto in fretta all'inizio.

I limiti di MSE e PSNR

MSE e PSNR sono molto usati perché semplici da calcolare, ma non sempre rispecchiano il giudizio del sistema visivo umano (HVS):

- la sensibilità dell'HVS agli errori è diversa per diversi tipi di errore e dipende anche dal contesto visivo, mentre l'MSE tratta tutti gli errori allo stesso modo;
- due immagini distorte possono avere tipi di errore molto diversi pur avendo lo stesso MSE;
- entrambe le misure sono fortemente influenzate anche da movimenti spaziali «impercettibili»: traslazioni, rotazioni, scambi di righe o colonne.



Figura 4. Quattro immagini a confronto (ritaglio). MSE e PSNR rispetto all'originale: luminosità più 10 con MSE 99,6 e PSNR 28,1 dB; rumore con MSE 97,8 e PSNR 28,2 dB; traslazione di 1 pixel con MSE 251,6 e PSNR 24,1 dB.

La figura mostra i tre casi. Una luminosità aumentata di 10 e un rumore con scarto di 10 hanno MSE e PSNR quasi uguali (circa 28,1 dB) ma si percepiscono in modo molto diverso. Una traslazione di **un solo pixel**, visivamente identica all'originale, ha MSE 251,6 e PSNR 24,1 dB, cioè **peggiore** delle altre due.

DA RICORDARE

$PSNR = 10 \cdot \log_{10}(MAX^2 / MSE)$, con MSE media dei quadrati delle differenze pixel per pixel. Più è alto, più l'immagine è simile all'originale; ogni 6 dB l'errore si dimezza. È una misura comoda ma non sempre fedele alla percezione.

Il colore

Che cos'è la luce, come l'occhio la trasforma in colore e come si specifica il colore in modo oggettivo: spettro, coni, sintesi additiva, diagramma CIE e spazio L*a*b*.

La luce e lo spettro

Nel 1666 Isaac Newton osservò che un raggio di luce bianca che attraversa un prisma di vetro non esce bianco: si ottiene uno **spettro** di colori che vanno dal violetto al rosso. La luce può quindi essere scomposta in onde luminose di tipo diverso. I colori sono legati alle **lunghezze d'onda** (Newton, Huygens), che si misurano in nanometri: un nanometro è un metro diviso un miliardo, 10^{-9} m.

Quello che il nostro occhio percepisce è solo una piccola porzione dello spettro elettromagnetico:

RADIAZIONE	LUNGHEZZA D'ONDA (ORDINE DI GRANDEZZA)
Onde radio	da circa 1 m a molti chilometri
Microonde	da circa 1 mm a 1 m
Infrarosso	da circa 780 nm a 1 mm
Luce visibile	da circa 380 a 780 nm
Ultravioletto	da circa 10 a 380 nm
Raggi X	da circa 0,01 a 10 nm
Raggi gamma e cosmici	meno di 0,01 nm

Tabella 1. Lo spettro elettromagnetico (valori di riferimento). Le slide riportano gli stessi intervalli con esponenti diversi; qui sono dati in forma usuale.

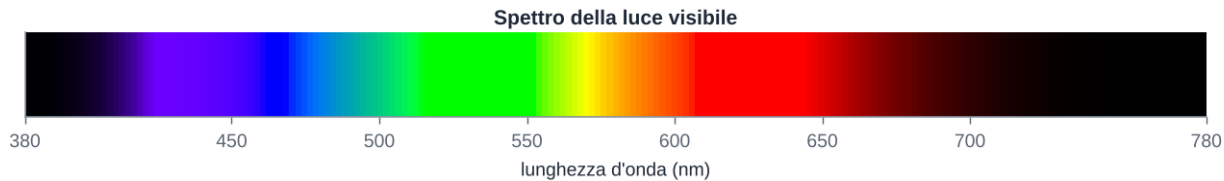


Figura 1. La porzione visibile dello spettro. Per comodità la si divide in sei regioni: violetto, blu, verde, giallo, arancio e rosso; le bande non hanno la stessa ampiezza e sfumano in quelle vicine.

L'occhio percepisce come colore di un oggetto la luce che l'oggetto **riflette**. Se un oggetto riflette tutte le lunghezze d'onda, è percepito come **bianco**; se ne assorbe quasi tutte, come nero. Un oggetto che riflette le lunghezze d'onda da 500 a 570 nm e assorbe tutto il resto è percepito di colore **verde**.

Per descrivere la luce bastano tre grandezze: la **radianza**, la quantità di luce emessa dalla sorgente; la **luminanza**, la misura dell'energia percepita dall'osservatore; la **brillantezza** (*brightness*), un valore soggettivo che indica la sensazione di colore.

I coni e la teoria tricromatica

Nella retina ci sono tre tipi di coni (capitolo 3): di tipo **S** (*short*), sensibili alle lunghezze d'onda corte, cioè ai colori bluastri; di tipo **M** (*middle*), sensibili alle lunghezze medie, colori verdastri; di tipo **L** (*long*), sensibili alle lunghezze lunghe, colori rossastri. Tutti i primati hanno questi tre tipi di cellule; gli altri mammiferi ne hanno solo due; gli uccelli ne hanno cinque.

Secondo la **teoria del tristimolo** (Young, 1802) tutti i colori si possono ottenere «mescolando» tre colori fondamentali in proporzioni diverse. L'ipotesi è vera soltanto in prima approssimazione: con tre primari fissi non si ottengono tutti i colori visibili (si vedrà con il diagramma CIE).

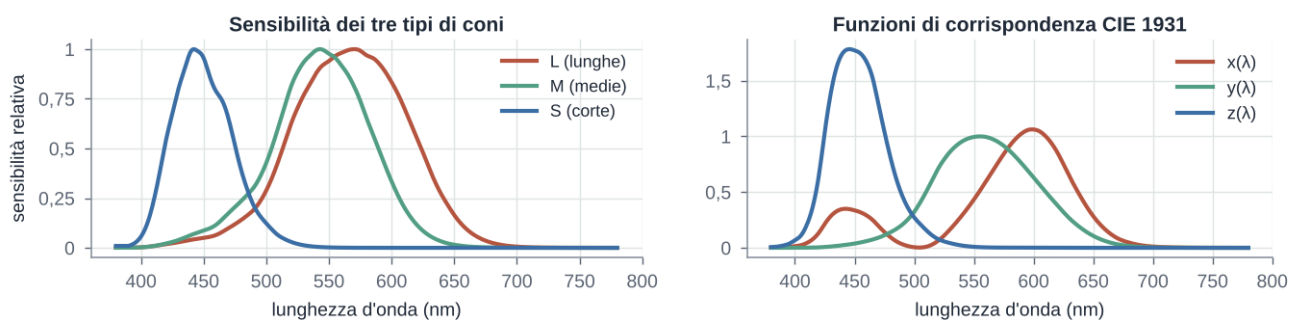


Figura 2. A sinistra la sensibilità dei tre tipi di coni (massimi vicino a 570, 543 e 442 nm). A destra le tre funzioni di corrispondenza dei colori dell'osservatore standard CIE 1931, da cui si ottengono i valori di tristimolo.

Poiché l'occhio riduce ogni stimolo a tre numeri, spettri diversi possono produrre la stessa sensazione di colore: coppie di spettri con questa proprietà si chiamano **metameri**. Il cervello non è uno spettrometro: raramente si vedono colori puri, ma miscele di radiazione di ogni lunghezza d'onda, e si mantiene una percezione costante del colore di una superficie anche se cambia la luce che la illumina.

Il colore non ha una scala fisica

A differenza della lunghezza o del peso, non esiste una scala fisica per misurare il colore: è una questione di percezione. Il riconoscimento del colore dipende dalla luce, dagli oggetti che la riflettono e da occhi e cervello dell'osservatore. Le differenze di colore dipendono da:

- la **sorgente luminosa** (luce solare, lampade a incandescenza, neon);
- l'**osservatore** (sensibilità, vista);
- le **dimensioni**: colori su superfici grandi appaiono più luminosi e vivaci (effetto area);
- lo **sfondo**: un colore appare più cupo su uno sfondo scuro (effetto contrasto, capitolo 5);
- la **direzione** di osservazione e di illuminazione, per esempio nelle vernici metallizzate.

Sintesi additiva e primari CIE

Illuminando una superficie bianca con una o più luci monocromatiche si ottiene per riflessione una radiazione colorata: è la **sintesi additiva**. Qualsiasi luce di spettro complesso può essere ottenuta come somma di tre luci monocromatiche agenti insieme. Nel 1931 la CIE (*Commission Internationale de l'Éclairage*) scelse come primari tre radiazioni con lunghezze d'onda il più possibile distanziate nello spettro:

PRIMARIO	LUNGHEZZA D'ONDA (CIE 1931)
Rosso	700,0 nm
Verde	546,1 nm
Blu	435,8 nm

Tabella 2. I primari standard del 1931. Le slide ricordano che dati sperimentali successivi, del 1964, hanno indicato valori lievemente diversi: i primari non sono unici.

Il diagramma di cromaticità CIE

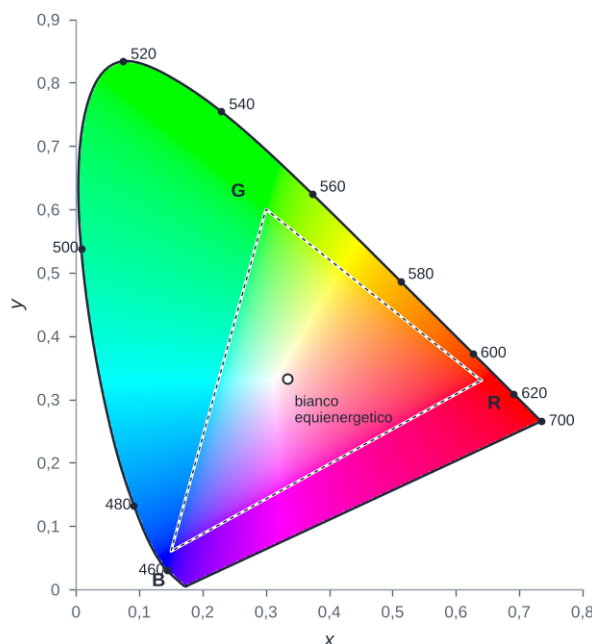
La retina ha tre tipi di coni, quindi un colore è descritto da tre numeri: i **valori di tristimolo** X , Y , Z dello spazio CIE 1931. Si ottengono integrando lo spettro $L(\lambda)$ della luce per le tre funzioni di corrispondenza del colore dell'osservatore standard:

$$X = \int L(\lambda) \cdot x(\lambda) d\lambda, \quad Y = \int L(\lambda) \cdot y(\lambda) d\lambda, \quad Z = \int L(\lambda) \cdot z(\lambda) d\lambda$$

con λ da 380 a 780 nm. La funzione $y(\lambda)$ coincide con la sensibilità luminosa dell'occhio, perciò Y è la **luminanza**. Dividendo per la somma si tolgono le informazioni sulla luminosità e si ottengono le **coordinate di cromaticità**:

$$x = X / (X + Y + Z), \quad y = Y / (X + Y + Z), \quad z = 1 - x - y$$

Le slide, e gli appunti, chiamano x «quantità di rosso» e y «quantità di verde»: è un modo intuitivo ma non esatto, perché x e y non sono i valori di due primari ma i rapporti tra tristimoli. Solo x e y sono indipendenti, perché z si ricava dagli altri due. La rappresentazione grafica di y in funzione di x è il **diagramma cromatico CIE**: una «fetta» dello spazio colorimetrico completo XYZ.



Il bordo curvo

contiene i colori puri di ogni lunghezza d'onda (in nm); i colori sul bordo non contengono bianco.

Il centro

è il bianco: punto di uguale energia, $x = y = 1/3$.

Un segmento

tra due punti contiene tutte le miscele dei due colori; unendo un colore al bianco si ottengono le sue tonalità.

Un triangolo

con vertici R, G, B contiene i colori ottenibili mescolando quei tre primari (gamut). Quello tratteggiato è il gamut tipico di un monitor sRGB: non copre tutta l'area.

Figura 3. Il diagramma cromatico CIE 1931 (colori approssimati: lo schermo o la stampa non possono riprodurre i colori fuori dal proprio gamut, quindi quelli vicini al bordo sono solo indicativi).

Il **color gamut** di un dispositivo è l'insieme dei colori che può produrre. Con tre primari è un triangolo, che non copre tutta l'area del diagramma: non tutti i colori si ottengono mescolando R, G e B. La stampante ha un gamut diverso, di forma irregolare e in genere più piccolo di quello del monitor, perché il modo di mescolare i colori è diverso (sottrattivo anziché additivo, capitolo 15).

Lo spazio CIE L*a*b*

Il difetto principale del sistema CIE XYZ, e dei modelli che ne derivano, è che non è **percettivamente uniforme**: due coppie di colori a uguale distanza numerica possono apparire a distanze diverse. Nel 1976 la CIE ha standardizzato due spazi percettivamente uniformi: CIE L*u*v* e **CIE L*a*b*** (le lettere u e v non hanno relazione con le componenti video U e V). In L*a*b*, L* è la **luminosità**, da 0 (nero) a 100 (bianco), mentre a* (asse verde, rosso) e b* (asse blu, giallo) sono le coordinate di cromaticità. Si ottengono da X, Y, Z e dal punto bianco (X_n, Y_n, Z_n):

$$L^* = 116 \cdot f(Y/Y_n) - 16, \quad a^* = 500 \cdot [f(X/X_n) - f(Y/Y_n)], \quad b^* = 200 \cdot [f(Y/Y_n) - f(Z/Z_n)]$$

dove $f(t)$ è la radice cubica di t per t maggiore di circa 0,009 (le slide indicano 0,01) e una funzione lineare per valori più piccoli. La radice cubica rispetta la risposta non lineare dell'occhio. La differenza tra due colori è la distanza nello spazio:

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

Uguali differenze numeriche corrispondono a uguali differenze percepite, e questo è il pregio dello spazio. Come ordine di grandezza, una differenza ΔE^* intorno a 1 è appena percettibile; i valori accettabili dipendono dall'industria (cosmetici, packaging, tessile, automobili). Per esempio, il rosso sRGB puro (255, 0, 0) ha $L^* = 53,2$, $a^* = 80,1$, $b^* = 67,2$; un rosso leggermente più scuro (242, 26, 13) ha $L^* = 51,5$, $a^* = 74,7$, $b^* = 61,7$, a distanza $\Delta E^* \approx 7,9$. Esiste anche la rappresentazione cilindrica **L*C*h**, con croma $C^* = \sqrt{a^{*2} + b^{*2}}$ e tinta h come angolo.

IL BIANCO E IL NERO

Il «nero matematico» ha tristimolo $X = Y = Z = 0$, ma nessun corpo reale lo raggiunge: i neri reali riflettono qualche percento della luce (il velluto nero è tra i più scuri). Anche il «bianco» va specificato: bianco equienergetico ($X = Y = Z = 1$, $x = y = 1/3$) oppure il bianco dell'illuminante scelto, tipicamente D65 (luce del giorno). Per misurare un colore bisogna dunque fissare osservatore e illuminante.

DA RICORDARE

Il colore dipende dalla luce riflessa, da tre tipi di coni e dal cervello. Si specifica con X, Y, Z o con le coordinate x, y del diagramma CIE; l'area coperta da tre primari è il gamut. Lo spazio L*a*b* è percettivamente uniforme e permette di misurare le differenze di colore con ΔE^* .

Gli spazi di colore RGB e CMY

Il modello additivo dei monitor, il modello sottrattivo della stampa e il rapporto tra i due: cubo RGB, colori complementari, conversione in CMY e CMYK, colori sicuri per il web.

Perché servono i modelli di colore

Lo scopo di un **modello** (o **spazio**, o sistema) **di colore** è consentire di specificare i colori in modo standardizzato, di solito con un sistema di coordinate tridimensionali (tre sono infatti le caratteristiche che definiscono un colore) o con un suo sottospazio, in cui ogni colore è un punto. I modelli più usati in elaborazione delle immagini sono orientati all'acquisizione e alla restituzione (RGB, CMY), alla trasmissione (YUV, YIQ), alla compressione (YCbCr) o all'elaborazione mediante trattamento del colore (RGB, HSI, HSV, LUV). In letteratura le lettere hanno significati ambigui (I e V indicano cose diverse in modelli diversi) e le equazioni di conversione esistono in versioni diverse: conviene sempre verificare le convenzioni usate.

Il modello additivo RGB

Nel modello **RGB** ogni colore si ottiene sommando tre luci, rosso (*Red*), verde (*Green*) e blu (*Blue*). Sommando le luci a due a due si ottengono i colori **secondari**: giallo (*Y*, *yellow* = $R + G$), magenta ($M = R + B$) e ciano ($C = G + B$). Sommando le tre luci si ottiene il bianco. È la **sintesi additiva**, su cui si basano monitor e televisori. Se ogni componente è una coordinata cartesiana, il modello è un **cubo**: i contributi di R, G e B sono indipendenti, rappresentati da direzioni perpendicolari.



Figura 1. Il cubo RGB. Ogni colore è un punto del cubo; la retta che congiunge il nero e il bianco, in cui $R = G = B$, è la retta dei grigi.

Con 8 bit per canale ogni componente è un intero da 0 a 255 e un pixel è una terna (R, G, B). Si hanno $256 \times 256 \times 256 = 16\,777\,216$ colori (*true color*, 24 bit per pixel). In HTML e CSS la terna si scrive in esadecimale come #RRGGBB: per esempio #FF8000 è (255, 128, 0), un arancione.

COLORI PRIMARI

Il termine «primario» è stato spesso frainteso: si pensava che tre primari standard, mescolati in proporzioni diverse, producessero **tutti** i colori visibili. Non è così, a meno di far variare anche la lunghezza d'onda dei primari (ma allora non sarebbero tre primari fissi): è il limite del gamut visto nel capitolo 14. I tre colori R, G, B sono detti primari, i tre ottenuti a due a due sono detti secondari.

Pro e contro. L'RGB è semplice da usare e da implementare in software e hardware ed è di fatto uno standard. È però poco comodo dal punto di vista percettivo: guardando un colore in natura è difficile capire in quale proporzione contribuiscano R, G e B.

Colori complementari

Due colori sono **complementari** quando la loro somma dà il bianco. Poiché $R + G + B = W$:

COLORE	COMPLEMENTARE	PERCHÉ
Giallo (R + G)	Blu	$R + G + B = W$
Magenta (R + B)	Verde	$R + B + G = W$
Ciano (G + B)	Rosso	$G + B + R = W$

Tabella 1. I tre colori secondari e i rispettivi complementari.

Il modello sottrattivo CMY

Il colore si può ottenere anche mescolando sostanze colorate (pigmenti, coloranti, inchiostri), il cui comportamento si può simulare con **filtri colorati**. Un filtro assorbe selettivamente una parte della luce bianca e lascia passare il resto; assume il colore della radiazione complementare di quella che assorbe. Sovrapponendo filtri si *sottrae* luce, perciò la sintesi si dice **sottrattiva**: parte dalla luce bianca e ne toglie componenti. I primari sottrattivi sono **ciano**, **magenta** e **giallo** (CMY), complementari di R, G, B.

SOVRAPPOSIZIONE DI FILTRI	RISULTATO	LUCE CHE PASSA
Giallo + Magenta	Rosso	il giallo toglie il blu, il magenta toglie il verde: resta il rosso
Giallo + Ciano	Verde	il giallo toglie il blu, il ciano toglie il rosso: resta il verde
Magenta + Ciano	Blu	il magenta toglie il verde, il ciano toglie il rosso: resta il blu
Giallo + Magenta + Ciano	Nero (K)	tutte le componenti sono assorbite

Tabella 2. Sintesi sottrattiva con tre filtri.



Figura 2. A sinistra la sintesi additiva, a destra la sottrattiva: i colori secondari dell'una sono i primari dell'altra, e viceversa.

Su questo principio si basano la fotografia a colori e la stampa con inchiostri: quando non c'è inchiostro sulla carta la luce riflessa è bianca, quando ci sono tutti e tre i colori la luce è in linea di principio assorbita e la carta appare nera. In pratica l'assorbimento completo è difficile da ottenere, quindi si usa un quarto inchiostro, il nero (*black*): è il modello **CMYK**.

Conversione tra RGB, CMY e CMYK

CMY è il modello **complementare** (duale) di RGB. Con componenti da 0 a 255:

$$C = 255 - R, \quad M = 255 - G, \quad Y = 255 - B$$

COLORE	RGB	CMY
Giallo	(255, 255, 0)	(0, 0, 255)
Ciano	(0, 255, 255)	(255, 0, 0)
Bianco	(255, 255, 255)	(0, 0, 0)
Nero	(0, 0, 0)	(255, 255, 255)
Arancione	(255, 128, 0)	(0, 127, 255)

Tabella 3. Esempi di conversione RGB, CMY.

Per passare a CMYK si estrae la parte di nero comune ai tre canali, $K = \min(C, M, Y)$, e la si toglie dagli altri tre. Con componenti normalizzate tra 0 e 1, se $K < 1$:

$$C' = (C - K) / (1 - K), \quad M' = (M - K) / (1 - K), \quad Y' = (Y - K) / (1 - K)$$

Esempio. Il colore (200, 100, 50) ha CMY = (55, 155, 205), cioè (0,216; 0,608; 0,804) normalizzato. Quindi $K = 0,216$ e $C' = 0$, $M' = (0,608 - 0,216) / 0,784 = 0,5$, $Y' = (0,804 - 0,216) / 0,784 = 0,75$: CMYK = (0; 0,5; 0,75; 0,216). Questa è una conversione di principio; le stampanti reali usano profili di colore più complessi.

Colori sicuri per il web

Si sa che 40 dei 256 colori usati tipicamente per il web vengono processati in modo diverso da sistemi operativi diversi, mentre 216 sono comuni alla maggior parte dei sistemi. Questi 216 colori sono diventati lo standard di fatto dei **colori sicuri**. Ogni componente può valere solo i sei numeri esadecimali 00, 33, 66, 99, CC, FF, cioè 0, 51, 102, 153, 204, 255 in decimale: $6 \times 6 \times 6 = 216$ combinazioni. Per esempio #33CCFF è sicuro, #12A3FE no.

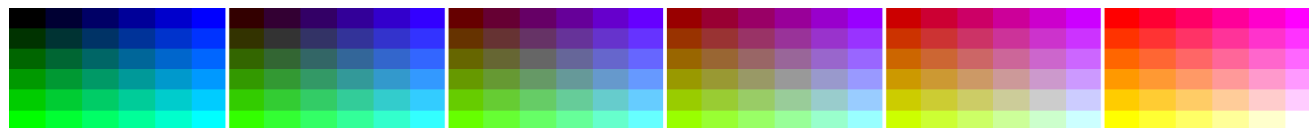


Figura 3. I 216 colori sicuri: sei blocchi, uno per ciascun valore di rosso; dentro il blocco il verde cresce verso il basso e il blu verso destra.

DA RICORDARE

RGB è additivo (luci, monitor), CMY è sottrattivo (filtri e inchiostri, stampa); sono complementari: $C = 255 - R$, e così via. La retta dei grigi del cubo RGB è la diagonale con $R = G = B$. CMYK aggiunge un quarto inchiostro nero. Gli spazi RGB e CMY non sono i più adatti a descrivere tinta, saturazione e luminosità: per questo si usano HSV e YCbCr.

Gli spazi di colore HSV e Munsell

Tinta, saturazione e valore: i modelli che descrivono il colore come lo percepiamo, dal modello del pittore all'atlante di Munsell.

Dalla fisica al modello del pittore

Gli spazi RGB e CMY (capitolo 15) non sono sempre i più convenienti: sono legati ai dispositivi, non a come descriviamo un colore. Sono disponibili rappresentazioni con componenti legate alla **tinta**, alla **saturazione** e alla **luminosità**:

- la **tinta** (*hue*) dice quale colore è presente (rosso, verde, giallo, e così via) e si può correlare alla lunghezza d'onda dominante della sorgente;
- la **saturazione** dice quanto è vivo il colore (molto forte, pastello, vicino al bianco) e si correla alla purezza dello spettro della sorgente;
- il **valore** (o luminanza, luminosità) è legato all'intensità della luce: quanto il colore è bianco, grigio o nero.

Il fondamento fisico è nei **metameri** (capitolo 14): ogni spettro ha un metamero particolarmente semplice, formato da un picco di energia e_{picco} sopra un livello costante e_{media} . La lunghezza d'onda del picco è responsabile del colore percepito, cioè la **tinta**. Il livello costante dà un contributo «bianco» al colore percepito, e il rapporto

$$S = (e_{\text{picco}} - e_{\text{media}}) / (e_{\text{picco}} + e_{\text{media}})$$

è la **saturazione**: meno luce bianca significa saturazione più alta. Infine e_{media} è proporzionale al contenuto energetico della radiazione e misura la **luminosità** o **valore**.

Il modello del pittore

Per ciascuna tinta si può disegnare un **triangolo**: un vertice è il colore puro (saturazione 1, valore massimo), un secondo vertice è il bianco puro (saturazione 0, valore massimo), il terzo è il nero (valore 0). Lungo il lato colore puro e bianco ci sono le *tinte* (il colore mescolato con bianco), lungo il lato colore puro e nero le *ombre* (il colore con nero), lungo il lato bianco e nero i *grigi*. È il modello usato nel mondo dell'arte e insegnato nelle accademie.

Mettendo insieme tutti i triangoli, uno per tinta, facendo coincidere le linee dei grigi come in una girandola, si ottiene lo spazio **HSV** (o HSI), una piramide a base esagonale (o un cono):

- **H** (*hue*, tinta) percorre tutti i colori in sequenza, come un angolo da 0° a 360°: rosso 0°, giallo 60°, verde 120°, ciano 180°, blu 240°, magenta 300°, e poi di nuovo rosso;
- **S** (saturazione) va da 0 al centro (bianco o grigio) a 1 sulla periferia (colore puro);
- **V** (valore) va da 0 (nessuna energia emessa, nero) al massimo.

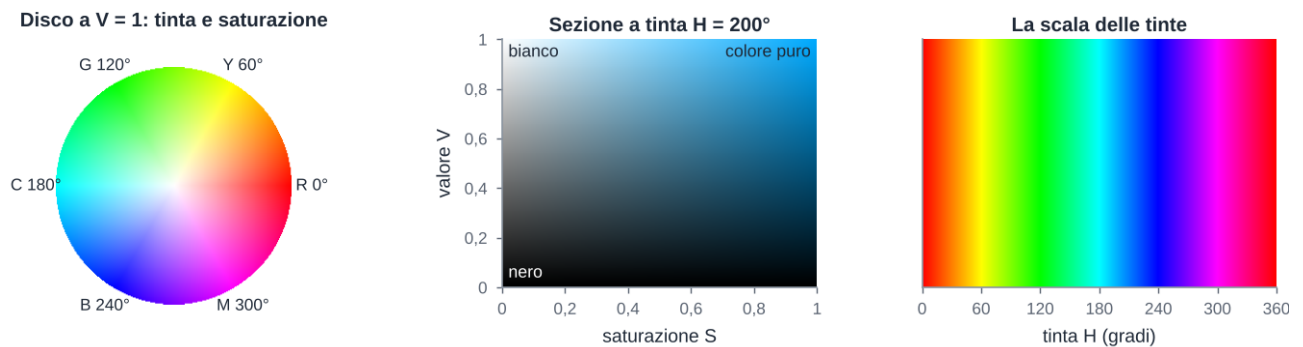


Figura 1. Lo spazio HSV. A sinistra il disco a valore massimo (tinta come angolo, saturazione come raggio), al centro la sezione a tinta fissa (il «triangolo del pittore» in forma rettangolare), a destra la scala delle tinte.

CONVERSIONE DA RGB A HSV

Con R, G, B tra 0 e 1, $M = \max(R, G, B)$ e $m = \min(R, G, B)$:

$$V = M, \quad S = (M - m) / M \quad (\text{con } S = 0 \text{ se } M = 0)$$

La tinta in gradi dipende da quale componente è la massima: $H = 60^\circ \cdot ((G - B) / (M - m))$ se $M = R$; $H = 60^\circ \cdot (2 + (B - R) / (M - m))$ se $M = G$; $H = 60^\circ \cdot (4 + (R - G) / (M - m))$ se $M = B$; se l'angolo risulta negativo si aggiungono 360° . Se $M = m$ (un grigio) la tinta non è definita e si pone 0.

COLORE	RGB	H	S	V
Rosso	(255, 0, 0)	0,0°	1,000	1,000
Giallo	(255, 255, 0)	60,0°	1,000	1,000
Verde	(0, 255, 0)	120,0°	1,000	1,000
Arancione	(255, 128, 0)	30,1°	1,000	1,000
Marrone chiaro	(200, 100, 50)	20,0°	0,750	0,784
Grigio	(128, 128, 128)	0,0°	0,000	0,502

Tabella 1. Conversioni da RGB a HSV (S e V tra 0 e 1). Per il marrone chiaro: $M = 200$, $m = 50$, $H = 60^\circ \cdot (100 - 50) / 150 = 20^\circ$, $S = 150 / 200 = 0,75$, $V = 200 / 255 = 0,784$.

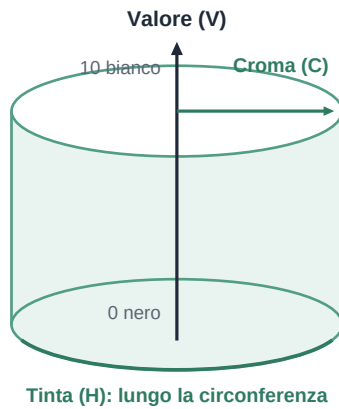
Il cubo RGB e la piramide HSV si trasformano l'uno nell'altra con semplici algoritmi, **non lineari**.

	PRO	CONTRO
Modello del pittore (HSV)	intuitivo; percettivamente significativo: H, S e V hanno una interpretazione diretta nelle nostre percezioni	modello non lineare; perché una piramide esagonale? quali sono i «colori base»?

Tabella 2. Discussione del modello HSV.

Il sistema di Munsell

La rappresentazione dei colori ha una lunga storia: dalla ruota dei colori (Blanc, Newton, Chevreul), che organizza i colori in uno schema simmetrico, fino ad Albert Munsell, che per primo li colloca in uno **spazio tridimensionale**. Da qui nasce poi la colorimetria con i sistemi CIE, che danno una misura fisica del colore. Nel **sistema di Munsell** un colore è dato da tre numeri, (H, C, V) secondo le slide, (H, V, C) nella notazione usuale:



Tinta (Hue)

rosso, giallo, verde, blu, porpora: 5 tinte principali e 5 intermedie, ciascuna divisa in 10 passi

Valore (Value)

chiarezza, da 0 (nero) a 10 (bianco)

Croma (Chroma)

intensità del colore, da 0 (grigio) verso valori alti, senza un massimo fisso

Figura 2. Il solido di Munsell: la tinta cambia lungo la circonferenza, il valore lungo la verticale, il croma a partire dal centro.

Il solido dei colori che ne risulta non è un cilindro regolare, perché i colori più saturi raggiungono croma diverso a seconda della tinta e del valore: è irregolare proprio perché i passi sono scelti in modo da essere **percettivamente uniformi**. L'atlante di Munsell è definito per l'osservatore CIE 1964 e per l'illuminante D65: le sue proprietà percettive non valgono più in condizioni diverse e i confronti visivi vanno fatti in condizioni controllate. In notazione compatta si scrive per esempio «5R 4/14»: tinta 5R (rosso), valore 4, croma 14.

DA RICORDARE

HSV descrive il colore con tinta (angolo), saturazione (distanza dall'asse dei grigi) e valore (altezza). Il modello deriva dal «modello del pittore» e dai metameri. Munsell ordina i colori con tinta, valore e croma, in uno spazio percettivamente uniforme ma irregolare.

Gli spazi di colore YUV e YCbCr

Separare la luminosità dal colore: le rappresentazioni luminanza e cromaticità, le formule YUV e YCbCr, il sottocampionamento della cromaticità e il perché della loro importanza nella compressione.

Luminanza e cromaticità

Gli spazi di colore in cui una componente rappresenta la luminosità e le altre due sono legate alla colorazione si chiamano rappresentazioni **luminanza e cromaticità**. La **luminanza** fornisce una versione in scala di grigi dell'immagine; la **cromaticità** fornisce le informazioni «extra» che trasformano l'immagine in grigi in un'immagine a colori. Sono particolarmente importanti nella **compressione**: l'occhio umano è più sensibile alla luminosità che ai colori (capitolo 3), quindi si possono «spendere» molti bit per registrare la luminosità e risparmiarne sulle cromaticità.

Lo spazio YUV

Con **YUV** non si indica un singolo spazio ma una **famiglia** di spazi con queste caratteristiche, usata per la codifica di immagini e video analogici. Come esempio si prende lo spazio definito nello standard ITU R BT.601. La luminosità è una combinazione lineare delle intensità dei tre canali RGB, con pesi che riflettono la sensibilità dell'occhio, massima per il verde:

$$Y = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B$$

La cromaticità è definita come la differenza tra il colore e un bianco di riferimento alla stessa luminosità, opportunamente pesata:

$$U = 0,564 \cdot (B - Y) = -0,169 \cdot R - 0,331 \cdot G + 0,5 \cdot B$$

$$V = 0,713 \cdot (R - Y) = 0,5 \cdot R - 0,419 \cdot G - 0,081 \cdot B$$

Se R, G, B sono compresi tra 0 e 1, Y è compreso tra 0 e 1, mentre U e V variano tra $-0,5$ e $0,5$. Quando $R = G = B$ si ha $U = V = 0$: si ottengono solo grigi, senza cromaticità.

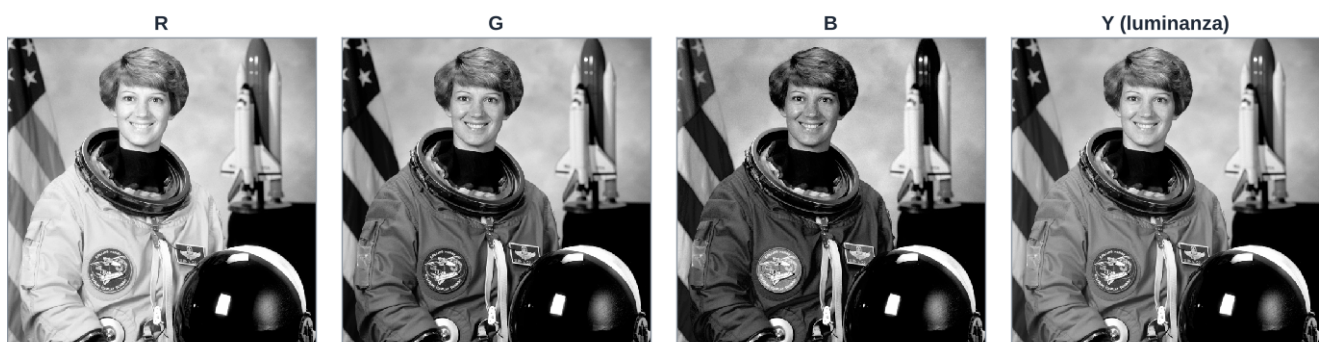


Figura 1. I canali R, G, B della fotografia dell'astronauta e la luminosità Y . Il canale verde è il più simile a Y , perché pesa di più (0,587): la luminosità è mantenuta soprattutto nel verde.



Figura 2. Le tre componenti dello spazio YCbCr. Quasi tutto il dettaglio è in Y; Cb e Cr sono molto più lisce, e per questo si possono ridurre di risoluzione senza accorgersene.

Lo spazio YCbCr

Gli spazi **YCbCr** sono la controparte digitale degli YUV e si ottengono normalizzando e quantizzando i canali di YUV. Se R , G , B sono interi tra 0 e 255, Y è lo stesso e si ottengono C_b e C_r traslando U e V di 128, così che tutti e tre i canali stiano tra 0 e 255. C_b è la cromaticanza del blu e C_r quella del rosso.

$$Y = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B$$

$$C_b = U + 128 = -0,169 \cdot R - 0,331 \cdot G + 0,5 \cdot B + 128$$

$$C_r = V + 128 = 0,5 \cdot R - 0,419 \cdot G - 0,081 \cdot B + 128$$

COLORE	RGB	Y	CB	CR
Rosso	(255, 0, 0)	76,2	85,0	255,0
Verde	(0, 255, 0)	149,7	43,6	21,3
Blu	(0, 0, 255)	29,1	255,0	107,3
Bianco	(255, 255, 255)	255,0	128,0	128,0
Nero	(0, 0, 0)	0,0	128,0	128,0
(200, 100, 50)	(200, 100, 50)	124,2	86,2	182,0
Grigio	(128, 128, 128)	128,0	128,0	128,0

Tabella 1. Esempi di conversione RGB, YCbCr. Per il rosso il calcolo dà $C_r = 255,45$ e per il blu $C_b = 255,42$: nella tabella i valori sono limitati a 255. Grigi e bianco hanno sempre $C_b = C_r = 128$.

La trasformazione inversa è

$$R = Y + 1,402 \cdot (C_r - 128), \quad B = Y + 1,772 \cdot (C_b - 128)$$

$$G = Y - 0,344 \cdot (C_b - 128) - 0,714 \cdot (C_r - 128)$$

Lo spazio YCbCr è largamente utilizzato nella compressione: lo ritroveremo nel formato JPEG (capitolo 29).

Sottocampionamento della cromaticanza

Poiché l'occhio ha meno risoluzione per il colore che per la luminosità, si registra Y a piena risoluzione e Cb e Cr a risoluzione ridotta. Si usano tre schemi, indicati con tre numeri:

SCHEMA	SIGNIFICATO	CAMPIONI PER OGNI 4 PIXEL (Y, CB, CR)	DATI RISPETTO A 4:4:4
4:4:4	nessuna riduzione	$4 + 4 + 4 = 12$	100 per cento
4:2:2	Cb e Cr dimezzati in orizzontale	$4 + 2 + 2 = 8$	circa 67 per cento
4:2:0	Cb e Cr dimezzati in orizzontale e in verticale	$4 + 1 + 1 = 6$	50 per cento

Tabella 2. I principali schemi di sottocampionamento.

Con 4:2:0 si dimezza la quantità di dati passando da 3 a 1,5 campioni per pixel, senza una perdita percepibile. Il confronto seguente lo verifica: sulla fotografia dell'astronauta si riducono a metà risoluzione (media 2×2 e poi interpolazione bilineare) prima Cb e Cr e poi, in alternativa, Y.

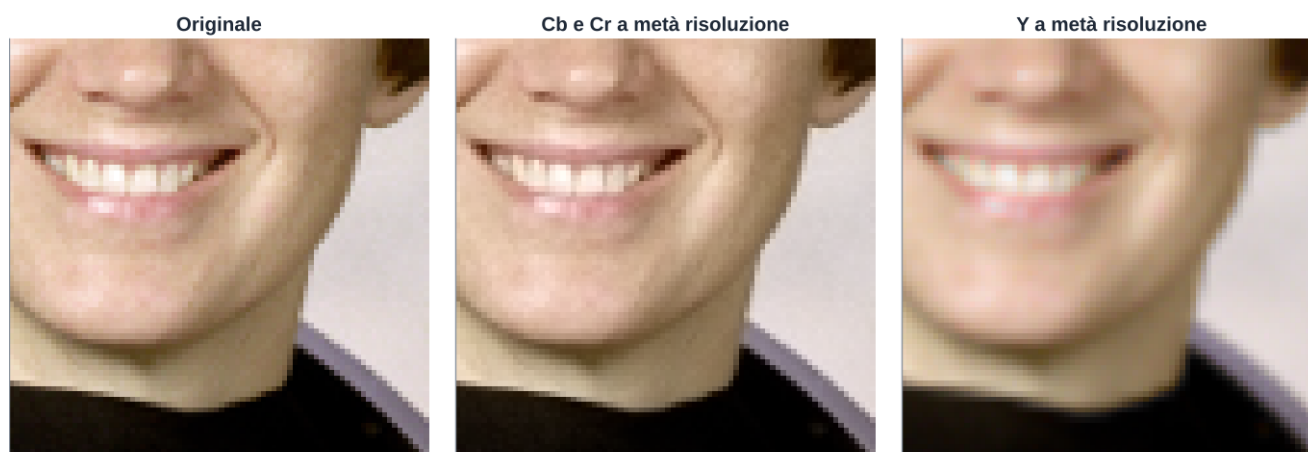


Figura 3. Ritaglio di 80×80 pixel (volto). Ridurre Cb e Cr passa quasi inosservato (PSNR 40,5 dB sull'intera immagine); ridurre Y rovina i dettagli (PSNR 29,3 dB).

DA RICORDARE

$Y = 0,299 R + 0,587 G + 0,114 B$ è la luminanza; Cb e Cr sono differenze di colore centrate su 128. L'occhio vede meglio la luminanza che il colore, quindi la cromaticanza si può sottocampionare (4:2:0 dimezza i dati). È la base della compressione JPEG.

Immagini indicizzate, palette e reindexing

Come si memorizza un'immagine a colori usando poche etichette al posto di terne RGB, come si costruisce la palette e come si ordina per aiutare la compressione.

Più colori che pixel

Il modo con cui la memoria video mostra i colori usa 8 bit per il rosso, 8 per il verde e 8 per il blu: 24 bit per pixel e circa 16 milioni di colori (*true color*). Questo schema costoso **non** è quello con cui i colori sono conservati nella memoria di massa e compressi in formati come JPEG o GIF. Un motivo è che nelle immagini reali si hanno **più colori possibili che pixel**:

IMMAGINE	DIMENSIONE	PIXEL	COLORI POSSIBILI
Grande	2400 × 1800	4 320 000	16 777 216
Media	1600 × 1200	1 920 000	16 777 216
Piccola	800 × 600	480 000	16 777 216

Tabella 1. Pixel e colori possibili. Le slide indicano 4.520.000 pixel per l'immagine grande, ma $2400 \times 1800 = 4\,320\,000$.

Inoltre le immagini naturali hanno una **coerenza interna**: raramente si ha un colore diverso per ogni pixel. Ne segue la modalità a **colori indicizzati** (*indexed color*), detta anche a palette o a look up table.

L'idea della palette

Invece di ricordare per ogni pixel la terna (R, G, B), si ricorda per ogni pixel una **etichetta** (un indice) e una piccola tabella che associa ogni etichetta a una terna. La tabella si chiama **palette**, tavolozza, tabella di indicizzazione dei colori, look up table (LUT).

Matrice degli indici (2 bit)			Palette	
00	00	01	00	(255, 0, 0)
00	00	01	01	(0, 255, 0)
11	10	00	10	(255, 255, 255)
			11	(0, 0, 255)

Figura 1. Una immagine 3×3 con 4 colori distinti, memorizzata con una palette di 4 voci e indici da 2 bit.

Esempio. Per memorizzare i 9 pixel in true color servono $9 \times 3 \text{ byte} = 27 \text{ byte} = 216 \text{ bit}$. Con la palette servono $9 \text{ pixel} \times 2 \text{ bit} = 18 \text{ bit}$ per la matrice degli indici più $4 \text{ voci} \times 3 \text{ byte} = 12 \text{ byte} = 96 \text{ bit}$ per la palette: in totale $18 + 96 = 114 \text{ bit}$. Il risparmio, qui modesto per la piccola dimensione, diventa grande su immagini reali. In generale, con K colori nella palette, una immagine $M \times N$ occupa

$$M \cdot N \cdot \lceil \log_2 K \rceil + 24 \cdot K \text{ bit}$$

Per una immagine di 800×600 pixel con palette di 256 colori si ha $480\,000 \times 8 + 24 \times 256 = 3\,846\,144 \text{ bit}$, cioè 480 768 byte, contro i 1 440 000 byte del true color: circa un terzo.

Costruire la palette

I programmi commerciali e alcuni formati, come il GIF, adottano una palette di 256 colori. Si può usare:

- una palette **personalizzata** (*custom*), adattata all'immagine: se l'immagine true color ha meno di 256 colori, alcune voci vengono replicate; se ne ha di più, i colori vengono «ridotti» scegliendo 256 rappresentanti che garantiscano una buona qualità visiva (esistono numerosi algoritmi, anche proprietari, per questo scopo, come il taglio mediano usato qui);
- una palette **standard**: Mac, Windows, web safe (216 colori, capitolo 15), oppure ottimizzata. Con una palette standard di 256 colori alcuni potrebbero non essere usati nell'indicizzazione di una immagine true color, anche se questa ha più di 256 colori.



Figura 2. Una immagine indicizzata con palette ottimizzata (algoritmo del taglio mediano) con 256, 16, 8 e 4 colori.

Più piccola è la palette, più l'immagine si degrada: il PSNR rispetto all'originale è 34,8 dB con 256 colori, 25,4 dB con 16, 22,2 dB con 8 e 19,2 dB con 4. Con 16 colori e meno si vedono falsi contorni nelle zone sfumate (compare lo stesso effetto della quantizzazione del capitolo 9).

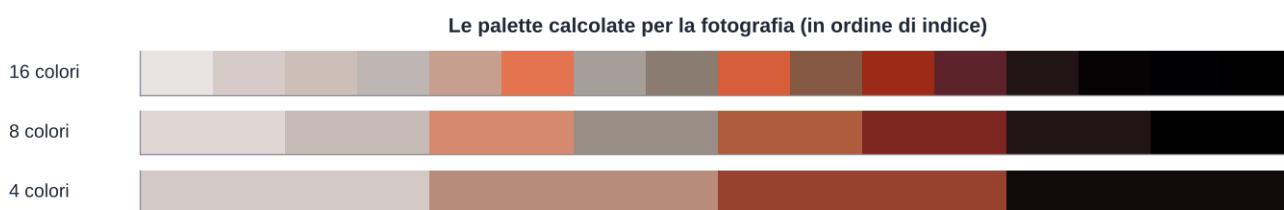


Figura 3. Le palette di 16, 8 e 4 colori dell'esempio, nell'ordine in cui compaiono nella tabella.

Il reindexing

Una palette è una tabella in cui l'ordine delle voci è arbitrario. Il **reindexing** consiste nel cambiare la posizione dei colori nella palette, e quindi l'indice corrispondente a ciascun colore, in modo da creare una matrice di indici che abbia **entropia minima**. L'idea generale: se ai colori di pixel adiacenti si associano indici numericamente **vicini**, la differenza tra gli indici diventa più piccola. Questo diminuisce l'entropia dei dati e garantisce un miglior rapporto di compressione con i formati che usano la codifica differenziale, che si basano cioè sulla differenza tra pixel vicini (capitolo 27).

Trovare la disposizione ottimale, cioè a entropia minima, è un problema **NP hard**: per una palette di M colori bisogna cercare tra $M!$ ordinamenti possibili, che per $M = 256$ sono circa 10^{507} . Gli algoritmi di reindexing cercano perciò soluzioni vicine alla migliore. Due strategie semplici sono l'ordinamento per **luminanza** e la costruzione di una catena in cui ogni colore è seguito dal colore più simile tra quelli rimasti.



Figura 4. La stessa immagine a 64 colori, con la matrice degli indici visualizzata in grigi per tre diversi ordinamenti della palette. Con indici vicini a colori simili l'immagine è regolare.

ORDINAMENTO DELLA PALETTE (64 COLORI)	ENTROPIA DELLA DIFFERENZA TRA PIXEL ADIACENTI (BIT PER PIXEL)
Casuale (media su 20 prove)	3,85
Per luminanza	2,87
Per colore più vicino	2,80

Tabella 2. Entropia delle differenze orizzontali tra indici adiacenti per tre ordinamenti, sulla fotografia dell'astronauta a 64 colori. La differenza viene dal solo cambio di etichette.

ATTENZIONE

L'entropia dell'istogramma degli indici (5,75 bit) non cambia con il reindexing, perché una permutazione delle etichette non cambia le frequenze. Quello che cambia è l'entropia delle **differenze** tra indici adiacenti, ed è questa che conta per la codifica differenziale. L'entropia è definita nel capitolo 27.

DA RICORDARE

Un'immagine indicizzata memorizza una matrice di indici e una palette di terne RGB. Si risparmia memoria perché i colori usati sono pochi e coerenti. Il reindexing ordina la palette in modo che colori simili abbiano indici vicini e le differenze tra pixel adiacenti diventino piccole.

L'istogramma di una immagine

La distribuzione dei toni di grigio, che cosa rivela sull'esposizione, e due operazioni per migliorare il contrasto: stretching ed equalizzazione.

Che cos'è l'istogramma

I pixel di un'immagine sono una «popolazione» su cui si possono calcolare le quantità statistiche descrittive usuali: media, mediana, varianza, deviazione standard, quartili, percentili. Particolarmente importante è la distribuzione delle frequenze dei toni di grigio, cioè l'**istogramma**.

ISTOGRAMMA

Per una immagine I di $M \times N$ pixel con L livelli di grigio, l'istogramma $H(k)$ è il **numero di pixel di valore k** , per $k = 0, \dots, L - 1$. La somma di tutti gli $H(k)$ è esattamente $M \times N$. Sull'asse x stanno i livelli di grigio (da 0, nero, al massimo, per esempio 255), sull'asse y la frequenza.

Dividendo per $M \cdot N$ si ottiene l'istogramma **normalizzato** $p(k) = n_k / (M \cdot N)$, che somma a 1 ed è la probabilità di trovare il livello k scegliendo un pixel a caso. Media e varianza dei toni di grigio si calcolano dall'istogramma:

$$m = \sum_k k \cdot p(k), \quad \sigma^2 = \sum_k (k - m)^2 \cdot p(k)$$

L'istogramma permette di capire a colpo d'occhio le caratteristiche dell'immagine.

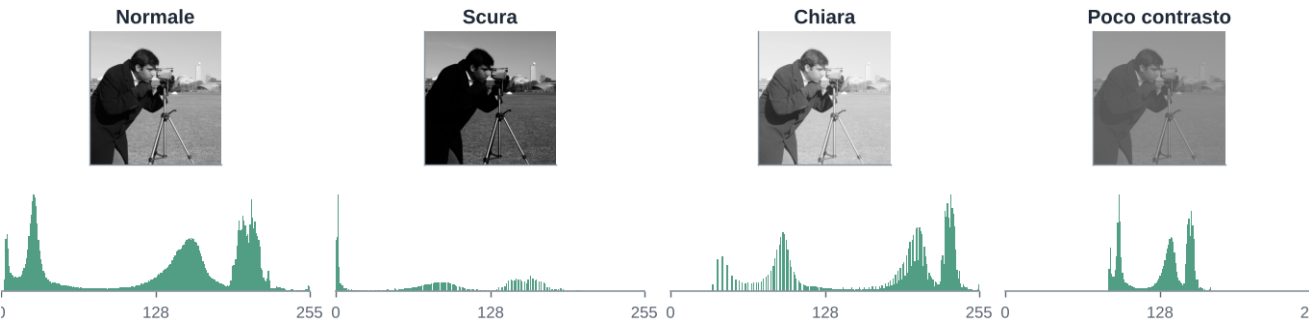


Figura 1. Quattro versioni della stessa fotografia con i rispettivi istogrammi: una immagine scura (sottoesposta) ha l'istogramma concentrato a sinistra, una chiara (sovraesposta) a destra, una a basso contrasto in una banda stretta.

ISTOGRAMMA	IMMAGINE
Denso a sinistra	scura, sottoesposta
Denso a destra	chiara, sovraesposta
Concentrato in un intervallo stretto	poco contrasto, grigia
Esteso su tutta la scala	buon contrasto, ma non è una garanzia di qualità

Tabella 1. Lettura di un istogramma.

UN ISTOGRAMMA NON IDENTIFICA L'IMMAGINE

L'istogramma **non tiene conto della distribuzione spaziale** dei pixel: dice quanti pixel hanno un certo valore, non dove stanno. Perciò immagini diverse possono avere lo stesso istogramma. Per esempio, se si mescolano a caso i pixel di una fotografia le frequenze non cambiano, ma l'immagine diventa rumore. Vale solo l'implicazione inversa: se due immagini hanno istogrammi *diversi*, allora sicuramente non sono identiche.

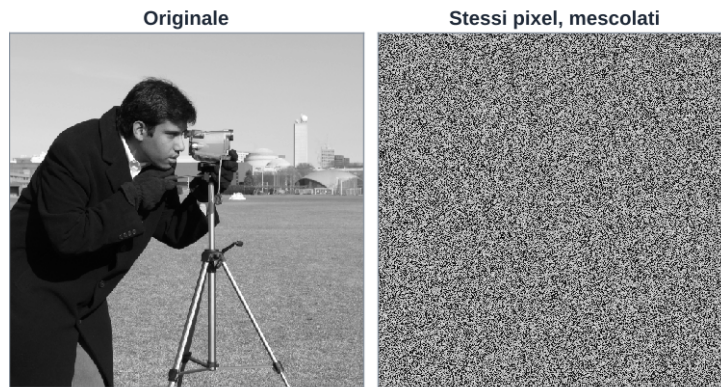


Figura 2. Le due immagini hanno esattamente lo stesso istogramma (le stesse frequenze di ogni livello di grigio), ma sono immagini del tutto diverse.

Espansione del contrasto (contrast stretching)

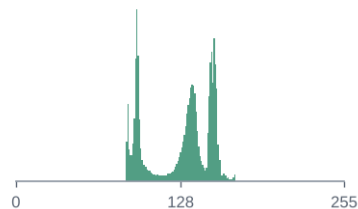
Se l'istogramma è concentrato in un intervallo limitato dei valori possibili, l'immagine ha poco contrasto.

L'**espansione del contrasto** serve ad aumentare la dinamica: si ridistribuiscono i valori, con una funzione che nella forma base è lineare, su tutto l'intervallo disponibile. Con $v_{\min}$ e $v_{\max}$ il minimo e il massimo osservati, ogni valore diventa

$$v_{\text{nuovo}} = 255 \cdot (v_{\text{vecchio}} - v_{\min}) / (v_{\max} - v_{\min})$$

Questo sposta i valori dei bin dell'istogramma verso bin prima inutilizzati. Poiché i livelli sono interi e il numero di livelli usati non aumenta, l'istogramma risultante ha un aspetto «a pettine»: i bin mancanti sono i livelli che nessun pixel può assumere. Lo stretching non crea nuovi dettagli, rende solo più evidenti quelli esistenti.

Prima (valori da 85 a 170)



Dopo lo stretching (da 0 a 255)

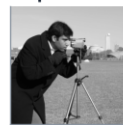
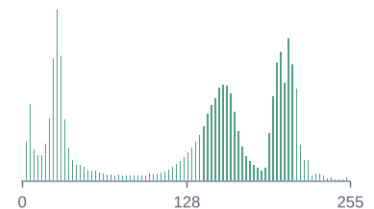


Figura 3. Una immagine con valori compresi tra 85 e 170 prima e dopo l'espansione del contrasto. L'istogramma a destra ha lo stesso numero di barre ma distanziate: aspetto a pettine.

PROBLEMI DI RANGE E NORMALIZZAZIONE

Operando aritmeticamente sui pixel può accadere che un valore sia negativo, maggiore del massimo (255) o non intero. I primi due sono problemi di **range**. Le soluzioni più comuni sono due: portare a 0 i valori negativi e a 255 quelli maggiori di 255 (**clipping**), oppure **rinormalizzare** il range con la stessa formula dello stretching. Il terzo problema si risolve arrotondando o troncando.

Equalizzazione dell'istogramma

Si parla di immagine **equalizzata** quando il contributo di ogni tonalità di grigio è pressappoco uguale: l'istogramma è «uniforme», o appiattito. Non vuol dire che i pixel abbiano tutti lo stesso valore, ma il contrario: i valori sono distribuiti il più uniformemente possibile su tutta la scala. Se r_k è un livello di grigio e n_k il numero di pixel di quel livello, con $p(r_k) = n_k / (M \cdot N)$ l'**algoritmo di equalizzazione** assegna ai nuovi livelli i valori

$$s_k = \text{arrotondamento di } (L - 1) \cdot \sum_{j=0 \dots k} p(r_j)$$

cioè $L - 1$ moltiplicato per la **distribuzione cumulativa** (CDF) dell'istogramma originale: i livelli frequenti vengono «allargati», quelli rari «compressi».

Esempio. Immagine a 3 bit ($L = 8$) di 64×64 pixel ($M \cdot N = 4096$) con la distribuzione seguente.

r_k	n_k	$p(r_k)$	CDF	$7 \cdot \text{CDF}$	s_k (arrotondato)
0	790	0,19	0,19	1,35	1
1	1023	0,25	0,44	3,10	3
2	850	0,21	0,65	4,55	5
3	656	0,16	0,81	5,67	6
4	329	0,08	0,89	6,23	6
5	245	0,06	0,95	6,65	7
6	122	0,03	0,98	6,86	7
7	81	0,02	1,00	7,00	7

Tabella 2. Calcolo dell'equalizzazione per l'immagine a 3 bit dell'esempio.

I nuovi livelli sono 1, 3, 5, 6, 6, 7, 7, 7: i vecchi livelli 3 e 4 confluiscono nel livello 6 ($656 + 329 = 985$ pixel) e i vecchi livelli 5, 6 e 7 nel livello 7 ($245 + 122 + 81 = 448$ pixel). L'istogramma non diventa perfettamente piatto, perché i valori sono discreti e i pixel di uno stesso livello non si possono dividere.

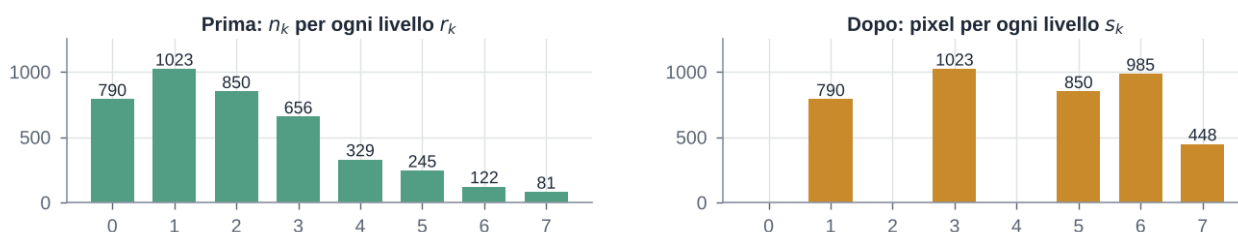


Figura 4. L'istogramma dell'esempio prima e dopo l'equalizzazione.

Su una fotografia scura l'equalizzazione rende visibili dettagli nelle ombre. In ambra la CDF: dopo l'equalizzazione è circa una retta.

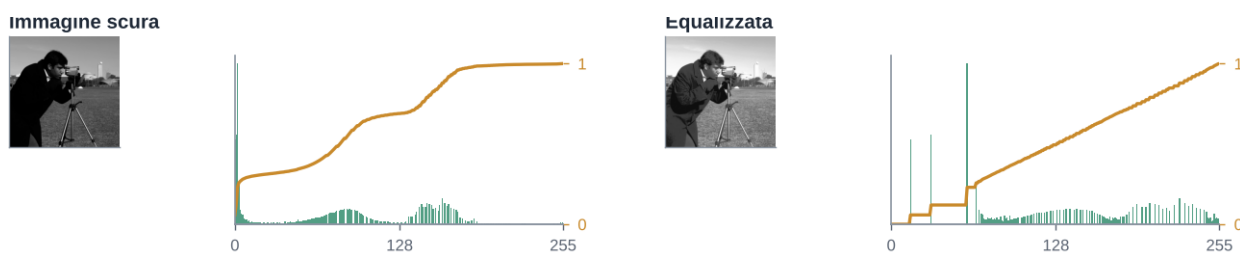


Figura 5. Equalizzazione su una fotografia scura, con istogramma (barre) e CDF (linea ambra).

ATTENZIONE

L'equalizzazione non sempre migliora l'immagine: amplifica anche il rumore e può rendere innaturale l'aspetto. È un'operazione puntuale, cioè dipende solo dal valore del pixel (capitolo 20), definita da una funzione che si può memorizzare in una LUT.

DA RICORDARE

L'istogramma conta i pixel per ogni livello di grigio e non conserva informazione spaziale. Lo stretching estende linearmente i valori su 0 a 255; l'equalizzazione usa la CDF per rendere l'istogramma il più uniforme possibile.

Le operazioni puntuali e le LUT

Operatori che trasformano ogni pixel in base al solo suo valore, la tabella di ricerca (LUT) che li descrive e le trasformazioni più usate: negativo, logaritmo, potenza, binarizzazione, contrasto, solarizzazione.

Operazioni nel dominio spaziale

Le elaborazioni di una immagine nel **dominio spaziale**, cioè direttamente sui pixel, si possono scrivere come

$$g(x, y) = T[f(x, y)]$$

dove f è l'immagine di ingresso, g quella di uscita e T è un **operatore** su f definito in un **intorno** del punto (x, y) . La dimensione dell'intorno definisce il carattere dell'elaborazione:

TIPO	INTORNO DI (x, y)	ESEMPIO
Puntuale	il pixel stesso	negativo, soglia, equalizzazione
Locale	una piccola regione, per esempio quadrata, centrata nel pixel	media, mediana, filtri di bordo (capitoli da 22 a 24)
Globale	l'intera immagine f	trasformata di Fourier (capitolo 26)

Tabella 1. I tre tipi di operazioni.

Per semplicità si lavora su immagini a toni di grigio. Le stesse operazioni si estendono alle immagini RGB operando separatamente sui tre canali (piani) R, G e B, ciascuno trattato come una immagine in grigi indipendente dagli altri. Questa soluzione è semplice ma non sempre adatta: trasformare i canali in modo indipendente può alterare le tinte.

Gli operatori puntuali

OPERATORE PUNTUALE

Un operatore che, preso in ingresso il valore di un pixel, ne restituisce uno cambiato che dipende **esclusivamente dal valore del pixel in ingresso**. Non dipende dalla posizione del pixel né dai suoi vicini: se il valore 0 diventa $T(0)$ in un punto, diventa $T(0)$ in ogni altro punto dell'immagine.

Le tipiche operazioni puntuali sono:

- aggiunta o sottrazione di una costante a tutti i pixel (per compensare sotto o sovraesposizioni);
- inversione della scala dei grigi (negativo);
- espansione del contrasto (capitolo 19);
- modifica dell'istogramma: equalizzazione o specifica dell'istogramma (capitolo 19);
- binarizzazione, trasformazioni logaritmica e di potenza;
- presentazione in falsi colori.

Un operatore puntuale può essere rappresentato da una funzione che, preso in ingresso $f(x, y)$, lo trasforma in $g(x, y) = T(f(x, y))$, con f e g appartenenti allo stesso campo di definizione (per esempio entrambi tra 0 e 255).

La LUT, tabella di ricerca

Poiché un operatore puntuale dipende solo dal valore del pixel, ed è quindi una funzione di una sola variabile, esso è completamente descritto da una tabella con una riga di ingresso e una di uscita:

IN	0	1	2	3	4	5	6	7	...	255
OUT	$T(0)$	$T(1)$	$T(2)$	$T(3)$	$T(4)$	$T(5)$	$T(6)$	$T(7)$	...	$T(255)$

Tabella 2. La LUT di un operatore puntuale per una immagine a 8 bit.

LUT (LOOK UP TABLE)

La **tabella di ricerca** di un operatore memorizza in anticipo il risultato dell'operazione per **ogni possibile valore di ingresso**. L'operazione si applica poi a un pixel semplicemente **leggendo** la posizione della tabella indicata dal suo valore: $g = \text{LUT}[f]$.

Il grafico di una LUT, con il valore di ingresso sull'asse orizzontale e il valore di uscita su quello verticale, è l'interfaccia che offrono tutti i programmi commerciali di elaborazione di immagini per gestire le operazioni puntuali (in Photoshop è lo strumento «Curve»).

A COSA SERVE

Lo scopo della LUT è **ottimizzare la velocità di elaborazione**. Se l'operazione è costosa (un logaritmo, una radice quadrata, una elaborazione complessa del colore) ed è applicata a moltissimi dati, conviene calcolarla una volta sola per ciascuno dei valori possibili e poi sostituire il calcolo con una ricerca in memoria. Per esempio, una immagine in grigi di 1000×1000 pixel richiederebbe 1 000 000 di logaritmi calcolati direttamente, ma solo 256 con la LUT (più 1 000 000 di semplici letture di tabella).

QUANDO SI PUÒ USARE

La LUT **non si può definire per tutti gli operatori** in senso assoluto. L'applicabilità dipende da due fattori:

1. L'operatore deve agire su un **dominio finito e discreto** di valori di ingresso, e il risultato deve dipendere solo da quel valore (dev'essere appunto puntuale). Un operatore che dipende dai pixel vicini, come la media o la mediana, non ha una LUT, perché il risultato non è una funzione del solo valore del pixel.
2. Anche con un dominio finito, la LUT può diventare **impraticabile per le dimensioni** se l'ingresso usa troppi bit.

INGRESSO	VALORI POSSIBILI	VOCI DELLA LUT	DIMENSIONE DELLA TABELLA
Grigi, 8 bit	256	256	256 byte (1 byte per voce)
Grigi, 16 bit	65 536	65 536	circa 64 KiB (1 byte per voce)
Colore RGB, 24 bit	16 777 216	16 777 216	circa 48 MiB (3 byte per voce)

Tabella 3. Dimensione della LUT al variare dei bit in ingresso.

Per questo, sulle immagini a colori la LUT si applica in genere **per canale**: tre tabelle da 256 voci, una per R, una per G e una per B, invece di una sola tabella da 16 milioni di voci.

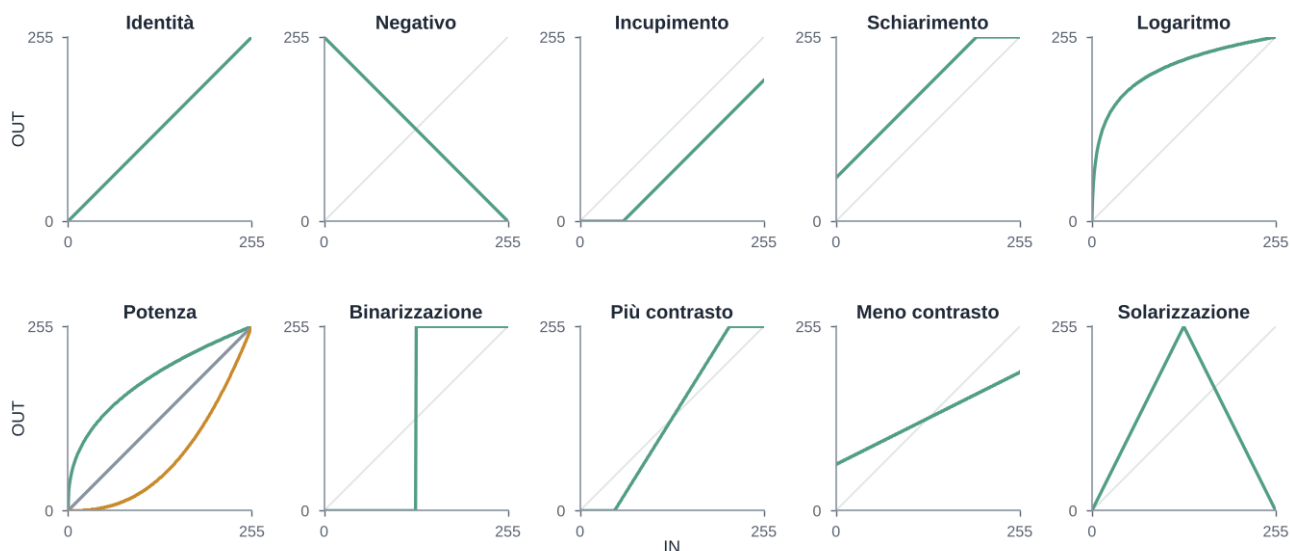


Figura 1. Le LUT delle principali operazioni puntuali (la linea chiara è l'identità, $g = f$; nel pannello della potenza: $\gamma = 0,4$ in verde, 1 in grigio, 2,5 in ambra). Le curve sono costruite nei paragrafi seguenti.

Negativo, incupimento e schiarimento

Il **negativo** è la più semplice operazione puntuale: associa al valore $f(x, y)$ del pixel il valore

$$g(x, y) = 255 - f(x, y)$$

La curva è una retta con pendenza -1 : i neri diventano bianchi e viceversa, il grigio 128 diventa 127.

Per **scurire** l'immagine si abbassa la curva rispetto all'identità, per **schiarirla** la si alza. La forma più semplice è l'addizione di una costante c , positiva o negativa: $g = f + c$. I valori che escono dall'intervallo da 0 a 255 sono portati al limite più vicino (**clipping**): con $c = +60$ tutti i pixel da 195 in su diventano 255 e l'informazione sulle luci va persa; con $c = -60$ si perdono le ombre.



Figura 2. Negativo, incupimento e schiarimento di una fotografia: i valori fuori scala vengono portati a 0 o a 255.

ATTENZIONE

Con il clipping l'operazione non è reversibile: schiarire di 60 e poi scurire di 60 non restituisce l'originale, perché i pixel saturati a 255 restano a 195. Il negativo invece lo è: $255 - (255 - f) = f$.

La trasformazione logaritmica

La trasformazione logaritmica **comprime la gamma dinamica**, permettendo di memorizzare o visualizzare con la scala dei grigi usuale immagini con escursioni di intensità molto ampie. Si esprime con

$$g(x, y) = c \cdot \log(1 + f(x, y))$$

dove c è una costante positiva che normalizza il risultato tra 0 e 255. Il termine 1 serve perché $\log(0)$ non esiste: con $1 + f$ si ha $\log 1 = 0$ per $f = 0$. Per f da 0 a 255 il valore massimo si ottiene per $f = 255$, quindi $c = 255 / \log(256) \approx 45,98$ se il logaritmo è quello naturale (la scelta della base non cambia il risultato, perché è assorbita da c).

Esempio. Per $f = 63$ si ha $g = 45,98 \cdot \ln(64) \approx 191,2$, cioè 191; per $f = 7$ si ha $g = 45,98 \cdot \ln(8) \approx 95,6$, cioè 96. I toni scuri vengono quindi «aperti» (i valori da 0 a 7 occupano l'intervallo da 0 a 96), mentre i toni chiari vengono compressi: i valori da 240 a 255 occupano solo l'intervallo da 252 a 255.

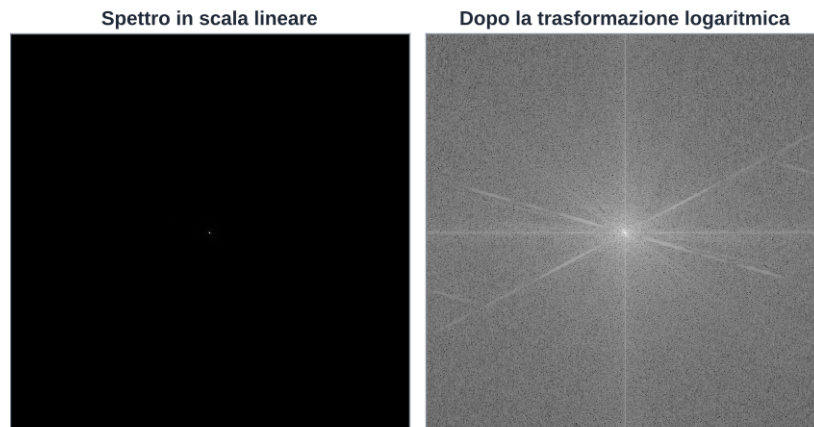


Figura 3. Modulo della trasformata di Fourier della fotografia (capitolo 26): il valore massimo è migliaia di volte maggiore del valore tipico, e in scala lineare si vede solo un punto. Il logaritmo comprime la gamma e rivela la struttura.

Il caso classico è lo **spettro di Fourier** (capitolo 26): il suo modulo ha un valore massimo di circa $3,4 \cdot 10^7$ nella fotografia mostrata, mentre il valore mediano è circa 3700. In scala lineare, a 256 livelli, quasi tutto lo spettro sarebbe nero. Il logaritmo, con $c = 14,71$, rende visibile la struttura.

La trasformazione di potenza

La trasformazione di potenza (o **correzione gamma**) si esprime con

$$g(x, y) = c \cdot (f(x, y))^{\gamma}$$

dove c e γ sono costanti positive, scelte in modo da normalizzare il risultato nell'intervallo da 0 a 255. Se i valori sono prima portati tra 0 e 1, la forma più comoda è $g = 255 \cdot (f / 255)^{\gamma}$, che lascia fissi 0 e 255.

- per γ minore di 1 la trasformazione ha effetti analoghi a quelli del logaritmo: **espansione** della dinamica per i bassi valori di f e **compressione** per gli alti. L'immagine si schiarisce;
- per γ maggiore di 1 si hanno esattamente gli effetti opposti: si espandono i toni chiari e si comprimono gli scuri. L'immagine si scurisce;
- per $\gamma = 1$ si ha l'identità.

Esempio. Per $f = 64$ si ha $255 \cdot (64 / 255)^{0,4} \approx 147$ con $\gamma = 0,4$ e $255 \cdot (64 / 255)^{2,5} \approx 8$ con $\gamma = 2,5$.



Figura 4. Trasformazione di potenza con γ minore e maggiore di 1.

GAMMA E MONITOR

La risposta di un monitor CRT è non lineare: la luminanza emessa è proporzionale al segnale elevato a circa $\gamma = 2,5$. Per compensarla si **pre elabora** l'ingresso con la funzione inversa: $g = f^{1/2,5} = f^{0,4}$. In questo modo l'effetto complessivo dei due passaggi è circa lineare. Il valore 0,4 è proprio quello del secondo pannello di Figura 4.

La binarizzazione

La **binarizzazione** produce una immagine con soli due livelli, nero e bianco. Si sceglie una soglia T e si mettono a nero tutti i pixel il cui valore è minore di T e a bianco tutti gli altri. La LUT è una funzione a gradino: vale 0 fino a $T - 1$ e 255 da T in poi.

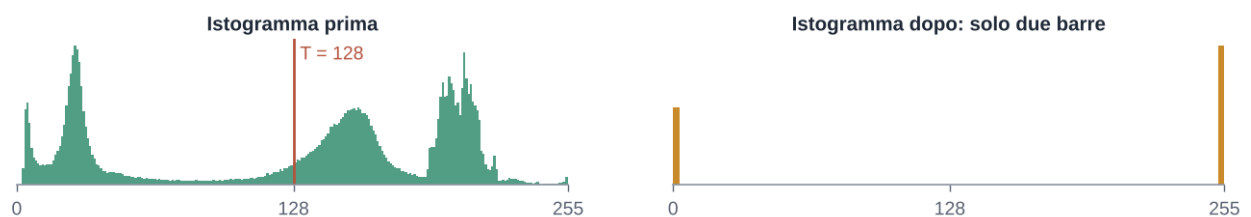


Figura 5. L'istogramma della fotografia prima e dopo la binarizzazione con soglia 128: tutti i pixel finiscono nelle due uniche barre, a 0 e a 255.

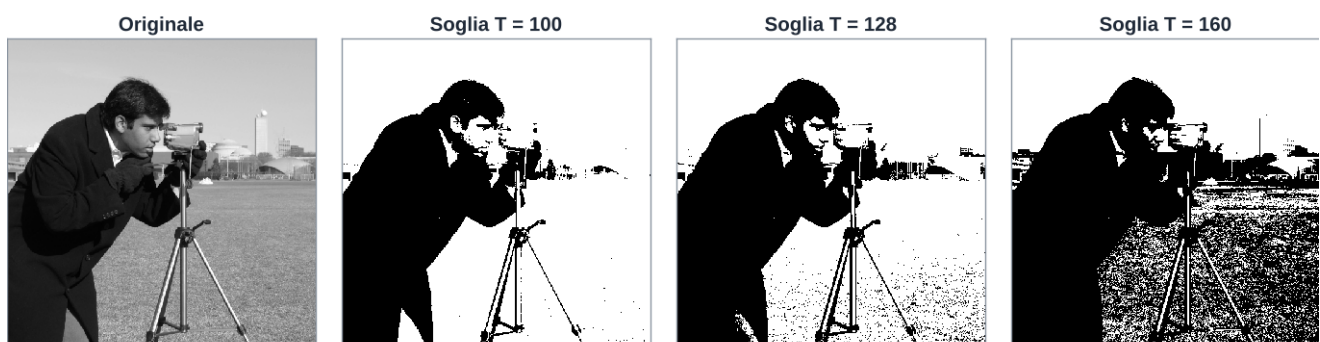


Figura 6. Binarizzazione con tre soglie diverse. La scelta della soglia cambia molto il risultato.

Le variazioni di contrasto

Aumentare il **contrasto** significa rendere più evidenti le differenze di tono. Ciò si ottiene sostituendo il valore di un pixel con uno più scuro, se il pixel è scuro, o più chiaro, se il pixel è chiaro. La curva diventa quindi **più ripida** nella zona centrale. Una forma semplice è la retta centrata sul grigio medio,

$$g = 128 + a \cdot (f - 128)$$

con a maggiore di 1 per aumentare il contrasto (nell'esempio $a = 1,6$) e minore di 1 per diminuirlo (nell'esempio $a = 0,5$), con clipping a 0 e 255. Diminuire il contrasto significa avvicinare tra loro i valori: la curva diventa più piatta e l'istogramma più stretto.

Le curve non monotone: la solarizzazione

Una LUT è **monotona** se non decresce mai. Le trasformazioni viste finora sono monotone: la curva o non scende mai (come per la potenza) o non sale mai (come per il negativo), quindi non fa mai «scavalcare» due toni in modo incoerente. Si possono però progettare curve **non monotone**. Un esempio è la **solarizzazione**, in cui i toni scuri sono schiariti fino a metà scala e quelli chiari sono invertiti:

$$g = 2 \cdot f \text{ se } f \text{ è minore di } 128, \quad g = 2 \cdot (255 - f) \text{ altrimenti}$$

È l'analogo digitale dell'effetto fotografico di Sabattier, in cui una pellicola in sviluppo viene esposta di nuovo alla luce: i toni chiari si invertono e l'immagine ha l'aspetto di un negativo parziale.



Figura 7. Variazioni di contrasto e solarizzazione. Nel primo caso i grigi scuri diventano più scuri e i chiari più chiari; nel secondo i valori si avvicinano al grigio medio.

Falsi colori

La presentazione in **falsi colori** è anch'essa una operazione puntuale: la LUT ha tre colonne (R, G e B) e associa a ciascuno dei 256 livelli un colore. Serve a rendere visibili differenze di intensità sottili, che l'occhio distingue meglio come differenze di colore che come differenze di grigio, e si usa per esempio nelle immagini termiche, mediche o satellitari.



Figura 8. Presentazione in falsi colori: a ogni livello di grigio si associa, attraverso una LUT con tre colonne (R, G, B), un colore. I pixel non cambiano, cambia solo come li si vede.

DA RICORDARE

Un operatore puntuale dipende solo dal valore del pixel ed è descritto da una LUT, una tabella di 256 voci calcolata una sola volta. Con la LUT un logaritmo o una potenza costano una lettura di memoria per pixel. Negativo, soglia, logaritmo, potenza, contrasto, solarizzazione ed equalizzazione sono tutte operazioni puntuali, e differiscono solo per la forma della curva.

I bit planes

La scomposizione di una immagine in piani binari, da quello più significativo al meno significativo, il problema del binario puro e il codice Gray.

Che cos'è un piano di bit

Una immagine con una profondità di colore di N bit può essere rappresentata da N **piani di bit** (*bit planes*), ciascuno dei quali si può vedere come una singola immagine binaria, con pixel che valgono 0 (nero) o 1 (bianco). I piani sono ordinati dal bit più significativo, **MSB** (*most significant bit*), al bit meno significativo, **LSB** (*least significant bit*).

PIANO DI BIT

Il *bit plane* di una immagine digitale a N bit è un insieme di N immagini binarie (piani), in cui l'immagine k contiene i valori del bit k della codifica scelta, per $k = 0$ (LSB), ..., $N - 1$ (MSB).

Con la codifica in **binario puro** un pixel di valore f si scrive $f = b_7 \cdot 2^7 + b_6 \cdot 2^6 + \dots + b_0 \cdot 2^0$, e il piano k è l'immagine dei bit $b_k = \lfloor f / 2^k \rfloor \bmod 2$. Il piano 7 pesa 128 e il piano 0 pesa 1, per cui il valore del pixel è la somma pesata dei suoi piani:

$$f = \sum_{k=0 \dots 7} b_k \cdot 2^k$$

Esempio. Si vogliono scrivere i due piani meno significativi (LSB) e i due più significativi (MSB) per la sequenza di toni di grigio 120, 240, 0, 255. Prima si scrivono i numeri in base 2 con 8 cifre, perché i toni sono su 8 bit: 120 = 01111000 (si aggiunge uno 0 iniziale a 1111000), 240 = 11110000, 0 = 00000000, 255 = 11111111. Poi, per ciascun pixel, si leggono i bit a destra (LSB) e a sinistra (MSB). I piani sono le «colonne» che si ottengono scrivendo i numeri uno sotto l'altro:

Valore	Binario	Piano 7	Piano 6	Piano 1	Piano 0
120	01111000	0	1	0	0
240	11110000	1	1	0	0
0	00000000	0	0	0	0
255	11111111	1	1	1	1

Tabella 1. I due piani MSB (7 e 6) e i due LSB (1 e 0) per i quattro pixel dell'esempio.

Leggendo le colonne, nell'ordine dei pixel 120, 240, 0, 255, si ottiene: piano 7 = 0 1 0 1; piano 6 = 1 1 0 1; piano 1 = 0 0 0 1; piano 0 = 0 0 0 1.

Osservazioni sul binario puro

Sulla fotografia seguente si vedono tutti gli otto piani, dal piano 7 al piano 0.

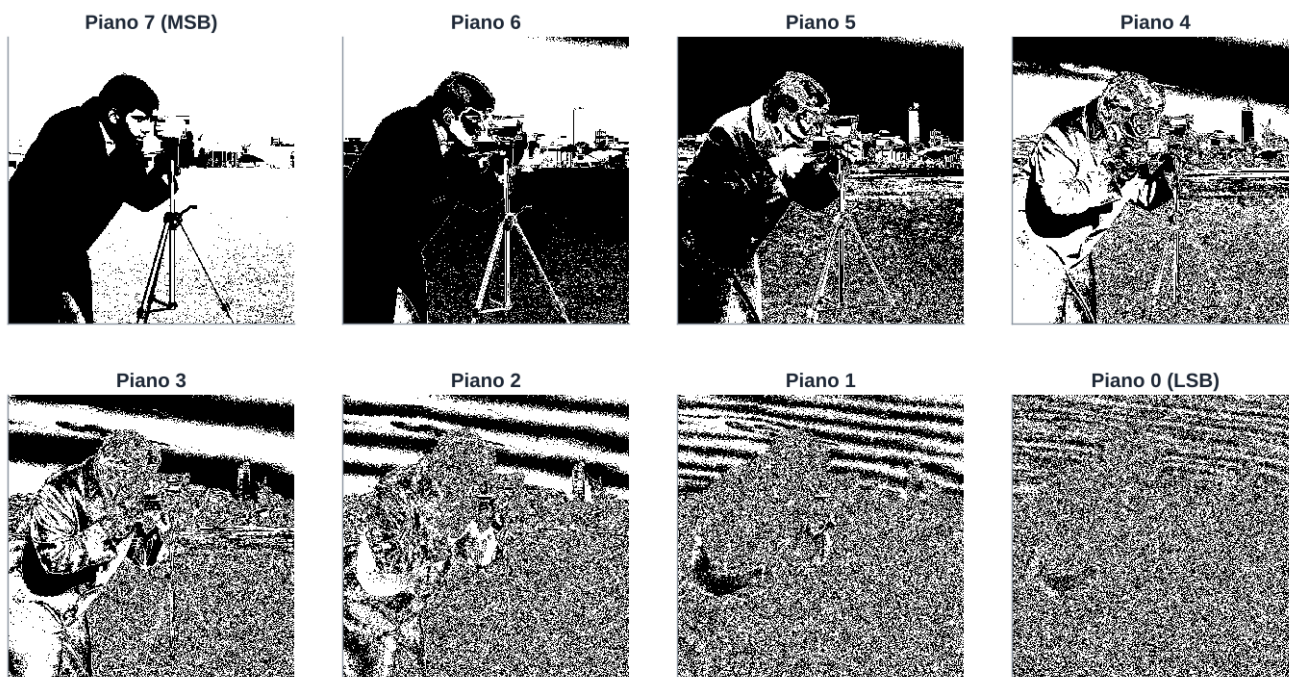


Figura 1. I piani di bit da 7 a 0 di una immagine a 8 bit in scala di grigi. Ogni piano è una immagine binaria.

- i piani **più significativi** contengono informazioni sulla struttura dell'immagine (contorni, grandi masse chiare e scure), mentre quelli via via meno significativi forniscono dettagli sempre più piccoli;
- solo i piani dal 7 al 3 contengono dati significativi dal punto di vista visuale;
- il **rumore** dell'immagine e gli errori di acquisizione sono più evidenti nei piani bassi, e il piano 0 sembra un puro rumore.



Figura 2. Ricostruzione con i soli piani più significativi (gli altri piani messi a zero). Con i piani 7 e 6 l'immagine assume solo i valori 0, 64, 128 e 192.

Con un solo piano, il piano 7, si distingue già il soggetto (PSNR 13,2 dB); con quattro piani, cioè 4 bit per pixel, la qualità è già discreta (29,2 dB), con qualche falso contorno nelle zone sfumate. Questo mostra che nelle immagini naturali l'informazione è concentrata nei piani alti.

ELIMINARE UN INTERVALLO DI TONI

La scomposizione in piani è utile per modificare i valori di un certo intervallo. Per esempio, per eliminare i grigi compresi tra 32 e 63 si pone a 0 il bit 5, cioè tutto il piano 5. Attenzione però a che cosa accade: il bit 5 vale 1 non solo nell'intervallo da 32 a 63, ma anche in quelli da 96 a 127, da 160 a 191 e da 224 a 255. Azzerarlo sottrae 32 a tutti questi valori: da 32 a 63 diventano da 0 a 31, ma anche da 96 a 127 diventano da 64 a 95, e così via. Le osservazioni di questo paragrafo valgono solo per il binario puro.



Figura 3. Ricostruzione dopo aver posto a zero un solo piano di bit. Senza il piano 0 la differenza è invisibile, senza il piano 7 si perde la struttura dell'immagine.

Il problema del binario puro e il codice Gray

Con il binario puro una piccola variazione di intensità può ripercuotersi su **tutti i piani**. Il caso estremo: un pixel di intensità 127 (01111111) con un vicino di intensità 128 (10000000) differisce in tutti gli otto bit, benché il tono sia quasi lo stesso. Ne segue che nei piani le transizioni bianco nero sono molte e disordinate. Serve un codice in cui valori vicini abbiano codifiche binarie molto simili.

CODICE GRAY

Il codice Gray a m bit $g_{m-1} \dots g_1 g_0$ che corrisponde al numero in binario puro $a_{m-1} \dots a_1 a_0$ si calcola con $g_{m-1} = a_{m-1}$ e $g_k = a_k \oplus a_{k+1}$ per gli altri bit, dove $\oplus$ è lo **XOR** (OR esclusivo). Due codici consecutivi differiscono per **un solo bit** (distanza di Hamming 1).

In altre parole, ogni bit del codice Gray vale 1 se il bit corrispondente e quello subito più a sinistra del binario puro sono **diversi**. Su un intero si scrive anche $g = a \text{ XOR } (a \gg 1)$, cioè lo XOR del numero con sé stesso traslato di un bit a destra. La conversione inversa, dal Gray al binario, è $a_{m-1} = g_{m-1}$ e $a_k = a_{k+1} \oplus g_k$, partendo dal bit più alto.

Numero decimale	Binario puro	Codice Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Tabella 2. Il codice Gray a 3 bit. Il passaggio più brusco in binario, da 3 (011) a 4 (100), cambia tutti e tre i bit; in Gray, da 010 a 110, ne cambia uno solo.

Tornando all'esempio di prima: in binario puro $127 = 01111111$ e $128 = 10000000$, che differiscono in tutti i bit; in codice Gray $127 = 01000000$ e $128 = 11000000$, che differiscono in **un solo bit** (il bit 7).

UN ESERCIZIO COMPLETO

Si vuole il codice Gray dei quattro pixel dell'esempio precedente. Si applica $g = a \text{ XOR } (a \gg 1)$ sui numeri a 8 bit:

Valore	Binario puro	Gray	Piano 7	Piano 6	Piano 1	Piano 0
120	01111000	01000100	0	1	0	0
240	11110000	10001000	1	0	0	0
0	00000000	00000000	0	0	0	0
255	11111111	10000000	1	0	0	0

Tabella 3. Il codice Gray dei quattro pixel e i suoi piani MSB (7 e 6) e LSB (1 e 0).

I piani del codice Gray, letti nell'ordine dei pixel, sono: piano 7 = 0 1 0 1; piano 6 = 1 0 0 0; piano 1 = 0 0 0 0; piano 0 = 0 0 0 0. Per i piani alti il risultato si ottiene anche lavorando sulle sole due cifre estratte, perché a sinistra del bit 7 non c'è nulla (il bit «virtuale» è 0). Per i piani bassi invece non basta: per calcolare il bit 1 del Gray servono i bit 1 e 2 del binario puro, e il bit 0 del Gray richiede i bit 0 e 1.

ATTENZIONE

Un errore frequente è calcolare il Gray dei piani 1 e 0 usando solo i due bit estratti, tenendo il primo e facendo lo XOR per il secondo. Per il pixel 255 (11111111) darebbe 10, ma il vero Gray è 10000000, e i suoi due bit meno significativi sono 00. I piani bassi del Gray di 255 sono tutti zero, perché in 255 tutti i bit sono uguali tra loro e ogni XOR dà 0, tranne il bit 7, che resta 1.

Binario puro e Gray a confronto

Poiché in Gray un aumento di 1 dell'intensità cambia un solo bit, cioè un solo piano, i piani di una immagine in codice Gray risultano più **coerenti** tra loro di quelli in binario puro: contengono regioni più uniformi e **meno transizioni** bianco nero (complessità descrittiva minore).

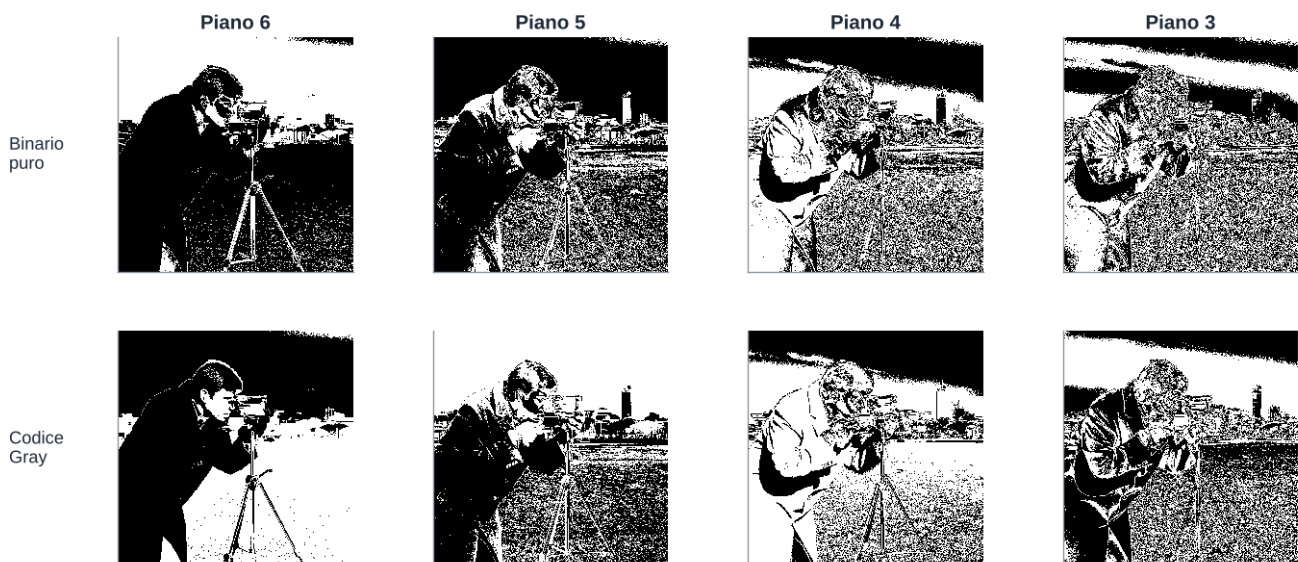


Figura 4. I piani di bit 6, 5, 4, 3 in binario puro (sopra) e in codice Gray (sotto). Il piano 7 è identico nei due casi. Nei piani Gray le zone sono più compatte e le transizioni meno numerose.

Piano	Transizioni, binario puro	Transizioni, codice Gray
Piano 7	20 218	20 218
Piano 6	31 849	14 849
Piano 5	89 191	72 798
Piano 4	118 645	81 582
Piano 3	147 346	118 839
Piano 2	182 125	150 009
Piano 1	217 285	181 404
Piano 0	249 199	218 724

Tabella 4. Numero di coppie di pixel adiacenti (orizzontali o verticali) di valore diverso in ciascun piano della fotografia. Il piano 7 è uguale, per gli altri il Gray ne ha sempre meno.

Le caratteristiche viste indicano una **minore entropia** (maggiore ridondanza) per la codifica Gray: i piani sono più regolari, quindi più facili da comprimere (capitolo 27). Per questo il codice Gray si usa quando si vogliono comprimere i piani di bit separatamente.

ATTENZIONE AL SIGNIFICATO DEI BIT

Il significato associato ai bit è diverso nelle due codifiche, quindi alcune proprietà dell'una **non valgono** per l'altra:

- se si azzerano dei piani in codice Gray, si eliminano intervalli di valori diversi, e meno ordinati, rispetto a quelli del binario puro;
- anche in Gray dettagli e rumore tendono a concentrarsi nei piani più bassi, ma eliminare direttamente quei piani può introdurre artefatti indesiderati.



Figura 5. Tenendo solo i tre piani più alti con le due codifiche si eliminano intervalli di valori diversi, e l'immagine risultante è diversa.

Tenendo solo i piani 7, 6, 5 si ottiene un PSNR di 22,9 dB in binario puro e di 22,2 dB in Gray (dopo aver riconvertito il risultato in binario puro per visualizzarlo): i due risultati sono diversi, perché gli stessi piani codificano informazioni diverse.

DA RICORDARE

Un piano di bit è l'immagine binaria di un bit di tutti i pixel. Il piano 7 pesa 128 e porta la struttura, il piano 0 pesa 1 e porta il rumore. Il binario puro cambia molti bit tra toni vicini, il codice Gray ($g = a \text{ XOR } (a \gg 1)$) ne cambia uno solo e produce piani più regolari. Nel Gray ogni bit dipende da due bit del binario puro, quindi non si possono calcolare i piani bassi guardando solo i due bit estratti.

Operatori lineari e invarianti per traslazione

Gli operatori sulle immagini visti come trasformazioni di vettori, la linearità, l'invarianza per traslazione, la risposta all'impulso e il kernel, e come si applica un filtro a una immagine.

Operatori sulle immagini

Un **operatore** è simile a una funzione, ma invece di prendere un numero x e restituire un numero y prende un **vettore** v e lo trasforma in un altro vettore w : $T(v) = w$. Una immagine $M \times N$ si può vedere come un vettore di $M \cdot N$ componenti, i valori dei suoi pixel; un operatore sulle immagini trasforma quindi una immagine in una immagine. Un operatore $F: V \rightarrow W$ trasforma gli elementi di uno spazio V in elementi di uno spazio W .

LA BASE CANONICA

Un vettore di lunghezza N è un elemento di uno spazio a N dimensioni e si può scomporre usando la **base canonica** di tale spazio, formata dai vettori che hanno un 1 in una sola posizione e 0 altrove. Per esempio, per $N = 3$:

$$(234, 204, 34) = 234 \cdot (1, 0, 0) + 204 \cdot (0, 1, 0) + 34 \cdot (0, 0, 1)$$

Lo stesso ragionamento vale per le immagini: gli elementi della base canonica sono le **immagini impulsive**, con un solo pixel pari a 1 e tutti gli altri nulli. Una immagine $M \times N$ è la somma di $M \cdot N$ impulsi, ciascuno moltiplicato per il valore del pixel che occupa quella posizione.

$$\begin{array}{|c|c|} \hline \text{Immagine} \\ \hline 234 & 204 \\ \hline 16 & 44 \\ \hline \end{array} = 234 \cdot \begin{array}{|c|c|} \hline \text{Impulso 11} \\ \hline 1 & 0 \\ \hline 0 & 0 \\ \hline \end{array} + 204 \cdot \begin{array}{|c|c|} \hline \text{Impulso 12} \\ \hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array} + 16 \cdot \begin{array}{|c|c|} \hline \text{Impulso 21} \\ \hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array} + 44 \cdot \begin{array}{|c|c|} \hline \text{Impulso 22} \\ \hline 0 & 0 \\ \hline 0 & 1 \\ \hline \end{array}$$

Figura 1. Una immagine 2×2 come somma pesata di quattro immagini impulsive: ogni impulso vale 1 in un solo pixel e 0 altrove. I pesi sono i valori dei pixel.

Operatori lineari

OPERATORE LINEARE

Un operatore $F: V \rightarrow W$ è **lineare** se per ogni coppia di vettori v_1 e v_2 di V e per ogni coppia di scalari reali a e b si ha: $F(a \cdot v_1 + b \cdot v_2) = a \cdot F(v_1) + b \cdot F(v_2)$.

La proprietà riunisce due requisiti più semplici: l'**omogeneità**, $F(a \cdot v) = a \cdot F(v)$ (moltiplicare l'ingresso per a moltiplica l'uscita per a), e l'**additività**, $F(v_1 + v_2) = F(v_1) + F(v_2)$ (la risposta alla somma è la somma delle risposte). Se anche uno solo dei due requisiti, per qualche valore, non vale, l'operatore **non è lineare**: basta un controesempio. Un test rapido: un operatore lineare manda sempre lo zero in zero, perché $F(0) = F(0 \cdot v) = 0 \cdot F(v) = 0$.

Conseguenza fondamentale. Se si conosce una base di V e il comportamento di F su ogni elemento di tale base, si può calcolare il comportamento di F su **ogni** elemento di V , perché ogni vettore è combinazione degli elementi della base. Per le immagini: per descrivere un operatore lineare basta conoscere la sua risposta a tutte le immagini impulsive.

Esempio 1. La funzione $F(x, y) = (x / 2, y / 3)$ è lineare. Il primo membro della proprietà è $a \cdot (x_1 / 2, y_1 / 3) + b \cdot (x_2 / 2, y_2 / 3) = ((a x_1 + b x_2) / 2, (a y_1 + b y_2) / 3)$, e il secondo membro è $F(a x_1 + b x_2, a y_1 + b y_2) = ((a x_1 + b x_2) / 2, (a y_1 + b y_2) / 3)$: i due membri sono uguali.

Esempio 2. La funzione $F(x, y) = (255 - x, 255 - y)$ **non** è lineare. Basta guardare i valori: $F(0, 0) = (255, 255)$, e non è il vettore nullo. Oppure, con $a = b = 1$, $x_1 = 100$, $x_2 = 50$, la somma delle uscite del primo componente vale $(255 - 100) + (255 - 50) = 360$, mentre l'uscita sulla somma vale $255 - 150 = 105$. Questa funzione è la forma del **negativo** (capitolo 20) su una immagine di due pixel: il negativo non è un operatore lineare, ma **affine**.

OPERATORE	LINEARE	MOTIVO
Moltiplicare per una costante ($g = c \cdot f$)	sì	rispetta omogeneità e additività
Media su un intorno 3×3	sì	somma pesata dei pixel
Aggiungere una costante ($g = f + c$)	no	se $c \neq 0$, $F(0) = c$, non zero
Negativo ($g = 255 - f$)	no	$F(0) = 255$
Logaritmo e potenza, binarizzazione	no	non sono somme pesate
Mediana su un intorno	no	controesempio sotto
Equalizzazione dell'istogramma	no	la funzione dipende dall'istogramma

Tabella 1. Alcuni operatori e la loro linearità.

La mediana non è lineare. Su un intorno di tre pixel si considerino $f_1 = (1, 0, 0)$ e $f_2 = (0, 0, 1)$: la mediana di f_1 è 0 e quella di f_2 è 0, quindi la somma delle risposte è 0. Ma $f_1 + f_2 = (1, 0, 1)$ ha mediana 1. Non vale l'additività.

Operatori invarianti per traslazione

OPERATORE INVARIANTE PER TRASLAZIONE (SHIFT INVARIANT)

Un operatore si dice **invariante per traslazione** quando il suo comportamento sulle immagini impulsive è sempre lo stesso, **indipendentemente dalla posizione** in cui si trova il pixel. Se si trasla l'ingresso, l'uscita viene traslata allo stesso modo.

La funzione $(x / 2, y / 3)$ dell'esempio 1 **non** è invariante per traslazione: se i due componenti sono due pixel adiacenti, il primo viene dimezzato e il secondo diviso per tre, cioè il comportamento sugli elementi della base canonica $((1, 0)$ e $(0, 1))$ dipende dalla posizione. Un altro esempio è la moltiplicazione di una immagine per una maschera di pesi variabili da pixel a pixel (per esempio un effetto «vignettatura»): è lineare, ma non è invariante per traslazione. Questi operatori non sono «cattivi», ma solo più **difficili** da studiare: bisogna conoscere il comportamento su ciascun impulso in ciascuna posizione.

Tutti gli operatori puntuali (capitolo 20) sono invarianti per traslazione, anche se non sono lineari. Il negativo, per esempio, prende il valore di grigio nel punto (x, y) e lo sottrae a 255: il comportamento è sempre lo stesso per tutti i livelli uguali, qualunque sia la posizione. Attenzione alla notazione: nella funzione $(x, y) \rightarrow (255 - x, 255 - y)$ le lettere sono i valori di due pixel, mentre nel negativo $g(x, y) = 255 - I(x, y)$ le stesse lettere indicano la **posizione** del pixel.

OPERATORE	LINEARE	INVARIANTE PER TRASLAZIONE
Negativo, soglia, logaritmo, potenza (puntuali)	no	sì
Media 3×3 , box, binomiale	sì	sì
Mediana, minimo, massimo	no	sì
Moltiplicazione per una maschera variabile	sì	no
Traslare l'immagine di un pixel	sì	sì

Tabella 2. Linearità e invarianza per traslazione di alcuni operatori. Le due proprietà sono indipendenti.

La risposta all'impulso

Riassumendo:

- se F è **lineare**, per descriverlo basta conoscere il comportamento su tutte le immagini impulsive;
- se F è **invariante per traslazione**, si comporta allo stesso modo su tutti gli impulsi, indipendentemente dalla loro posizione;
- se F è **sia lineare sia invariante per traslazione**, per descriverlo basta conoscere come si comporta su **un solo impulso**.

La **risposta all'impulso** (*impulse response*, o *point spread function*, PSF) è la carta d'identità di un tale operatore. La matrice che descrive la risposta all'impulso si chiama **kernel** o **maschera** dell'operatore; si chiama anche **maschera di convoluzione**, per ragioni che si vedranno nel capitolo 26. Un operatore su una immagine prende spesso il nome di **filtro**.

Impulso		Risposta all'impulso (kernel)
0	0	0
0	1	0,5
0	0	0

Figura 2. Un operatore lineare e invariante per traslazione è completamente definito dalla sua risposta all'impulso. Questo operatore calcola la media dei due vicini, a sinistra e a destra.

Ad un operatore lineare e invariante per traslazione corrisponde una maschera, e vale anche il viceversa: a ogni maschera corrisponde un operatore lineare e invariante per traslazione. I filtri di questo tipo si chiamano anche **filtri convolutivi**.

Dimensione del kernel. La dimensione del kernel può essere anche infinita, ma per ragioni pratiche si usano solo kernel finiti. La sua dimensione influenza la complessità del filtraggio, che dipende anche dal numero di pixel dell'immagine: con un kernel $s \times t$ servono $s \cdot t$ moltiplicazioni per pixel. Su una immagine di 1000×1000 pixel un kernel 3×3 richiede 9 000 000 di moltiplicazioni, uno 11×11 ne richiede 121 000 000.

Applicare un filtro: il filtraggio

Gli operatori che si usano per migliorare la qualità di una immagine o per estrarre informazioni da essa sono spesso **locali**: il valore di uscita di ogni pixel dipende da un limitato intorno del punto corrispondente nell'ingresso. Si possono pensare come **filtraggi** dell'immagine, ottenuti facendo la «convoluzione» tra l'immagine e una matrice. Applicare un filtro lineare e invariante per traslazione all'immagine f è equivalente a calcolare la convoluzione del kernel h con l'immagine.

Con le convenzioni del corso, se il kernel h ha dimensioni $s \times t$ e i suoi indici sono disposti in modo da avere il punto di coordinate (0, 0) nella posizione **centrale**, l'uscita è

$$g_{m,n} = \sum_{i=-s/2 \dots s/2} \sum_{j=-t/2 \dots t/2} h_{i,j} \cdot f_{m+i,n+j}$$

(con s e t dispari, gli indici vanno da $-(s-1)/2$ a $(s-1)/2$). In parole: si **sovrappone** il kernel all'immagine con il suo centro sul pixel (m, n) , si moltiplicano uno a uno gli elementi del kernel per i pixel sottostanti e si **sommano** i prodotti; il risultato è il valore del pixel (m, n) nell'immagine di uscita. Poi si ripete l'operazione per tutti i pixel.

ATTENZIONE

La formula vista è, a rigore, una **correlazione**. La convoluzione vera ruota il kernel di 180 gradi prima di sovrapporlo. I due risultati coincidono se il kernel è simmetrico rispetto al centro (media, binomiale, Laplaciano). Per i kernel antisimmetrici, come Sobel e Prewitt, la convoluzione vera cambia solo il segno del risultato. In generale bisogna conoscere la convenzione usata. Il corso usa la formula con somma di $h_{i,j} \cdot f_{m+i,n+j}$.

UN ESEMPIO COMPLETO

Si applica il kernel 3×3 della Figura 3 all'immagine 5×5 , calcolando prima solo le zone in cui il kernel sta tutto dentro l'immagine. Il primo valore si ottiene sovrapponendo il kernel alle prime tre righe e colonne:

$$2 \cdot 1 + 1 \cdot 2 + 2 \cdot 2 + 1 \cdot 3 + 2 \cdot 2 + 1 \cdot 2 + 2 \cdot 2 + 1 \cdot 5 + 2 \cdot 2 = 8 + 9 + 13 = 30$$

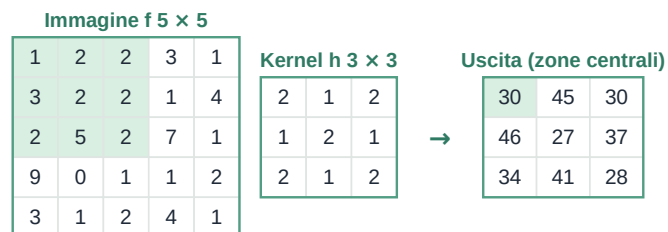


Figura 3. Filtraggio delle sole zone centrali: l'uscita ha dimensione 3×3 e il suo primo elemento, 30, è il risultato della finestra evidenziata. Gli indici di riga e colonna dell'immagine partono da 1.

Il problema dei bordi

Quando il centro del kernel si trova sul bordo dell'immagine, una parte del kernel cade fuori e mancano dei valori. Come si fa il filtraggio ai bordi? Le soluzioni possibili sono quattro.

1. **Filtrare solo le zone centrali.** L'uscita è più piccola dell'ingresso: con un kernel 3×3 si perdono una riga e una colonna per lato (Figura 3).
2. **Supportre che intorno all'immagine ci sia 0 (zero padding).** L'uscita ha le stesse dimensioni dell'ingresso, ma i valori ai bordi risultano più scuri.
3. **Assumere una topologia toroidale:** quando si «sfora a destra» si rientra a sinistra, quando si sfora in basso si rientra in alto, e viceversa.
4. **Replicare i valori più vicini:** si aggiunge in testa una riga uguale alla prima, in fondo una uguale all'ultima, e lo stesso con le colonne.

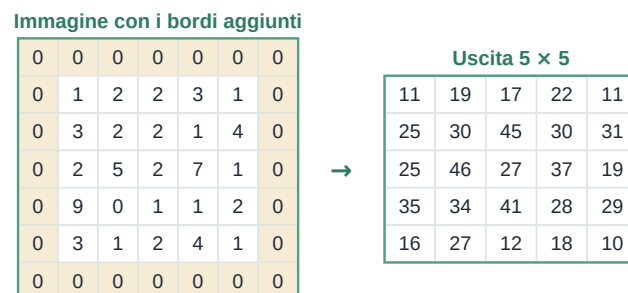


Figura 4. Immagine contornata da zeri e uscita. Le celle in ambra sono le righe e le colonne aggiunte.

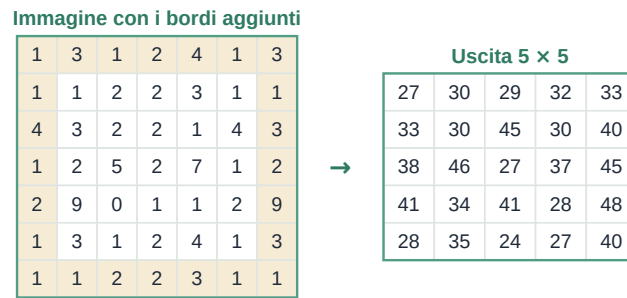


Figura 5. Bordi toroidali: la riga aggiunta in testa è l'ultima riga, e così via. Le celle in ambra sono le righe e le colonne aggiunte.

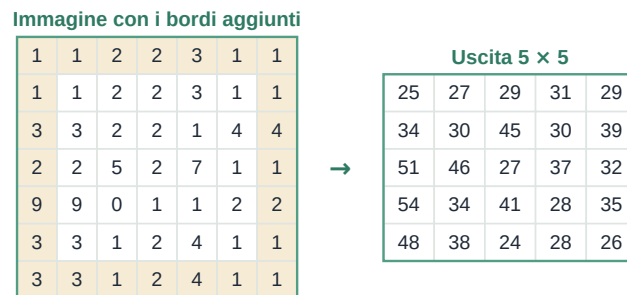


Figura 6. Bordi replicati con i valori più vicini. Le celle in ambra sono le righe e le colonne aggiunte.

Le tre uscite 5 × 5 coincidono nella zona centrale 3 × 3 (i valori 30, 45, 30 / 46, 27, 37 / 34, 41, 28) e differiscono solo sul bordo, cioè nella cornice di 16 pixel.

BORDO	USCITA	PREGI E DIFETTI
Solo zone centrali	più piccola	nessuna ipotesi sui dati, ma si perde un contorno
Zeri	stessa dimensione	semplice, ma crea un contorno scuro (artefatto)
Toroidale	stessa dimensione	adatta a immagini periodiche; per altre inventa informazione
Replica	stessa dimensione	di solito la scelta più naturale per le fotografie

Tabella 3. Confronto tra i metodi di gestione dei bordi.

CORREZIONE

Nelle slide del corso l'uscita con bordi replicati riporta 33 (riga 1, colonna 5), 32 (riga 5, colonna 2) e 34 (riga 5, colonna 4). Calcolando a mano i prodotti si ottengono 29, 38 e 28, come nella Figura 6. Per esempio, per la riga 5, colonna 2: il kernel copre le righe 4, 5 e la riga 5 replicata, colonne 1, 2, 3; i prodotti sono $(9 \cdot 2 + 0 \cdot 1 + 1 \cdot 2) + (3 \cdot 1 + 1 \cdot 2 + 2 \cdot 1) + (3 \cdot 2 + 1 \cdot 1 + 2 \cdot 2) = 20 + 7 + 11 = 38$.

DA RICORDARE

Un operatore lineare soddisfa $F(a v_1 + b v_2) = a F(v_1) + b F(v_2)$, e se è anche invariante per traslazione è descritto da un solo kernel, la sua risposta all'impulso. Il filtraggio è la somma dei prodotti tra kernel e pixel sovrapposti, ripetuta per ogni pixel; ai bordi si sceglie una regola (solo centro, zeri, toroidale, replica).

Noise reduction

Come si elimina il rumore da una immagine: filtri di media e binomiali, filtri d'ordine (mediano, minimo, massimo), rumore impulsivo e gaussiano, e come si sceglie il filtro e la dimensione del kernel.

Operatori locali per ridurre il rumore

Gli **operatori locali** hanno un valore di uscita, per ogni pixel, che dipende da un limitato intorno del punto in ingresso. Si usano per migliorare la qualità delle immagini, per esempio togliendo il rumore, o per estrarre informazioni, come i contorni (capitolo 24). Si dividono in **lineari**, quelli che si descrivono con un kernel (capitolo 22), e **non lineari**, come i filtri d'ordine. Per gli esempi di questo capitolo si usa una fotografia in scala di grigi di 512×512 pixel, con bordi replicati.

I filtri d'ordine: mediano, minimo e massimo

Il **filtro mediano** è un filtro **non lineare** che fornisce in uscita il valore mediano dell'intorno del pixel: si ordinano in ordine crescente i valori dell'intorno e si prende quello centrale. Il filtro mediano, il minimo e il massimo sono filtri statistici detti **order statistics** (statistiche d'ordine).

- il **filtro di minimo**, dato un intorno $m \times m$ di un pixel (con m in genere dispari), sostituisce il valore del pixel con il valore minimo di tutti quelli osservati nell'intorno. Il risultato è un **incupimento** dell'immagine, perché si eliminano, per esempio, le macchioline chiare;
- il **filtro di massimo** sostituisce il pixel con il valore massimo dell'intorno. Il risultato è uno **schiarimento** dell'immagine, perché si eliminano, per esempio, i punti neri.

Esempio. Per il pixel centrale della finestra 3×3 con righe (2, 2, 3), (2, 2, 1), (5, 2, 7), i nove valori ordinati sono 1, 2, 2, 2, 2, 2, 3, 5, 7. Il mediano è il quinto, cioè **2**; il minimo è 1, il massimo è 7; la media sarebbe $26 / 9 \approx 2,89$, un valore che non compare nella finestra.



Figura 4. Filtri d'ordine con un intorno 5×5 . Il minimo scurisce e allarga le zone scure, il massimo schiarisce e allarga le zone chiare, il mediano leviga i dettagli piccoli senza cambiare la luminosità media.

I filtri di media

N BOX

I filtri **N box** (o di media) hanno un kernel $N \times N$ con ogni elemento pari a $1 / N^2$; N è generalmente dispari. Hanno l'effetto di **sfocare** le immagini. La sfocatura è molto forte in orizzontale e in verticale, ma meno in diagonale.

N BINOMIALE

I filtri **N binomiali** sono filtri di smussamento con kernel derivati dalla distribuzione binomiale. Poiché questa è una approssimazione discreta della distribuzione gaussiana, sono anche detti filtri **gaussiani**. Hanno il pregio di smussare **egualmente in tutte le direzioni** e smussano meno vigorosamente dei box. Il kernel si costruisce come prodotto di una riga e di una colonna del triangolo di Tartaglia: (1, 2, 1) per il 3 binomiale, (1, 4, 6, 4, 1) per il 5 binomiale, divisi per la somma dei pesi.

Diagram illustrating the multiplication of three matrices to form a 5x5 matrix:

- Matrix 1 (3x3): **3 box**, value $\frac{1}{9}$.

1	1	1
1	1	1
1	1	1
- Matrix 2 (3x3): **3 binomiale**, value $\frac{1}{16}$.

1	2	1
2	4	2
1	2	1
- Matrix 3 (5x5): **5 box**, value $\frac{1}{25}$.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
- Result Matrix (5x5): **5 binomiale**, value $\frac{1}{256}$.

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

Figura 1. I kernel di media N box e N binomiale più usati: davanti a ciascuna matrice il fattore che la normalizza. Gli elementi dei kernel binomiali sono i prodotti delle righe del triangolo di Tartaglia.

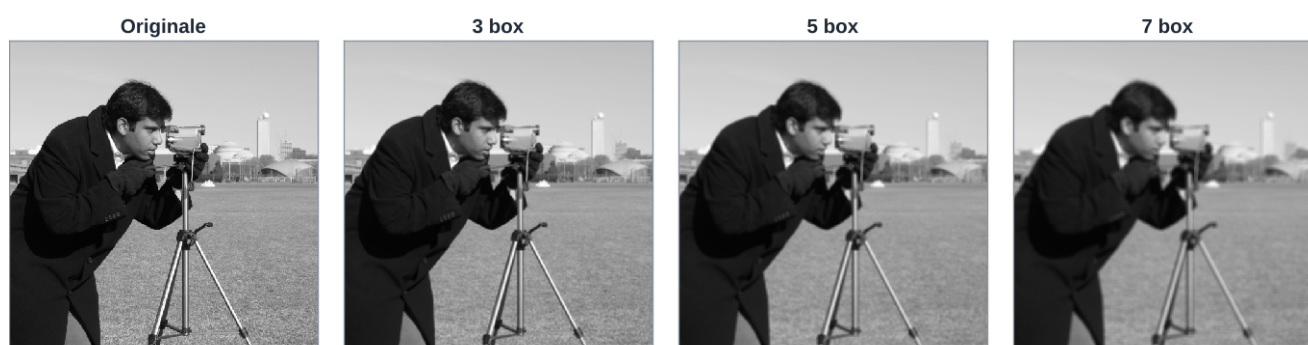


Figura 2. Effetto dei filtri N box: più grande è il kernel, più forte è la sfocatura.

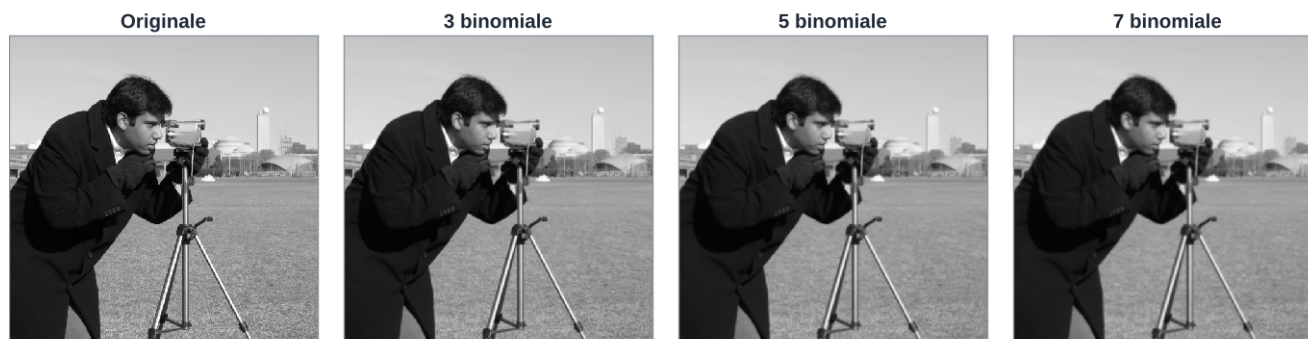


Figura 3. Effetto dei filtri N binomiali: smussano in modo più uniforme in tutte le direzioni e meno vigorosamente dei box.

CONSERVAZIONE DELL'ENERGIA

Con un abuso di linguaggio, derivato dalla fisica, si dice che un filtro **conserva l'energia** (*energy preserving*) se la **somma dei suoi pesi fa 1**. Tutti i filtri di smussamento visti sono di questo tipo: la somma dei valori di luminanza nell'immagine non cambia, quindi la luminosità media resta uguale. Nella fotografia usata, la media dei grigi è 129,06 prima e 129,06 dopo un 5 box. Esistono anche filtri che non conservano l'energia (capitolo 24: i filtri di bordo hanno somma dei pesi nulla).

Un filtro di media riduce anche il **rumore**. Se il rumore è indipendente da pixel a pixel con deviazione standard σ , un filtro con pesi h produce un rumore con deviazione standard $\sigma \cdot \sqrt{\sum h^2}$. Per il 3 box si ha $\sigma \cdot \sqrt{9 \cdot (1/9)^2} = \sigma / 3$, e in generale σ / N per un N box. Su un rumore gaussiano con $\sigma = 20$ la prova numerica dà 6,7 per il 3 box ($20 / 3$), 4,0 per il 5 box ($20 / 5$) e 7,5 per il 3 binomiale ($20 \cdot 6/16$).

Noise cleaning e smoothing

I filtri visti servono sia per **ridurre il rumore** (*noise cleaning*) sia per **sfocare** l'immagine (*smoothing*). Nel primo caso, più è grande il kernel e migliore è il risultato, anche se si rischia di aumentare la sfocatura. Nel secondo, più è grande il kernel e maggiore è la sfocatura, ma si riduce anche meglio il rumore. Sono le due facce della stessa operazione: ridurre il rumore con un filtro di media significa perdere anche i dettagli fini.

Il rumore

Ci sono due tipi principali di rumore.

- **Rumore impulsivo**, o **sale e pepe** (*salt and pepper*). Alcuni pixel assumono un valore «saturato», uguale al minimo ($a = 0$) o al massimo ($b = 255$) dell'immagine. Si caratterizza con la **frazione dell'immagine modificata** (in percentuale);
- **Rumore gaussiano bianco**. A ogni pixel si somma un numero casuale con distribuzione gaussiana, indipendente da pixel a pixel. Si caratterizza con la **media** (di solito 0) e la **varianza** (o la deviazione standard).

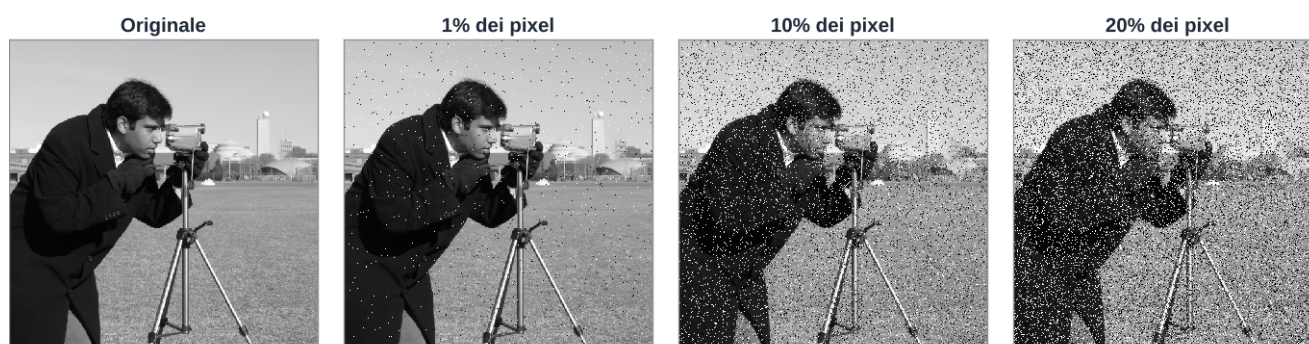


Figura 5. Rumore sale e pepe con il 1%, il 10% e il 20% dei pixel danneggiati.



Figura 6. Rumore gaussiano bianco con media 0 e deviazione standard 20 (varianza 400).

Rimozione del rumore

Quale filtro usare dipende dal tipo di rumore. Per il rumore **sale e pepe** funziona meglio il filtro **mediano**; per il rumore **gaussiano** funziona meglio il filtro **di media** (o binomiale).



Figura 7. Rimozione del rumore sale e pepe (10%) con un filtro di media e con un filtro mediano, entrambi 3 × 3. Il mediano lo elimina quasi del tutto; la media lo spalma in macchie grigie.



Figura 8. Rimozione del rumore gaussiano con gli stessi filtri. Qui la media ottiene un risultato leggermente migliore.

RUMORE	IMMAGINE RUMOROSA	MEDIA 3 × 3	MEDIANO 3 × 3
Sale e pepe 1%	24,6	28,1	30,5
Sale e pepe 10%	14,8	22,4	29,6
Sale e pepe 20%	11,7	19,3	26,8
Gaussiano $\sigma = 10$	28,2	28,8	29,3
Gaussiano $\sigma = 20$	22,4	27,4	27,0
Gaussiano $\sigma = 30$	19,1	25,8	24,8

Tabella 2. PSNR in dB rispetto all'originale, per diversi rumori e filtri 3 × 3 (fotografia da 512 × 512 pixel). Nei risultati sul rumore gaussiano il vantaggio della media dipende dal livello di rumore.

Con il rumore sale e pepe la differenza è netta a ogni livello (con il 10% il mediano ottiene un PSNR di circa 29,6 dB contro 22,4 dB della media). Con quello gaussiano la media è migliore solo a partire da una certa intensità del rumore: con $\sigma = 10$ il mediano è leggermente più bravo, con $\sigma = 20$ e 30 vince la media. In nessuno dei due casi i filtri recuperano la qualità originale: ogni elaborazione elimina il rumore ma anche parte del dettaglio.

PERCHÉ IL MEDIANO FUNZIONA MEGLIO CON SALE E PEPE

- il filtro di media tende a creare dei livelli di grigio **prima non esistenti**: un pixel nero isolato in una zona chiara diventa un grigio un po' più scuro, e quello stesso «errore» viene spalmato sui pixel vicini;
- il filtro di media non attenua solo il rumore ma tutte le **alte frequenze spaziali** in maniera indiscriminata, dando origine a immagini sfocate;
- il filtro mediano **non deteriora i lati** (le transizioni nette) ed **elimina i picchi** con base piccola rispetto al kernel: un picco di larghezza minore di metà kernel non arriva mai a essere il valore centrale dell'ordinamento.

Un esempio con una sola riga di pixel e un intorno di tre: un segnale con un picco isolato (200) seguito da un gradino (da 10 a 100).

Filtro	1	2	3	4	5	6	7	8	9	10
Ingresso	10	10	10	200	10	10	100	100	100	100
Media su 3	10	10	73,3	73,3	73,3	40	70	100	100	100
Mediano su 3	10	10	10	10	10	10	100	100	100	100

Tabella 3. Filtri su una riga di 10 pixel (posizioni da 1 a 10, bordi replicati). Il mediano cancella il picco e lascia il gradino intatto; la media abbassa il picco ma lo allarga su tre pixel e trasforma il gradino in una rampa con valori intermedi (40 e 70).

DIMENSIONI DEL KERNEL

Se il rumore è molto diffuso, è meglio un kernel più grande oppure applicare iterativamente lo stesso kernel? Con il rumore sale e pepe al 20%, si confronta il mediano 3×3 applicato due volte con il mediano 5×5 applicato una volta.

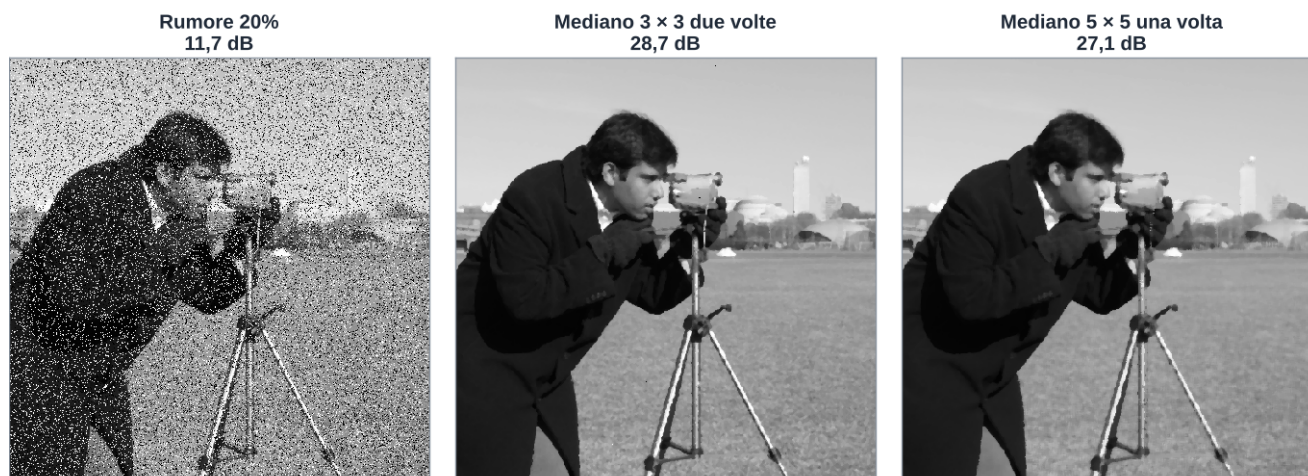


Figura 9. Con il 20% di rumore sale e pepe, il filtro 3×3 applicato due volte lascia l'immagine più nitida del filtro 5×5 applicato una volta.

In entrambi i casi il rumore viene eliminato in larga parte, ma il primo approccio **sfoca di meno** l'immagine finale: il PSNR è 28,7 dB contro 27,1 dB. Il costo è una elaborazione in più, ma il kernel 3×3 ha 9 elementi contro i 25 del 5×5 , quindi le due operazioni costano meno di una sola con il kernel grande.

Altri filtri per la rimozione del rumore

Esistono altri filtri non lineari molto importanti:

- **outlier**: il valore del pixel centrale viene confrontato con la **media dei suoi 8 vicini**. Se il valore assoluto della differenza è maggiore di una soglia, il punto viene sostituito dal valore medio, altrimenti non viene modificato;
- **olimpico**: da un dato intorno si scartano i valori **massimo e minimo** e sul resto si fa la media. Il nome richiama i punteggi delle gare olimpiche, in cui si scartano il voto più alto e quello più basso.

L'outlier modifica solo i pixel sospetti e lascia intatti gli altri, e perciò preserva i dettagli. Con una soglia di 50, su un rumore sale e pepe del 5% (PSNR 17,7 dB) si ottiene 29,7 dB con l'outlier, 30,1 dB con il mediano 3×3 e 28,4 dB con l'olimpico. Su un rumore gaussiano con $\sigma = 20$ (PSNR 22,4 dB) l'olimpico dà 27,5 dB, contro 27,4 dB della media e 27,0 dB del mediano: l'olimpico è una media senza i due valori estremi, quindi resiste a un impulso isolato per volta e resta alla pari con la media sul rumore gaussiano.

FILTRO	LINEARE	RUMORE PER CUI È ADATTO
N box, N binomiale	sì	gaussiano; sfocano anche i dettagli
Mediano	no	sale e pepe; conserva i lati
Minimo, massimo	no	macchie chiare o scure isolate; alterano la luminosità
Outlier	no	picchi isolati, pochi pixel modificati
Olimpico	no	gaussiano e impulsi isolati; media senza i due estremi

Tabella 4. Riepilogo dei filtri per la riduzione del rumore.

DA RICORDARE

Il filtro mediano è non lineare ed è il migliore contro il rumore sale e pepe; il filtro di media (box o binomiale) è lineare, conserva la luminosità media e riduce il rumore gaussiano di un fattore N per un box $N \times N$, a costo di sfocare l'immagine. Più grande è il kernel, più rumore si elimina e più dettaglio si perde.

Edge detection

Come si trovano i contorni di una immagine: la derivata prima con i filtri di Sobel e Prewitt, il modulo del gradiente, la derivata seconda con il Laplaciano e lo zero crossing, e il filtro di sharpening.

I contorni di una immagine

Gli operatori locali ci aiutano a estrarre i **contorni** (*edges*, lati) da una immagine. I contorni sono definiti come **discontinuità locali della luminanza**. Gli **edge detector** forniscono immagini in cui sono preservate le variazioni di luminanza ed eliminate tutte le altre informazioni. In una dimensione un lato può essere un gradino netto, una rampa (passaggio graduale) o una linea sottile.

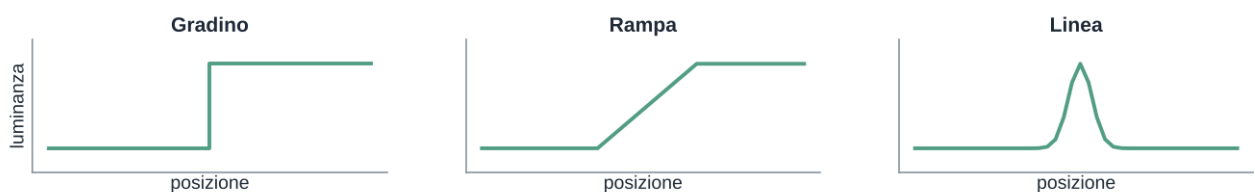


Figura 1. Alcuni esempi di lato in una dimensione: gradino, rampa, linea.

Un lato è una zona in cui la luminanza cambia rapidamente, e un cambiamento rapido si misura con una **derivata**. Per questo i due approcci principali si basano sulla derivata prima e sulla derivata seconda.



Figura 2. Un profilo di luminanza con un lato, la sua derivata prima (che ha un massimo in corrispondenza del lato) e la sua derivata seconda (che passa per lo zero in corrispondenza del lato, con un valore positivo da una parte e negativo dall'altra).

Edge detector basati sulla derivata prima

Se si ha un segnale monodimensionale e se ne calcola la derivata prima, i lati si trovano in corrispondenza dei **massimi** (in valore assoluto) della derivata. Nel caso discreto la derivata si approssima con una differenza tra campioni vicini. Per un'immagine servono due derivate, una in direzione x e una in direzione y , da **combinare** poi insieme.

I filtri più usati sono i kernel di **Prewitt** e di **Sobel**. Entrambi calcolano la differenza tra le due righe (o le due colonne) ai lati del pixel e, contemporaneamente, mediano lungo la direzione ortogonale, per ridurre l'effetto del rumore: la media è uniforme nel Prewitt e pesata con $(1, 2, 1)$ nel Sobel, che dà più peso alla riga centrale.

Con la notazione del corso, il pedice x indica il filtro che rileva i **lati orizzontali** (differenza tra righe), il pedice y quello che rileva i **lati verticali** (differenza tra colonne):

Sobel x	Prewitt x	Sobel y	Prewitt y
-1 -2 -1	-1 -1 -1	-1 0 1	-1 0 1
0 0 0	0 0 0	-2 0 2	-1 0 1
1 2 1	1 1 1	-1 0 1	-1 0 1

Figura 3. I kernel di Sobel e di Prewitt. I filtri per i lati verticali sono quelli per i lati orizzontali ruotati di 90 gradi.

La somma dei pesi di questi kernel è **zero**: in una zona uniforme la risposta è nulla, mentre un salto di luminanza produce una risposta diversa da zero. Questi filtri non conservano quindi l'energia, e il loro risultato ha valori positivi e negativi (il segno indica se si passa da scuro a chiaro o viceversa); per visualizzarlo si usa spesso il valore assoluto, eventualmente seguito da una equalizzazione o da un'amplificazione.

Immagine con un lato verticale		Uscita Sobel y	Uscita Sobel x
10 10 10 100 100 100	→	0 0 360 360 0 0	0 0 0 0 0 0
10 10 10 100 100 100		0 0 360 360 0 0	0 0 0 0 0 0
10 10 10 100 100 100		0 0 360 360 0 0	0 0 0 0 0 0
10 10 10 100 100 100		0 0 360 360 0 0	0 0 0 0 0 0

Figura 4. Un lato verticale da 10 a 100 produce una risposta di Sobel y pari a $4 \cdot 90 = 360$ nelle due colonne a cavallo del lato, e una risposta nulla di Sobel x.

Il valore massimo teorico della risposta a un gradino di ampiezza A è $4 \cdot A$ per il Sobel e $3 \cdot A$ per il Prewitt (nell'immagine a 8 bit, fino a 1020 e 765 rispettivamente).

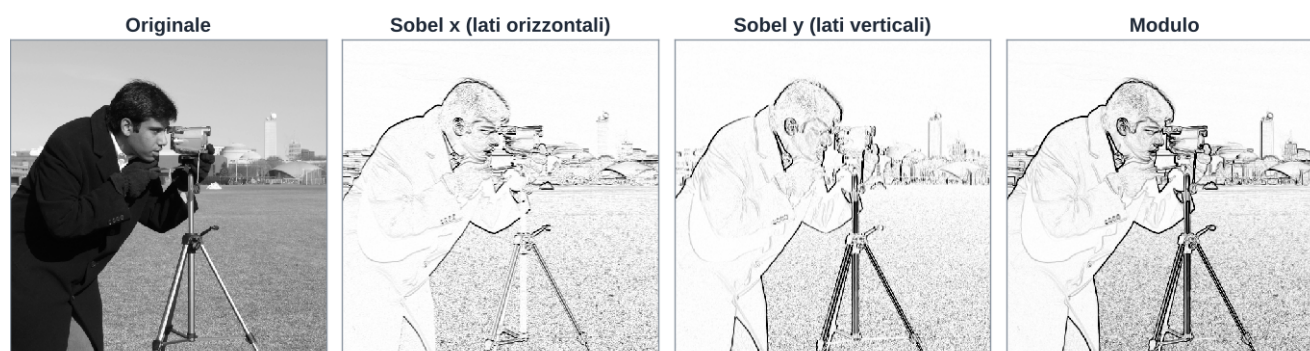


Figura 5. Risposta dei due filtri di Sobel e del modulo, in valore assoluto e su fondo bianco (più è scuro il pixel, più il lato è forte; valori oltre 400 saturati).

IL MODULO DEL GRADIENTE

Sobel x fornisce una matrice in cui i lati orizzontali (e le componenti orizzontali dei lati obliqui) hanno valori non nulli; Sobel y fa lo stesso con i lati verticali. Le due matrici si combinano con la formula del **modulo** (o magnitudo del gradiente):

$$\text{modulo} = \sqrt{(\text{sobel}_x)^2 + (\text{sobel}_y)^2}$$

La matrice ottenuta ha valori non nulli per i pixel «di lato». Fissando una **soglia** adeguata si ottiene una matrice binaria che indica, per ogni pixel, se è o non è di lato. Le stesse considerazioni valgono per il Prewitt. Per risparmiare calcoli si usa a volte la stima $|\text{sobel}_x| + |\text{sobel}_y|$.

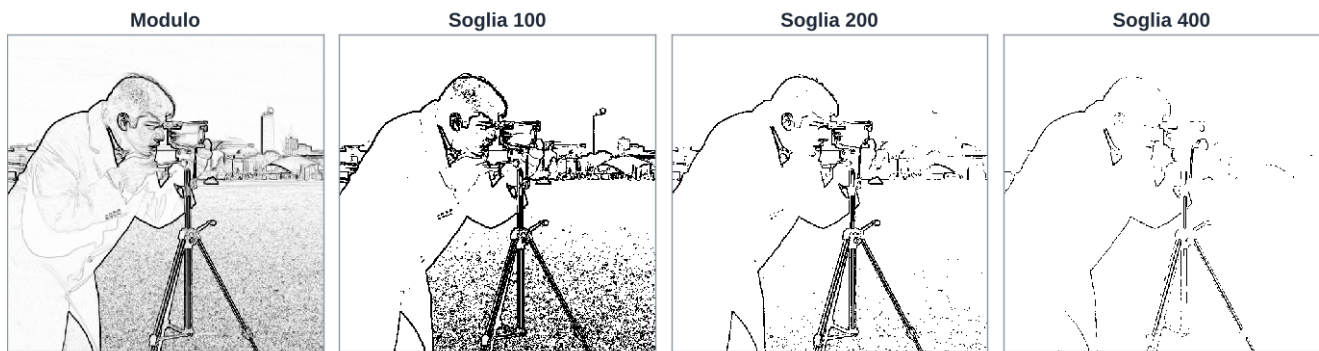


Figura 6. Mappa binaria dei lati ottenuta fissando una soglia sul modulo di Sobel. Una soglia bassa lascia anche tessiture e rumore, una alta spezza i contorni.

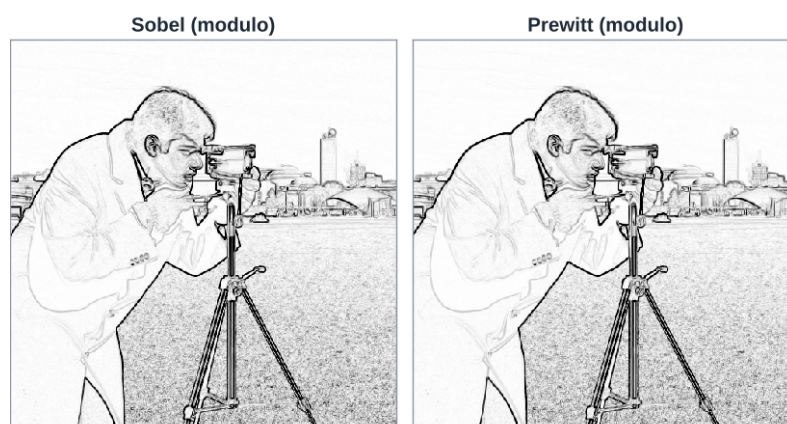


Figura 7. Modulo di Sobel e di Prewitt, ciascuno scalato con la propria somma dei pesi (4 e 3). Le differenze sono modeste.

Il modulo di Sobel e quello di Prewitt sono molto simili (coefficiente di correlazione 0,996): la differenza principale è che il Sobel dà un po' più peso ai pixel vicini al centro ed è quindi meno sensibile al rumore. Con una soglia di 100, 200 e 400 il 13,8%, il 5,0% e il 1,6% dei pixel della fotografia risultano di lato.

Si ottengono risultati migliori con algoritmi più sofisticati per il calcolo della grandezza del gradiente, e con strategie più «intelligenti», come l'**algoritmo di Canny**, gli algoritmi fuzzy, le tecniche di backtracking, che assottigliano e collegano i contorni.

Edge detector basati sulla derivata seconda

Se si calcola la derivata seconda di un segnale monodimensionale, in corrispondenza del lato essa **passa per lo zero**, con un valore positivo da un lato e negativo dall'altro (Figura 2). Nel caso discreto la derivata prima è $f(x) - f(x - 1)$ e la derivata seconda, differenza di due derivate prime successive, è $f(x + 1) - 2f(x) + f(x - 1)$. Un esempio con una rampa:

Posizione	1	2	3	4	5	6	7	8
Luminanza	10	10	10	40	70	100	100	100
Derivata prima	0	0	0	+30	+30	+30	0	0
Derivata seconda	0	0	+30	0	0	-30	0	0

Tabella 1. Derivate discrete di una rampa da 10 a 100. La derivata prima è costante (30) lungo la rampa; la derivata seconda è +30 all'inizio, -30 alla fine e 0 tra i due: il lato sta dove il segno cambia.

Il filtro più diffuso per calcolare la derivata seconda è il **Laplaciano**. Nella forma più comune somma le derivate seconde lungo x e lungo y, con kernel a 4 vicini, che qui si scrive con il segno cambiato (centro positivo). Il corso

usa una variante con i quattro vicini **diagonali**, definita dalla maschera

Del corso			A 4 vicini		
-1	0	-1	0	-1	0
0	4	0	-1	4	-1
-1	0	-1	0	-1	0

Figura 8. Il kernel del Laplaciano usato nel corso (vicini diagonali) e quello a 4 vicini più comune. Hanno somma dei pesi zero e, a meno della rotazione di 45 gradi, producono lo stesso tipo di risposta.

LO ZERO CROSSING

Dopo aver applicato il Laplaciano non basta cercare i valori nulli: bisogna che si verifichi la condizione di **zero crossing** (passaggio per lo zero). Cioè, rispetto al punto in questione, deve esserci nel suo intorno **un valore positivo e un valore negativo**. Per evitare di segnalare le piccole oscillazioni, si richiede in pratica che la differenza tra i due valori sia maggiore di una soglia.

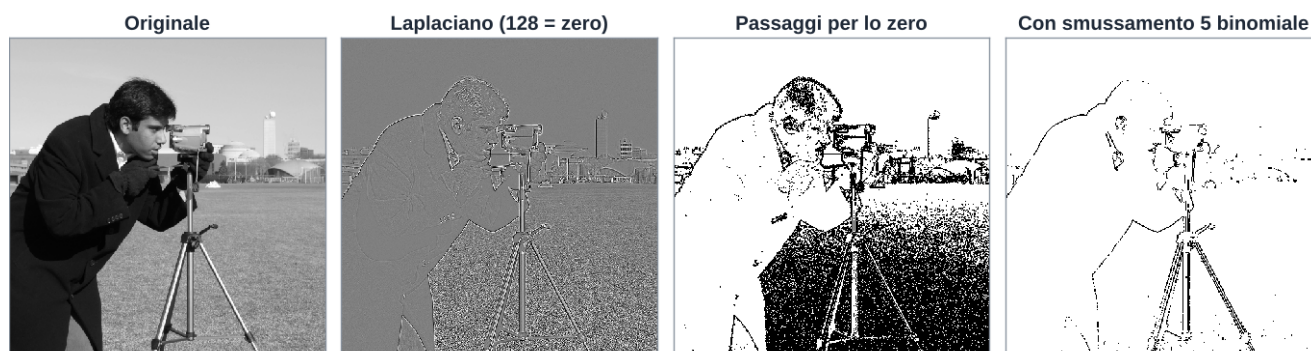


Figura 9. Laplaciano della fotografia (grigio medio dove la risposta è nulla), passaggi per lo zero calcolati direttamente e dopo uno smussamento 5 binomiale: il rumore produce moltissimi falsi lati nel primo caso.

Il Laplaciano è una derivata seconda, e perciò è molto **sensibile al rumore**: nella Figura 9 i passaggi per lo zero calcolati direttamente sull'immagine segnano il 30,0% dei pixel, mentre dopo uno smussamento con il filtro 5 binomiale (capitolo 23) scendono al 2,7% e diventano veri contorni. La combinazione di smussamento gaussiano e Laplaciano si chiama infatti *Laplacian of Gaussian* (LoG).

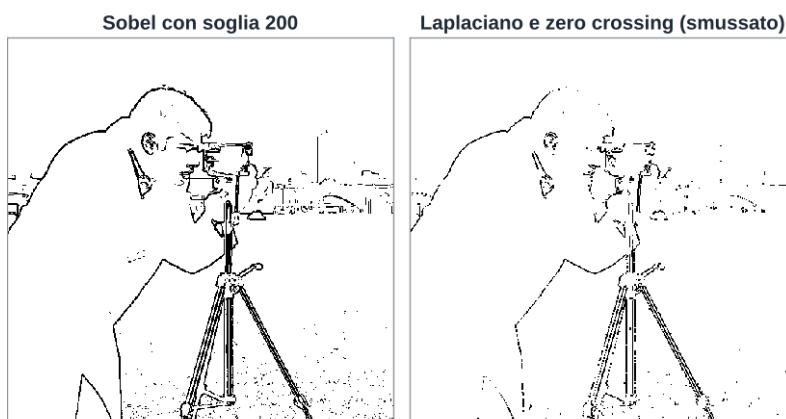


Figura 10. I due approcci a confronto. Sobel dà contorni più spessi e frammentati, ma più stabili; lo zero crossing dà contorni sottili e chiusi, ma sensibili al rumore.

	DERIVATA PRIMA (SOBEL, PREWITT)	DERIVATA SECONDA (LAPLACIANO)
Cerca	massimi del modulo	passaggi per lo zero
Direzione	una per ogni asse, da combinare	unica, senza direzione
Contorno	spesso, serve una soglia	sottile, spesso chiuso
Rumore	sensibile	molto sensibile, serve smussare

Tabella 2. Confronto tra i due tipi di edge detector.

Filtri di sharpening

I filtri di **sharpening** (incremento della nitidezza) hanno lo scopo di aumentare la nitidezza di una immagine **aumentando il contrasto locale**. È una operazione opposta allo sfocamento. Per ottenere l'effetto si usa una maschera derivata dal Laplaciano che «rinforza» i lati presenti nell'immagine. Si ottiene sommando all'immagine il suo Laplaciano: la maschera è la somma del kernel che lascia l'immagine inalterata (1 al centro) e del Laplaciano del corso,

Identità		Laplaciano		Sharpening
0 0 0		-1 0 -1		-1 0 -1
0 1 0	+	0 4 0	=	0 5 0
0 0 0		-1 0 -1		-1 0 -1

Figura 11. La maschera di sharpening come somma dell'identità e del Laplaciano.

La somma dei pesi è 1: il filtro **conserva l'energia**, cioè lascia invariata la luminosità media, ed esalta solo le variazioni locali. Purtroppo rinforza anche il **rumore** presente nell'immagine.

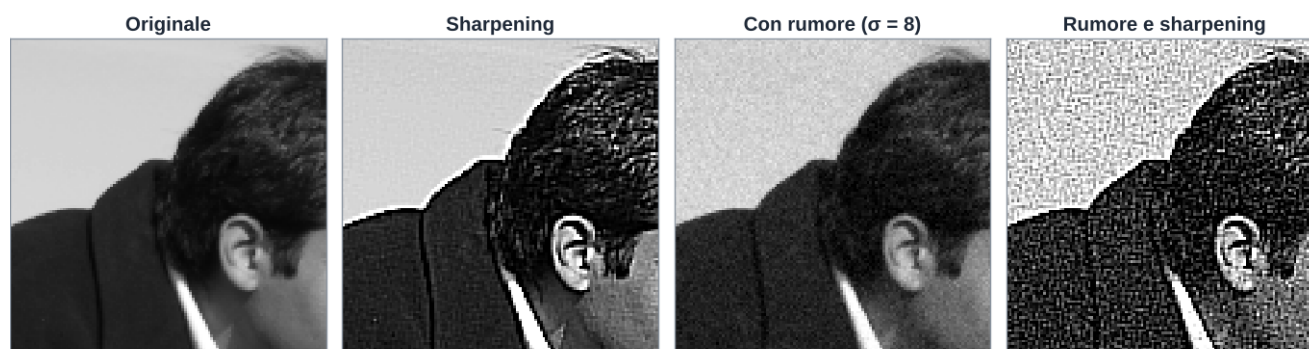


Figura 12. Ritaglio di 120×120 pixel. Il filtro rende più netti i contorni, ma rinforza anche il rumore: nella colonna di destra il rumore è molto più evidente di prima.

DA RICORDARE

I lati sono discontinuità della luminanza. Sobel e Prewitt approssimano la derivata prima (due kernel, combinati nel modulo $\sqrt{x^2 + y^2}$, poi una soglia); il Laplaciano approssima la derivata seconda (si cerca il passaggio per lo zero) ed è più sensibile al rumore. Sommando il Laplaciano all'immagine si ottiene il filtro di sharpening, che aumenta il contrasto locale ma anche il rumore.

Dominio spaziale e dominio delle frequenze

Due modi di descrivere la stessa immagine: come pixel in una griglia e come somma di onde. Segnali, frequenza, frequenza spaziale, basse e alte frequenze, e perché conviene cambiare dominio.

Il dominio spaziale

Finora una immagine è stata trattata come una matrice di valori $f(x, y)$, con x e y posizioni nella griglia dei pixel. Lavorare sulla matrice significa lavorare nel **dominio spaziale**: ogni operazione agisce direttamente sui pixel. Rientrano in questo dominio le operazioni dei capitoli precedenti:

- le operazioni **puntuali** (capitolo 20), in cui il nuovo valore dipende solo dal valore del pixel;
- le operazioni **locali** (capitoli da 22 a 24), in cui dipende da un intorno del pixel, descritto da un kernel o da una statistica;
- le operazioni **globali**, in cui ogni pixel dell'uscita dipende da tutta l'immagine.

Il dominio spaziale risponde bene a domande locali: «quanto è luminoso questo punto?», «c'è un lato qui?». Meno bene a domande del tipo «con quale ritmo varia la luminosità in tutta l'immagine?» o «c'è una trama che si ripete?», perché la risposta riguarda l'immagine intera. A queste domande risponde il dominio delle frequenze.

Segnali e frequenza

Una funzione **periodica** si può esprimere come somma di seni e/o coseni di differenti **frequenze e ampiezze** (serie di Fourier). Anche una funzione non periodica può, sotto certe condizioni, essere espressa come un integrale di seni e coseni moltiplicati per opportune funzioni peso: è la **trasformata di Fourier**, dal nome di Jean Baptiste Joseph Fourier (Auxerre 1768, Parigi 1830). L'analisi di Fourier nacque nello studio della diffusione del calore, dove permise di formulare e risolvere in modo originale le equazioni differenziali di alcuni fenomeni fisici.

Un segnale che varia con il tempo, come un suono, si può rappresentare in due modi. Nel **dominio del tempo** si descrive il valore del segnale in ogni istante. Nel **dominio della frequenza** si descrive quanto di ogni frequenza è presente. Un'onda sinusoidale $A \sin(2\pi f t + \varphi)$ è caratterizzata da:

- l'**ampiezza** A , il valore massimo raggiunto dall'onda, nell'unità del segnale (se espressa su scala logaritmica si usano i decibel, dB);
- il **periodo** T , il tempo di un ciclo, in secondi;
- la **frequenza** $f = 1 / T$, il numero di cicli al secondo, misurato in hertz (Hz);
- la **fase** φ , che indica dove l'onda inizia il suo ciclo.

CORREZIONE

Nelle slide l'ampiezza è indicata «espressa in decibel». In realtà l'ampiezza si misura nelle unità del segnale (per esempio volt, o livelli di grigio per un'immagine); il decibel è una unità **logaritmica relativa**, usata per esprimere rapporti tra ampiezze, come nel PSNR del capitolo 13.

Esempio. Si sommano quattro sinusoidi di frequenza 1, 3, 5 e 7 Hz e ampiezze 1, 1/3, 1/5 e 1/7. La somma ha una forma a gradini, simile a un'onda quadra. Nel dominio del tempo il risultato è una curva complicata; nel dominio della frequenza è semplicemente quattro barre, una per ogni componente, con la loro altezza.

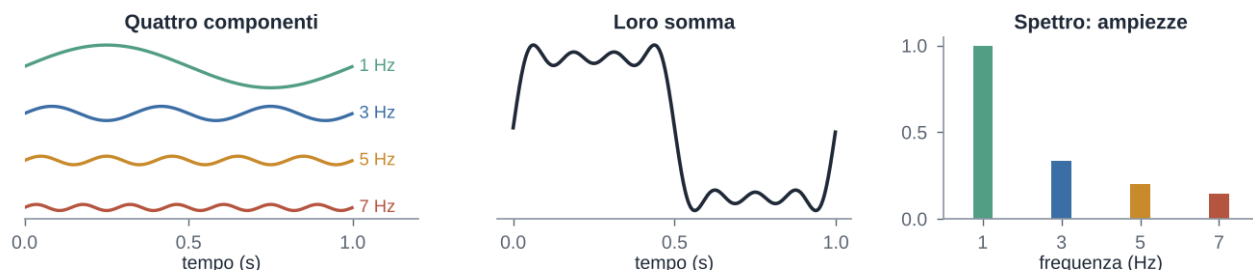


Figura 1. Quattro funzioni periodiche (a sinistra), la loro somma (al centro) e la rappresentazione nel dominio della frequenza: un grafico delle ampiezze in funzione delle frequenze.

Calcolando la trasformata discreta della somma campionata con 256 campioni, le quattro ampiezze risultano 1,000, 0,333, 0,200 e 0,143, esattamente quelle delle componenti, mentre alla frequenza di 2 Hz, che non è presente, l'ampiezza è 0,000.

Sia la serie sia la trasformata di Fourier condividono un fatto fondamentale: la funzione si può **ricostruire** con un semplice processo di inversione, **senza perdita di informazione**. Si può cioè lavorare nel cosiddetto «dominio di Fourier» e tornare al dominio originale in modo del tutto naturale. Il dominio delle frequenze non aggiunge né toglie informazione: è una rappresentazione **alternativa** dello stesso segnale.

Le immagini come segnali

Una immagine è una funzione discreta in due dimensioni, i cui valori rappresentano il livello di grigio di un pixel. Si può quindi vedere come un **segnale**, cioè una funzione variabile in un dominio con una propria frequenza, costante o variabile. In questo caso il dominio non è il tempo ma lo **spazio**, e si parla di **frequenza spaziale**: quante volte per unità di lunghezza la luminosità ripete il suo ciclo.

Per una immagine di $M \times N$ pixel, una sinusoide con u cicli lungo la larghezza e v cicli lungo l'altezza ha frequenza spaziale (u, v) e periodo di M / u pixel in orizzontale e N / v pixel in verticale. La frequenza massima rappresentabile è di mezzo ciclo per pixel (un pixel chiaro, uno scuro, uno chiaro...), cioè $u = M / 2$: è il limite di Nyquist del capitolo 8. In un'immagine ogni sinusoide ha anche una **direzione**: le righe chiare e scure sono perpendicolari al vettore (u, v) .

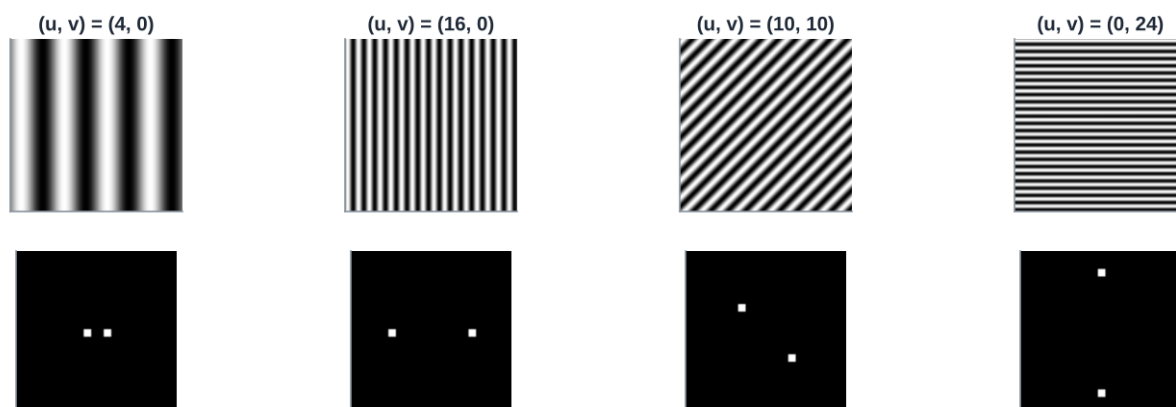


Figura 2. Quattro immagini di 256×256 pixel fatte di una sola sinusoide (sopra) e il loro spettro (sotto, nella zona centrale; la componente continua è esclusa). Ogni sinusoide dà due punti simmetrici rispetto al centro: la distanza dal centro misura la frequenza, la direzione ne indica l'orientamento.

Nella prima immagine ci sono 4 cicli su 256 pixel (periodo di 64 pixel: righe larghe), nella seconda 16 (periodo di 16 pixel: righe più fitte), nella terza 10 cicli in orizzontale e 10 in verticale (righe oblique, periodo $256 / \sqrt{200} \approx 18,1$ pixel), nella quarta 24 cicli in verticale (periodo di circa 10,7 pixel).

Basse e alte frequenze

Una immagine reale non è una sinusoide, ma una somma di molte. Escludendo i casi banali è normalmente **impossibile** associare direttamente parti specifiche dell'immagine a parti della sua trasformata (si ha una **perdita di localizzazione spaziale**): ogni frequenza dipende da tutta l'immagine. Ricordando però che la frequenza è legata alla **velocità di variazione**, si può dire che:

- le **basse frequenze** corrispondono alle zone uniformi e alle variazioni lente (il cielo, una superficie liscia, la forma generale degli oggetti);
- le **alte frequenze** corrispondono alle variazioni più o meno brusche, cioè ai bordi, ai dettagli fini e al **rumore**.



Figura 3. L'immagine come somma di una parte a basse frequenze, che contiene le zone uniformi e le grandi forme, e di una parte ad alte frequenze, che contiene bordi, dettagli e rumore.

Per questo un filtro di sfocatura, come la media (capitolo 23), è un filtro che **lascia passare le basse frequenze** e riduce le alte, mentre un filtro di bordo, come il Sobel (capitolo 24), fa il contrario. Anche l'aliasing del capitolo 10 ha una spiegazione naturale in questo dominio: frequenze troppo alte per il passo di campionamento si ripiegano su frequenze basse.

Perché cambiare dominio

Che vantaggio si ottiene dalla trasformata di Fourier? Nello spazio delle frequenze è possibile:

- **sopprimere frequenze indesiderate**, per esempio il disturbo periodico di una scansione, che nel dominio spaziale è sparso su tutta l'immagine ma nel dominio delle frequenze è concentrato in pochi punti;
- **ridurre lo spazio occupato** dai dati, limitando la degenerazione del segnale: è il principio di JPEG, MPEG, DivX e MP3 (capitoli da 27 a 29);
- **rigenerare segnali degradati**, cioè restaurarli;
- **progettare filtri** in modo più intuitivo, decidendo direttamente quali frequenze tenere e quali scartare.

La trasformazione diretta può essere vista come un processo di **analisi**: il segnale viene scomposto nelle sue componenti elementari, i vettori di base, e i coefficienti della trasformata dicono quanto di ciascuna componente è presente. La trasformazione inversa è un processo di **sintesi**: il segnale si ricostruisce come somma pesata delle componenti di base, con pesi pari ai coefficienti. Il coefficiente è una misura della **correlazione** tra il segnale e il corrispondente vettore di base.

Fourier non è l'unica trasformata di questo tipo. Appartengono alla stessa classe delle **trasformate unitarie**, molto usate nel restauro e soprattutto nella compressione, la trasformata di Walsh, quella di Hadamard, quella del **coseno** (DCT, usata nel JPEG) e quella di Karhunen e Loève (KLT).

DA RICORDARE

Nel dominio spaziale una immagine è una griglia di pixel; nel dominio delle frequenze è una somma di sinusoidi, ciascuna con una frequenza, una direzione, un'ampiezza e una fase. Le basse frequenze descrivono le zone uniformi, le alte frequenze bordi e rumore. Il passaggio da un dominio all'altro non perde informazione: è una rappresentazione diversa dello stesso segnale.

La trasformata di Fourier e la convoluzione

La trasformata discreta di Fourier in una e due dimensioni, modulo e fase, le sue proprietà, la convoluzione e il teorema della convoluzione, e il filtraggio nel dominio delle frequenze con i filtri passa basso, passa alto e band reject.

La trasformata discreta di Fourier

La trasformata discreta di Fourier (**DFT**) di una sequenza di N campioni $f(x)$, per $x = 0, 1, \dots, N-1$, è la sequenza di N numeri **complessi**

$$F(u) = (1/N) \cdot \sum_{x=0 \dots N-1} f(x) \cdot e^{-i 2\pi u x / N}, \quad u = 0, 1, \dots, N-1$$

e la trasformazione inversa, che ricostruisce la sequenza di partenza, è

$$f(x) = \sum_{u=0 \dots N-1} F(u) \cdot e^{i 2\pi u x / N}$$

Qui i è l'unità immaginaria. L'esponenziale complesso si scrive con la **formula di Eulero**: per ogni numero reale θ si ha $e^{i\theta} = \cos \theta + i \sin \theta$, e quindi $e^{-i\theta} = \cos \theta - i \sin \theta$. Ogni $F(u)$ è dunque la correlazione tra il segnale e una coppia di sinusoidi (coseno e seno) di frequenza u cicli sull'intera sequenza. La normalizzazione $1/N$ è quella adottata nel corso; altri testi la mettono nella trasformazione inversa o la dividono tra le due ($1/\sqrt{N}$ in entrambe).

Esempio. Per $N = 4$ e $f = (1, 2, 3, 4)$, i fattori $e^{-i\pi u x / 2}$ valgono per $u = 1$: $1, -i, -1, i$ (per $x = 0, 1, 2, 3$). Allora $F(1) = (1 \cdot 1 + 2 \cdot (-i) + 3 \cdot (-1) + 4 \cdot i) / 4 = (-2 + 2i) / 4 = -0,5 + 0,5i$. In modo analogo si ottengono tutti gli altri valori:

u	$F(u)$	Modulo	Fase (gradi)
0	2,50	2,500	0,0
1	$-0,50 + 0,50 i$	0,707	135,0
2	$-0,50$	0,500	180,0
3	$-0,50 - 0,50 i$	0,707	-135,0

Tabella 1. La DFT di $(1, 2, 3, 4)$. $F(0) = 2,5$ è la media dei quattro valori; $F(3)$ è il complesso coniugato di $F(1)$, come avviene sempre per un segnale reale.

Si noti che $F(0)$ è la **media** dei valori, e che per un segnale reale $F(N-u)$ è il complesso coniugato di $F(u)$: metà dei coefficienti è ridondante. Tutti i valori di f contribuiscono alla costruzione di ciascun campione di F , e, analogamente, tutti i campioni della trasformata contribuiscono, durante l'antitrasformata, a ciascuno dei valori di f .

IL CASO BIDIMENSIONALE

Per una immagine $f(x, y)$ di $M \times N$ pixel la coppia trasformata e antitrasformata è

$$F(u, v) = (1/MN) \cdot \sum_{x=0 \dots M-1} \sum_{y=0 \dots N-1} f(x, y) \cdot e^{-i 2\pi (u x / M + v y / N)}$$

$$f(x, y) = \sum_{u=0 \dots M-1} \sum_{v=0 \dots N-1} F(u, v) \cdot e^{i 2\pi (u x / M + v y / N)}$$

con u e v indici degli assi delle frequenze discretizzati, e M e N dimensioni in pixel dell'immagine. Ogni coefficiente $F(u, v)$ misura quanto è presente nell'immagine la sinusoide di frequenza spaziale (u, v) vista nel capitolo 25.

Modulo, fase e potenza spettrale

Poiché F ha valori complessi, si può esprimere con la parte reale $R(u, v)$ e la parte immaginaria $I(u, v)$:

GRANDEZZA	FORMULA	SIGNIFICATO
Spettro (modulo)	$ F(u, v) = \sqrt{R^2 + I^2}$	ampiezza della sinusoide di frequenza (u, v)
Angolo di fase	$\varphi(u, v) = \arctan(I / R)$	posizione della sinusoide
Potenza spettrale	$P(u, v) = F(u, v) ^2 = R^2 + I^2$	energia associata a quella frequenza

Tabella 2. Le grandezze derivate dalla trasformata. Per la fase si usa in pratica la funzione $\arctan$ a due argomenti, che distingue i quattro quadranti.

L'**ampiezza** contiene l'informazione relativa al fatto che una certa struttura periodica è presente nell'immagine; la **fase** contiene l'informazione sul **dove** le strutture periodiche sono collocate. La fase è quindi molto più significativa di quanto possa sembrare guardando la sua immagine (Figura 3).

IL RANGE DINAMICO

Quando si visualizza lo spettro come immagine di intensità, esso ha in genere una dinamica molto più grande di quella riproducibile su un display, perché i coefficienti a bassa frequenza sono molto maggiori degli altri: normalizzando a 256 livelli, solo pochissime parti molto luminose risultano visibili. Si ovvia, come nel capitolo 20, con una compressione logaritmica: invece dello spettro si visualizza

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

con c costante di scala scelta per far ricadere i valori in $[0, L - 1]$. Se R è il massimo dello spettro, con R molto maggiore di 1 si pone $c \log R = L - 1$, e quindi $c = (L - 1) / \log R$. Nelle slide, per l'immagine di Lena, lo spettro varia da circa 0 a $R = 6,47 \cdot 10^6$ e quindi $c = 255 / \ln(6,47 \cdot 10^6) = 16,26$. Il valore assoluto dipende dalla normalizzazione: con il fattore $1 / MN$ il massimo dello spettro è la media dei grigi.

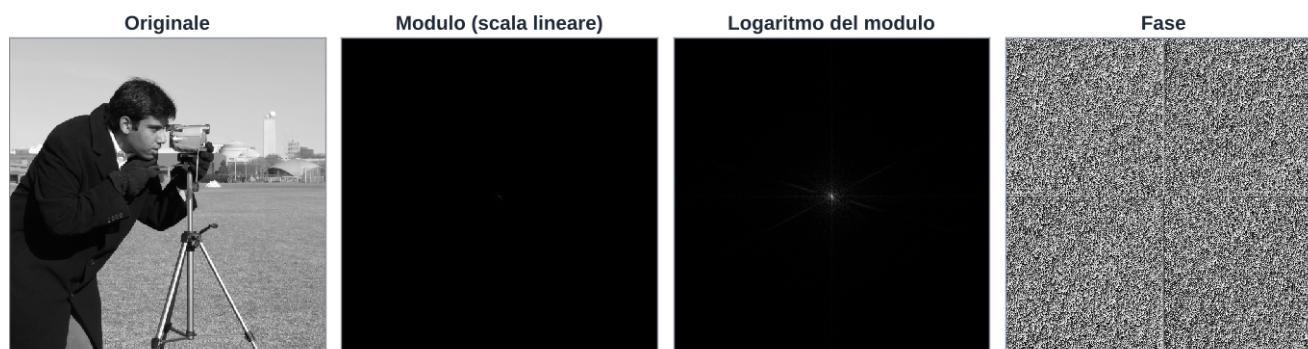


Figura 1. Fotografia, modulo della sua trasformata in scala lineare (si vede solo un punto al centro), logaritmo del modulo e fase. Lo spettro è centrato: la frequenza zero sta al centro del quadro.

Per la fotografia della Figura 1 con il fattore $1 / MN$ il massimo (nel centro) vale 129,1, il valore mediano degli altri è circa 0,014 e $c = 255 / \ln(1 + 129,1) = 52,4$.

Qualche esempio

Un impulso approssimato da un rettangolo di lato 10 e altezza 2 su una finestra di 256 valori ha come trasformata una funzione con un lobo centrale largo e lobi laterali decrescenti, di tipo sinc. Un **segnale stretto nello spazio è largo nelle frequenze**, e viceversa. Nel caso bidimensionale, un piccolo cerchio bianco su fondo nero ha come spettro anelli concentrici di ampiezza decrescente; un quadrato ha lobi lungo gli assi orizzontale e verticale.

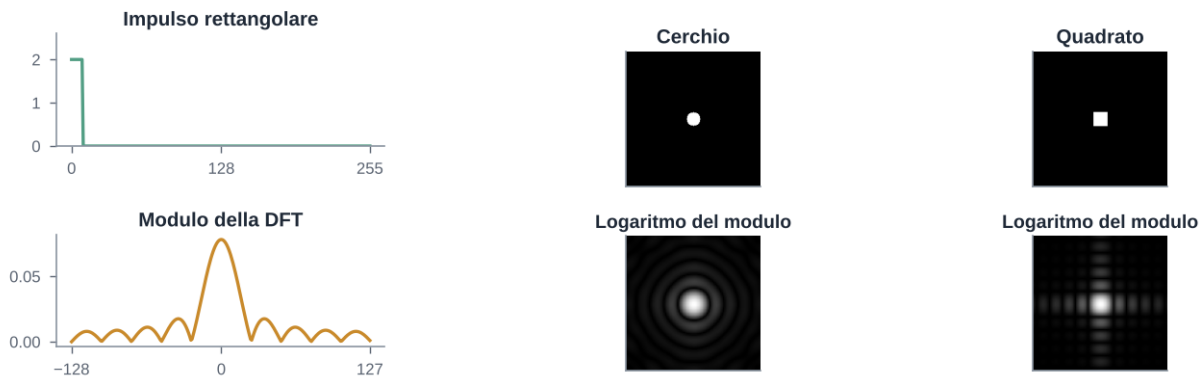


Figura 2. A sinistra un impulso rettangolare di lato 10 e altezza 2 su 256 campioni e il modulo della sua DFT. A destra un cerchio e un quadrato bianchi (lato circa 20 pixel) su fondo nero in una immagine 200×200 , con il logaritmo del modulo della DFT.

MODULO E FASE A CONFRONTO

Per capire il ruolo di modulo e fase si può ricostruire una immagine a partire da una sola delle due componenti: con il solo modulo (e fase nulla) si ottiene una macchia luminosa al centro, senza struttura riconoscibile; con la sola fase (e modulo costante) si riconoscono i contorni dell'immagine. Se poi si scambiano tra due immagini, l'immagine risultante assomiglia a quella da cui proviene la fase.



Figura 3. Ricostruzione dalla sola informazione di modulo (con fase nulla) o di fase (con modulo costante). Solo la fase conserva la struttura riconoscibile dell'immagine.

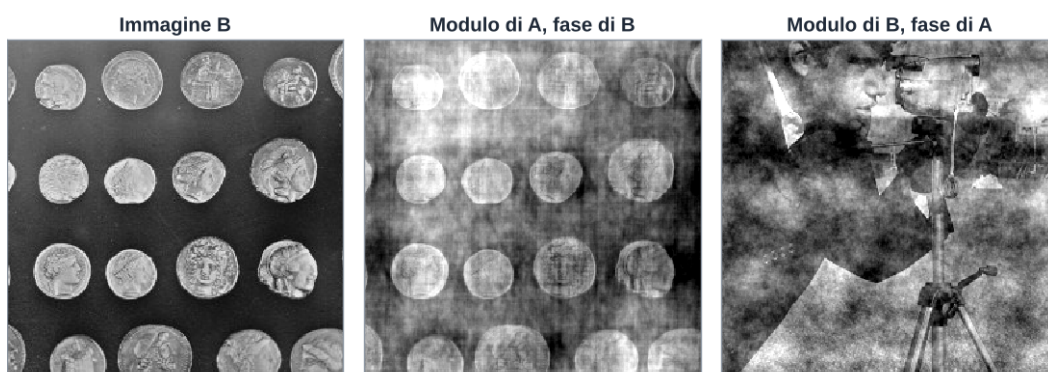


Figura 4. Scambio di modulo e fase tra due immagini: il risultato somiglia all'immagine da cui proviene la fase.

Proprietà della DFT

SEPARABILITÀ

La DFT bidimensionale si può scrivere in forma **separabile**: $F(u, v) = (1/M) \sum_x k(x, v) e^{-i2\pi u x / M}$, con $k(x, v) = (1/N) \sum_y f(x, y) e^{-i2\pi v y / N}$. Il vantaggio è che la $F(u, v)$ si ottiene applicando in **due passi successivi** la trasformata monodimensionale: prima su tutte le righe, poi su tutte le colonne (o viceversa). Sull'esempio di un ritaglio 64×64 della fotografia i due metodi coincidono fino all'arrotondamento macchina (differenza massima $5 \cdot 10^{-13}$).

TRASLAZIONE

Prima di visualizzare o filtrare la trasformata conviene **traslare l'origine** delle frequenze nel punto $(M/2, N/2)$, il centro della matrice dei coefficienti, in modo che $F(0, 0)$ risulti nel centro del rettangolo delle frequenze definito tra $[0, M-1]$ e $[0, N-1]$. Si ottiene scambiando i quattro quadranti in diagonale, come nello schema seguente.



Figura 5. Lo scambio dei quadranti che porta la frequenza zero al centro dello spettro.

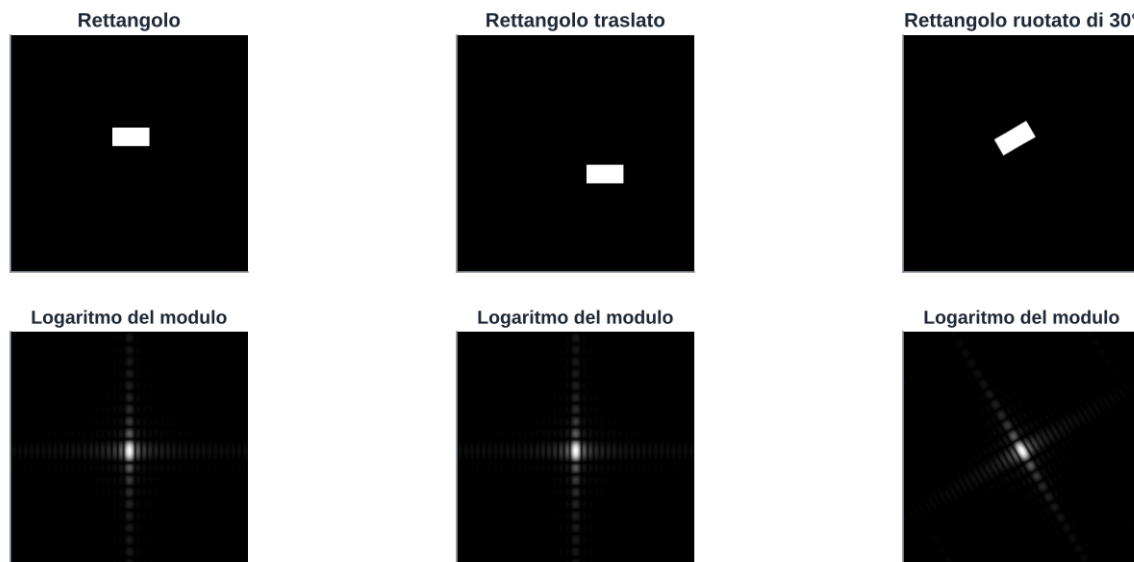


Figura 6. Un rettangolo bianco su fondo nero, lo stesso traslato e lo stesso ruotato, con il logaritmo del modulo della trasformata sotto ciascuno. La traslazione non cambia il modulo; la rotazione ruota lo spettro dello stesso angolo.

Si dimostra che uno **shift nella $f(x, y)$** non modifica la **magnitudo** della trasformata (nell'esempio la differenza massima tra i due moduli è inferiore a 10^{-6} , cioè nulla a meno degli errori numerici, su valori fino a 10^6), ma ne cambia la fase. Al contrario, la trasformata è **sensibile alla rotazione**: ruotare l'immagine ruota lo spettro dello stesso angolo. Questa differenza si vede guardando il modulo; la fase, invece, cambia anche per la traslazione.

VALOR MEDIO

Il valore della trasformata nell'origine, cioè in $(u, v) = (0, 0)$, è $F(0, 0) = (1/MN) \sum \sum f(x, y)$: è la **media** dei valori di grigio dell'immagine, detta anche **componente continua** o componente DC. Per la fotografia usata, $F(0, 0) = 129,06$, uguale alla media calcolata direttamente. Segue che f si può ricostruire anche come $F(0, 0)$ più le oscillazioni attorno alla media.

LA TRASFORMATA VELOCE, FFT

Nella forma classica calcolare la DFT di N campioni richiede un numero di operazioni proporzionale a N^2 : N moltiplicazioni complesse e $N-1$ addizioni per ciascuno degli N valori di u . Con opportune tecniche di decomposizione si scende a $N \log_2 N$: è la **FFT** (*fast Fourier transform*), il cui avvento negli anni Sessanta ha rivoluzionato l'elaborazione digitale dei segnali. Per $N = 1024$ si passa da circa 1 000 000 di operazioni a circa 10 000, cento volte meno.

La convoluzione e il teorema della convoluzione

La **convoluzione** di due funzioni f e h su una griglia di $M \times N$ punti, con la normalizzazione del corso, è

$$g(x, y) = f(x, y) \otimes h(x, y) = (1/MN) \cdot \sum_{m=0 \dots M-1} \sum_{n=0 \dots N-1} f(m, n) \cdot h(x-m, y-n)$$

Per indicarla si usa la notazione $g = f \otimes h$ (in altri testi con un asterisco). Rispetto alla formula del filtraggio del capitolo 22, che sovrappone il kernel all'immagine senza rovesciarlo, qui h compare come $h(x - m, y - n)$, cioè **ruotato di 180 gradi**: per i kernel simmetrici le due operazioni coincidono. Gli indici sono intesi modulo M e N (convoluzione **circolare**): è la topologia toroidale dei bordi del capitolo 22.

La convoluzione è **commutativa**, $f \otimes h = h \otimes f$, e **associativa**, $(f \otimes h) \otimes h_1 = f \otimes (h \otimes h_1)$: applicare in sequenza due filtri è come applicare il filtro ottenuto dalla loro convoluzione.

IL TEOREMA

TEOREMA DELLA CONVOLUZIONE

La trasformata della convoluzione di due segnali nel dominio spaziale equivale al **prodotto delle trasformate** dei due segnali: se $g = f \otimes h$ allora $G(u, v) = F(u, v) \cdot H(u, v)$. Equivalentemente, la convoluzione nel dominio spaziale equivale all'antitrasformata del prodotto delle trasformate: $g = \text{DFT}^{-1}\{F \cdot H\}$.

Il teorema è il fondamento teorico delle tecniche di elaborazione nel dominio della frequenza basate sulla manipolazione della DFT dell'immagine. Con le normalizzazioni del corso ($1 / MN$ sia nella trasformata sia nella convoluzione) vale esattamente $G = F \cdot H$.

Esempio. Si prendono $f = (1, 2, 3, 4)$ e $h = (1, 1, 0, 0)$, con indici circolari modulo 4. La convoluzione (con il fattore $1 / 4$) è $g = (1,25, 0,75, 1,25, 1,75)$: ogni valore è la somma di f nel punto e nel punto precedente, divisa per 4. Le trasformate sono:

u	$F(u)$	$H(u)$	$F(u) \cdot H(u)$	$G(u)$
0	2,50	0,50	1,25	1,25
1	$-0,50 + 0,50 i$	$0,25 - 0,25 i$	$0,25 i$	$0,25 i$
2	$-0,50$	0	0	0
3	$-0,50 - 0,50 i$	$0,25 + 0,25 i$	$-0,25 i$	$-0,25 i$

Tabella 3. Verifica del teorema della convoluzione: la colonna del prodotto $F \cdot H$ coincide con la trasformata G della convoluzione calcolata direttamente.

Su una immagine il risultato è identico: filtrando la fotografia con un kernel 7×7 di media, una volta con la somma dei prodotti (bordi toroidali) e una volta moltiplicando le trasformate, la differenza massima tra i due risultati è $4 \cdot 10^{-13}$ livelli di grigio, cioè solo l'arrotondamento macchina.

QUANDO CONVIENE IL DOMINIO DELLE FREQUENZE

Per un segnale monodimensionale di n campioni la convoluzione diretta ha complessità $O(n^2)$, mentre passando per la FFT si ha $O(n \log n)$: vale la pena passare al dominio delle frequenze. Su una immagine con un kernel $k \times k$ il filtro nel dominio spaziale costa k^2 operazioni per pixel, mentre quello nel dominio delle frequenze costa circa tre FFT (due dirette e una inversa), quindi un numero di operazioni che dipende da $\log(MN)$ e non da k . Per una immagine 512×512 questo vale circa $3 \cdot 18 = 54$ operazioni per pixel, e il dominio spaziale è più economico per k fino a circa 7. In sintesi:

- se il filtro ha dimensioni **confrontabili con quelle dell'immagine** è più efficiente il filtraggio nel dominio delle frequenze;
- con **maschere piccole** è più efficiente il calcolo nel dominio spaziale;
- la definizione di un filtro nel dominio delle frequenze è **più intuitiva**, perché si decide direttamente quali frequenze tenere.

DALLA MASCHERA SPAZIALE AL FILTRO IN FREQUENZA

Per ottenere il filtro H equivalente a una maschera spaziale h : (1) H ha le stesse dimensioni dell'immagine; (2) si pongono i valori della maschera in alto a sinistra, e zero nel resto (con il centro della maschera nell'origine, cioè con gli indici che si avvolgono ai bordi, la fase risulta nulla); (3) si calcola la trasformata di Fourier di questa matrice; (4) si sposta l'origine al centro per visualizzarla.

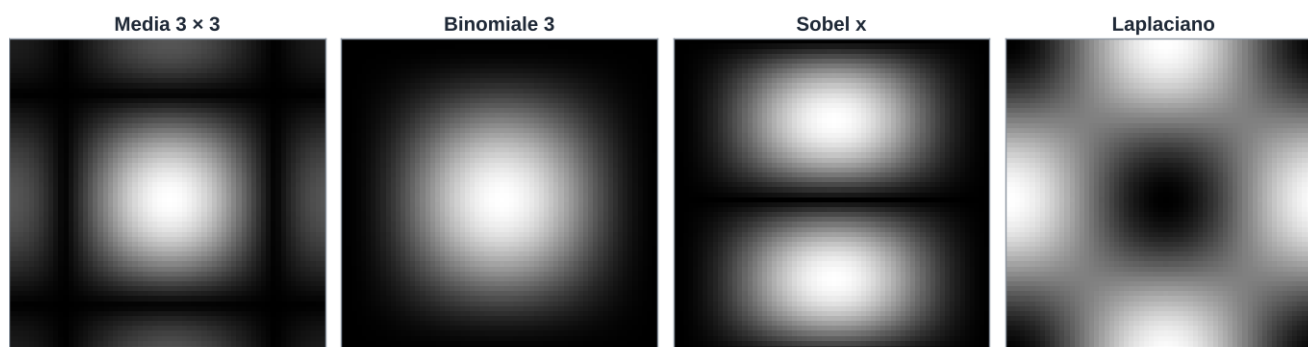


Figura 7. Modulo della trasformata di quattro kernel, riportati su una griglia di 64×64 con zeri. Gli ultimi due (con somma nulla) si annullano al centro e non fanno passare le basse frequenze.

La figura mostra che i kernel di media e binomiale sono filtri **passa basso** (grande al centro, piccolo ai bordi): il binomiale scende in modo più dolce e senza oscillazioni, come atteso dalla sua forma gaussiana. Il Sobel e il Laplaciano, la cui somma dei pesi è zero, valgono zero nel centro e sono di tipo **passa alto** (il Sobel solo lungo una direzione).

Filtraggio nel dominio delle frequenze

Filtrare nel dominio delle frequenze significa moltiplicare la trasformata dell'immagine per una funzione $H(u, v)$, detta **funzione di trasferimento** o filtro, e riportare il risultato nel dominio spaziale:

$$g(x, y) = \text{DFT}^{-1}\{ F(u, v) \cdot H(u, v) \}$$

H agisce su alcune frequenze lasciando le altre immutate. Molto spesso è una funzione **reale**, e ciascuna sua componente moltiplica sia la parte reale sia la parte immaginaria di F : questo tipo di filtri è detto *zero phase shift*, perché non introduce sfasamenti. I passi sono: calcolare F , centrarla, moltiplicare per H , decentrarla e antitrasformare, prendendo la parte reale.

Nel seguito $D(u, v)$ indica la distanza del punto (u, v) dal centro dello spettro, e D_0 la **frequenza di taglio**.

FILTRI PASSA BASSO

Un filtro **passa basso** tiene le basse frequenze e riduce le alte: sfoca. Tre forme classiche:

FILTRO	FUNZIONE DI TRASFERIMENTO $H(u, v)$
Ideale	1 se $D(u, v) \leq D_0$, 0 altrimenti
Butterworth di ordine n	$1 / (1 + (D(u, v) / D_0)^{2n})$
Gaussiano	$e^{-D^2(u, v) / (2 D_0^2)}$

Tabella 4. I tre filtri passa basso. Il Gaussiano ha il vantaggio che la sua trasformata di Fourier è ancora una gaussiana.

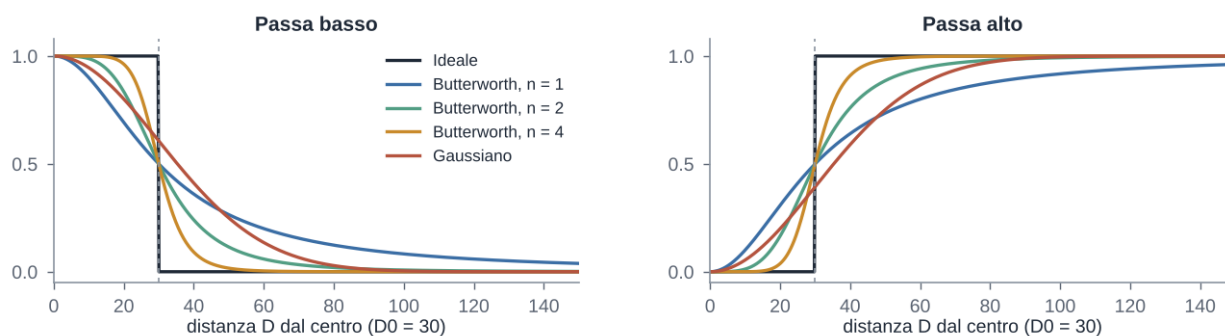


Figura 8. Profili dei filtri passa basso (a sinistra) e dei corrispondenti passa alto (a destra, 1 meno il passa basso), con frequenza di taglio $D_0 = 30$. Il Butterworth ha una transizione più dolce dell'ideale, tanto più ripida quanto maggiore è l'ordine; il Gaussiano scende senza ripidità.

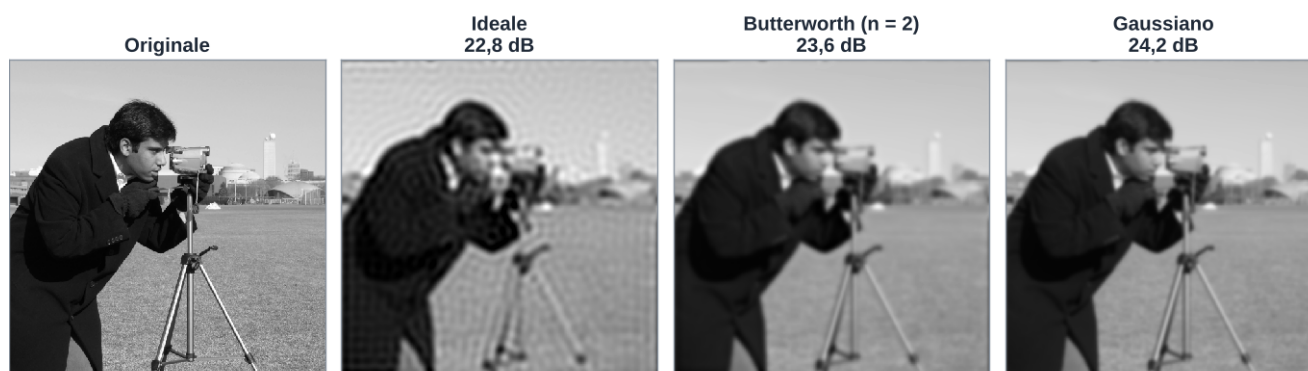


Figura 9. Filtraggio passa basso con $D_0 = 30$ di una fotografia di 512×512 pixel. Il filtro ideale produce onde (ringing) intorno ai contorni, che con Butterworth sono meno marcate e con il Gaussiano scompaiono.

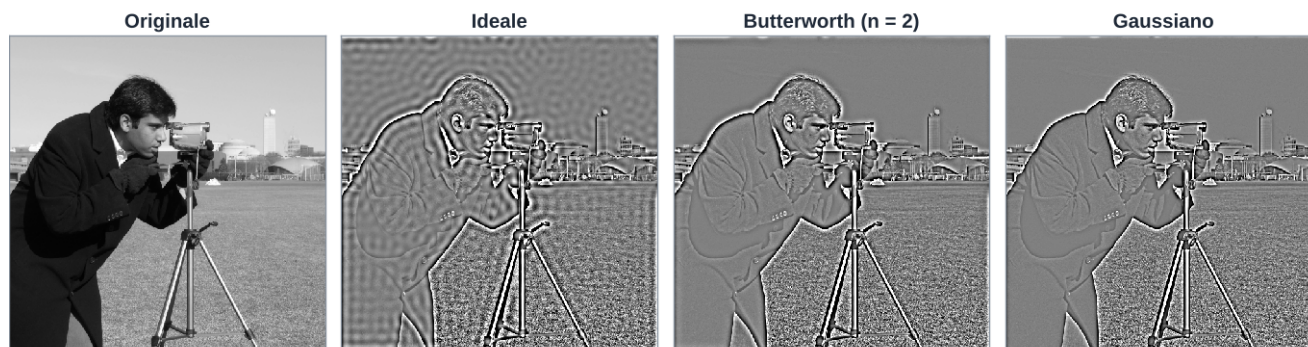


Figura 10. Filtraggio passa alto con $D_0 = 30$: resta l'informazione sui contorni e sui dettagli (grigio 128 = zero, risultato amplificato 3 volte), mentre le zone uniformi sono azzerate.

Il filtro **ideale** taglia di netto tutte le frequenze oltre D_0 , ma questo salto nello spettro provoca nel dominio spaziale delle **onde** intorno ai contorni (*ringing*, fenomeno di Gibbs): la trasformata inversa di un gradino è una funzione di tipo sinc. Il Butterworth riduce il fenomeno, tanto più quanto minore è l'ordine; il Gaussiano non ha ringing perché non ha salti. I PSNR rispetto all'originale sono 22,8 dB (ideale), 23,6 dB (Butterworth) e 24,2 dB (Gaussiano).

FILTRI PASSA ALTO

Un filtro **passa alto** è il complemento del passa basso, $H_{\text{alto}} = 1 - H_{\text{basso}}$: l'ideale vale 0 dentro il raggio D_0 e 1 fuori; allo stesso modo si ottengono il Butterworth e il Gaussiano. Tengono le basse frequenze, quindi la componente continua e le zone uniformi, e lasciano contorni e dettagli.

FILTRI BAND REJECT

Un filtro **band reject** (a eliminazione di banda) elimina le frequenze comprese in una banda attorno a un raggio D_0 di larghezza W e lascia le altre. Nella forma ideale vale 0 se $D_0 - W/2 \leq D(u, v) \leq D_0 + W/2$ e 1 altrimenti; esistono le versioni di Butterworth e Gaussiana, più dolci. È utile per togliere un **disturbo periodico**, che nello spettro appare come punti isolati e molto luminosi.



Figura 11. Un disturbo sinusoidale a strisce oblique (frequenza (60, 40)) appare nello spettro come due punti luminosi simmetrici rispetto al centro. Un filtro band reject ad anello centrato alla loro distanza (72,1, larghezza 10) lo elimina.

Nell'esempio, il disturbo periodico fa scendere il PSNR a 19,4 dB; dopo il band reject il PSNR è 32,0 dB. Il filtro elimina anche parte delle frequenze dell'immagine nell'anello, e per questo il risultato non coincide con l'originale.

DA RICORDARE

La DFT scompone l'immagine in sinusoidi complesse: il modulo dice quanto, la fase dove. $F(0, 0)$ è la media dei grigi. Per il teorema della convoluzione filtrare con un kernel nel dominio spaziale equivale a moltiplicare per la sua trasformata nel dominio delle frequenze: $F \cdot H$. Il filtro passa basso sfoca (il Gaussiano senza ringing), il passa alto lascia i bordi, il band reject elimina una banda, per esempio un disturbo periodico. La FFT costa $N \log N$ invece di N^2 .

Compressione, entropia e teorema di Shannon

Come si riduce il numero di bit necessari a descrivere una immagine: ridondanza, compressione lossless e lossy, entropia, teorema di Shannon e i metodi più semplici, RLE, codifica differenziale e requantization.

Che cos'è la compressione

Con il termine **compressione dati** si indica la tecnica di elaborazione che, attuata a mezzo di opportuni algoritmi, permette di **ridurre la quantità di bit** necessari a rappresentare in forma digitale una informazione. Un file compresso occupa meno spazio in memoria, e una trasmissione di dati compressi occupa meno banda.

Il processo ha sempre due metà. Un **codificatore** (encoder) trasforma i dati originali in dati compressi; un **decodificatore** (decoder) ricostruisce i dati a partire da quelli compressi. Nella compressione senza perdita i dati ricostruiti sono identici agli originali; nella compressione con perdita sono soltanto simili.

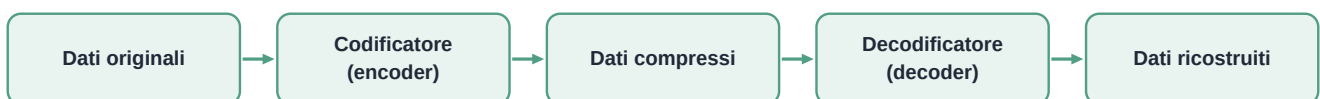


Figura 1. Il processo di compressione e decompressione. I dati ricostruiti coincidono con gli originali se la compressione è lossless, sono una approssimazione se è lossy.

Per misurare l'efficacia di un metodo si usano due numeri. Se n_1 è il numero di bit dei dati originali e n_2 quello dei dati compressi, il **rapporto di compressione** C e la **ridondanza relativa** R dei dati originali sono:

$$C = n_1 / n_2 \quad R = 1 - 1 / C$$

Esempio. Una immagine di 512×512 pixel a 8 bit occupa 2 097 152 bit (262 144 byte). Se dopo la compressione ne occupa 524 288 (65 536 byte), $C = 4$ e $R = 1 - 1/4 = 0,75$: il 75% dei bit originali era ridondante. Con $C = 1$ non si guadagna nulla; se $C < 1$ il metodo allunga i dati invece di accorciarli, cosa che accade davvero con alcuni metodi e alcuni dati (si veda la RLE più avanti).

La ridondanza

Dato che quantità diverse di dati possono rappresentare la stessa quantità di informazione, una rappresentazione che contiene informazioni **irrilevanti** o **ripetute** contiene dati **ridondanti**. Comprimerne significa eliminare la ridondanza. Se ne distinguono tre tipi.

- **Ridondanza di codifica.** Un codice è un sistema di simboli (lettere, numeri, bit) usato per rappresentare informazione; a ogni evento è assegnata una sequenza di simboli, la **codeword**, e il numero di simboli della sequenza è la sua **lunghezza**. Se si usa la stessa lunghezza per eventi frequenti e per eventi rari, si spreca spazio: assegnare codeword corte ai simboli frequenti riduce la lunghezza media (codifica di Huffman, capitolo 28).
- **Ridondanza spaziale e temporale.** In una immagine ogni pixel è simile ai pixel del suo intorno, o ne dipende, quindi l'informazione è replicata inutilmente nei pixel correlati. In una sequenza video lo stesso vale per i pixel di fotogrammi vicini.
- **Informazione percettivamente irrilevante.** Spesso i dati contengono informazione che il sistema visivo (o uditivo) umano ignora. È ridondante nel senso che non viene utilizzata.

La ridondanza spaziale si vede bene su una fotografia reale: il coefficiente di correlazione tra un pixel e quello alla sua destra è 0,978. Il primo grafico della figura 2 mostra che quasi tutte le coppie stanno vicino alla diagonale: noto un pixel, il suo vicino è prevedibile. Questa prevedibilità si traduce in una distribuzione delle *differenze* molto concentrata intorno allo zero, con entropia minore di quella dei valori (la nozione di entropia è nella sezione seguente).

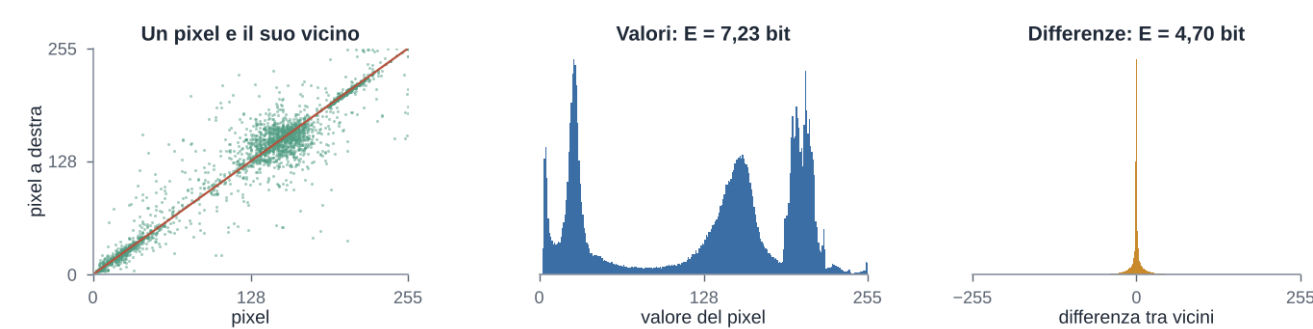


Figura 2. Ridondanza spaziale nella fotografia 512×512 . Sinistra: il valore di un pixel (asse orizzontale) contro quello del pixel a destra (verticale), per 4000 coppie casuali; la retta è la diagonale. Centro e destra: istogramma dei valori e istogramma delle differenze tra pixel adiacenti di una riga, con la rispettiva entropia.

Compressione lossless e lossy

I metodi di compressione si classificano in base al tipo di dati a cui si applicano (generici, audio, immagini, video) e, soprattutto, in base al tipo di compressione:

	LOSSLESS (REVERSIBILE)	LOSSY (IRREVERSIBILE)
Perdita di informazione	Nessuna: i dati si ricostruiscono esattamente, bit per bit.	Sì: i dati ricostruiti sono approssimati.
Risparmio di memoria	Moderato, limitato dall'entropia.	Molto maggiore.
Quando si usa	Documenti, programmi eseguibili, immagini mediche o scientifiche.	Fotografie, musica, video destinati alla percezione umana.
Esempi	RLE, codifica differenziale, Huffman, ZIP, PNG.	Requantization, JPEG, MP3, MPEG.
Criterio di bontà	Avvicinarsi al limite teorico di Shannon.	Minima distorsione per un dato numero di bit, o minimo numero di bit per una data distorsione.

Tabella 1. Compressione lossless e lossy a confronto.

Nella compressione **lossless** i dati si trasformano per occupare meno memoria e poi si ricostruiscono perfettamente, senza errore e senza perdere alcun bit. È ovviamente indispensabile per documenti ed eseguibili: un programma ricostruito con un bit sbagliato non funziona. Un buon metodo lossless cerca di raggiungere il limite teorico fornito dal primo teorema di Shannon, descritto tra poco.

Nella compressione **lossy** si accetta di perdere informazione in cambio di un risparmio molto maggiore. Una volta scartata, l'informazione non si può ricostruire: la compressione lossy è **irreversibile**.

Frequenza ed entropia

Sia data una sequenza S di N caratteri tratti da un alfabeto di M possibili simboli $a_1, \dots, a_M$. La **frequenza** del simbolo a_i è la frazione di volte in cui compare:

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

L'**entropia** E della sequenza è la quantità media di informazione associata alla generazione di un simbolo. Si misura in bit per simbolo:

$$E = - \sum f_i \cdot \log_2(f_i) \quad (\text{somma sui simboli presenti})$$

Un simbolo di frequenza f porta una informazione di $-\log_2(f)$ bit: un evento che accade metà delle volte vale 1 bit, uno che accade una volta su otto ne vale 3. L'entropia è la media di queste quantità pesata con le frequenze. Più è grande l'incertezza della sequenza, maggiore è l'entropia; il **massimo** si ha quando i simboli sono **equiprobabili**, e vale $\log_2(M)$. Il minimo, 0, si ha quando un solo simbolo ha frequenza 1.

SE LA CALCOLATRICE NON HA IL LOGARITMO IN BASE 2

Si cambia base: $\log_2(x) = \log_{10}(x) / \log_{10}(2) = \ln(x) / \ln(2)$. Per esempio $\log_2(1/8) = -\log_{10}(8) / \log_{10}(2) = -0,9031 / 0,3010 = -3$.

UN ESEMPIO COMPLETO

Si calcola l'entropia della sequenza **AABABCAACAAADDDD**, di $N = 16$ caratteri.

Simbolo	Occorrenze	Frazione	f	$-\log_2 f$	$-f \log_2 f$
A	8	8/16	0,500	1	0,500
B	2	2/16	0,125	3	0,375
C	2	2/16	0,125	3	0,375
D	4	4/16	0,250	2	0,500
Totale	16	1			1,75

Tabella 2. Calcolo dell'entropia di AABABCAACAAADDDD: $E = 0,5 + 0,375 + 0,375 + 0,5 = 1,75$ bit per simbolo.

L'entropia vale 1,75 bit per simbolo, contro i 2 bit che servirebbero con un codice a lunghezza fissa per 4 simboli. La figura 3 mostra l'andamento dell'entropia: a sinistra per una sorgente con due soli simboli (massima, 1 bit, quando $p = 0,5$; nulla quando uno dei due è certo), a destra per quattro simboli con distribuzioni via via più sbilanciate.

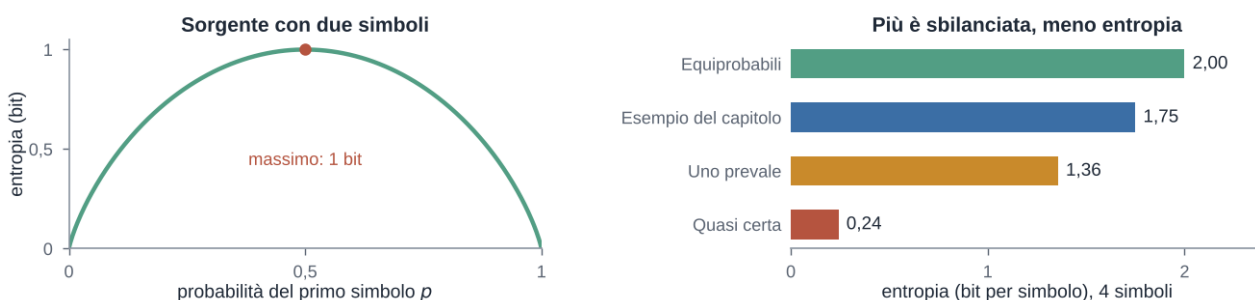


Figura 3. L'entropia come misura dell'incertezza. A sinistra: sorgente binaria in funzione di p . A destra: quattro simboli, da equiprobabili (entropia 2 bit, il massimo) a quasi certi (0,24 bit).

Lo stesso calcolo si applica a una immagine, con i livelli di grigio al posto dei caratteri e l'istogramma normalizzato (capitolo 19) al posto delle frequenze. Per la fotografia 512×512 usata in questi capitoli, a 8 bit per pixel, l'entropia dell'istogramma è 7,23 bit per pixel: la distribuzione dei grigi non è piatta, quindi il limite teorico per una codifica che consideri ogni pixel da solo è inferiore agli 8 bit di partenza.

Il teorema di Shannon

Il **primo teorema di Shannon** (Claude Shannon, 1948) afferma che per una sorgente discreta e **senza memoria**, cioè che emette simboli indipendenti l'uno dall'altro, il bitrate minimo è pari all'entropia della sorgente. I dati si possono rappresentare senza perdere informazione usando almeno un numero di bit pari a:

$$N \cdot E$$

dove N è il numero di caratteri ed E l'entropia. Per la sequenza dell'esempio: $16 \times 1,75 = \mathbf{28 \text{ bit}}$. Con un codice a lunghezza fissa (2 bit per ciascuno dei 4 simboli) servirebbero $16 \times 2 = 32$ bit: il limite di Shannon permette di risparmiare 4 bit, il 12,5%.

ATTENZIONE

Il teorema fissa il numero **minimo** di bit, ma non dice come trovare il codice che lo raggiunge. Serve un algoritmo che codifichi i caratteri usando (quasi) esattamente quel numero di bit: uno di questi è quello di Huffman, nel capitolo 28.

Due precisazioni rendono l'enunciato più esatto.

- Un codice a **lunghezza intera** per simbolo, come quello di Huffman, non può in generale raggiungere esattamente l'entropia, perché le lunghezze delle codeword sono numeri interi di bit. Si dimostra che la lunghezza media L per simbolo soddisfa $E \leq L < E + 1$, cioè ci si avvicina al limite con un eccesso inferiore a un bit per simbolo. Nell'esempio dei capitoli seguenti tutte le frequenze sono potenze di $1/2$ e il limite è raggiunto esattamente.
- Se la sorgente **ha memoria**, cioè i simboli dipendono dai precedenti, come i pixel vicini di una immagine, l'entropia dei singoli simboli non è il limite: il limite è la minore *entropia per simbolo del blocco*, che sfrutta le dipendenze. È il motivo per cui la codifica differenziale della sezione seguente può comprimere più di una codifica pixel per pixel.

La codifica a lunghezza di sequenza (RLE)

La **Run Length Encoding** sfrutta le ripetizioni di uno stesso valore lungo le righe (o le colonne) di una immagine. Una sequenza di valori uguali, detta **run**, si rappresenta con la sola lunghezza, o con la coppia (valore, lunghezza). È usata in CCITT, JBIG2, JPEG e BMP.

Esempio. Si vuole comprimere la sequenza di 29 bit **00000111001011101110101111111**. Si può dire «5 volte 0, 3 volte 1, 2 volte 0, ...». Meglio ancora ci si accorda che la sequenza inizia sempre con 0 e si memorizzano soltanto le lunghezze delle run: **5, 3, 2, 1, 1, 3, 1, 3, 1, 1, 7**. Questi numeri si scrivono in binario, con 3 bit ciascuno (il massimo è 7): 101 011 010 001 001 011 001 011 001 001 111.

Servono $12 \times 3 = \mathbf{36 \text{ bit}}$, più dei 29 di partenza.

QUANDO LA RLE CONVIENE

La RLE porta un risparmio solo se le run sono **lunghe**: ogni run costa un numero fisso di bit (quello per scrivere la lunghezza massima), quindi le run di 1 o 2 pixel costano più dei pixel stessi. Nell'esempio le run sono 12 su 29 bit, in media meno di 2,5 bit ciascuna, e il risultato peggiora.

RLE E PIANI DI BIT

Un uso sensato della RLE nelle immagini si ottiene dopo avere separato i **piani di bit** (capitolo 21): i piani più significativi hanno macchie estese e uniformi, quindi molte run lunghe. Si prende ad esempio l'immagine a 8 livelli (3 bit per pixel) delle slide, di 4×4 pixel, con i valori della tabella 3.

Piano	Bit, righe di seguito	Lunghezze delle run	Bit RLE
Piano 2 (MSB)	0110111111011100	1, 2, 1, 6, 1, 3, 2	21
Piano 1	1010000000100010	0, 1, 1, 1, 7, 1, 3, 1, 1	27
Piano 0 (LSB)	0110111100101000	1, 2, 1, 4, 2, 1, 1, 1, 3	27

Tabella 3. I tre piani di bit dell'immagine 4×4 [2 5 7 0 / 5 5 5 5 / 4 4 3 4 / 5 4 2 0] in ordine di riga, e la loro RLE a 3 bit per run. Una lunghezza 0 iniziale significa che la sequenza comincia con 1.

In totale servono 75 bit contro i 48 originali ($16 \text{ pixel} \times 3 \text{ bit}$): su una immagine così piccola la RLE non conviene. L'effetto cambia con immagini vere. La tabella 4 la applica ai piani della fotografia 512×512 , con run di lunghezza al più 512, quindi 10 bit per ciascuna run; ogni piano occupa 262 144 bit.

Piano	Run (righe)	Bit RLE	Rapporto 262 144 / bit
7	11 646	116 460	2,25
6	17 440	174 400	1,50
5	44 751	447 510	0,59
4	59 383	593 830	0,44
3	73 884	738 840	0,35
2	91 411	914 110	0,29
1	108 876	1 088 760	0,24
0	124 685	1 246 850	0,21

Tabella 4. RLE applicata riga per riga ai piani di bit della fotografia, con 10 bit per run. Solo i due piani più significativi si comprimono (rapporto maggiore di 1); dal piano 5 in giù la RLE allunga i dati.

I piani 7 e 6, che descrivono la struttura generale, si comprimono (rapporti 2,25 e 1,5). Già dal piano 5 il rapporto scende sotto 1, e i piani bassi, che contengono rumore, hanno una run ogni pochi pixel e sarebbero allungati di oltre quattro volte. Nella pratica, dunque, la RLE è un componente di schemi più grandi (JPEG la usa per le lunghe sequenze di zeri, capitolo 29), non un metodo universale.

La codifica differenziale

È una tecnica molto usata. Se i valori di una sequenza variano lentamente, invece di registrarli si memorizzano il **valore iniziale** e le **differenze successive**. Si ricostruisce poi la sequenza sommando via via le differenze.

Esempio. La sequenza 134, 137, 135, 128, 130, 134, 112 si memorizza come valore iniziale 134 seguito dalle differenze +3, -2, -7, +2, +4, -22.

CORREZIONE

Nelle slide le differenze sono riportate come «-3, +2, 7, -2, -4, 22»: con il segno opposto a quello usuale, perché calcolate come valore precedente meno valore successivo ($134 - 137 = -3$), e con il segno + mancante sul 7. In questa dispensa si usa la convenzione più comune, valore successivo meno precedente ($137 - 134 = +3$), con cui la ricostruzione è una semplice somma: $134 + 3 = 137$, $137 - 2 = 135$, e così via.

Perché conviene? Si dimostra sperimentalmente che per le immagini la sequenza delle differenze ha una **entropia minore** di quella dei valori originali, quindi richiede meno bit. Per la fotografia 512×512 la differenza tra pixel adiacenti sulla stessa riga ha entropia 4,70 bit contro 7,23 dei valori (figura 2): se il primo pixel di ogni riga è conservato così com'è, il rapporto di compressione teorico passa da $8 / 7,23 = 1,11$ a $8 / 4,70 = 1,70$. Ricostruendo le righe con una somma cumulativa si ottiene di nuovo l'immagine originale, identica pixel per pixel.

Le differenze hanno però un intervallo più ampio dei valori (da -255 a 255, quindi 9 bit con una codifica a lunghezza fissa): il vantaggio si ottiene solo associando alla codifica differenziale una codifica di entropia, come Huffman, che dà codeword corte ai valori vicini allo zero, i più frequenti. Per questo la codifica differenziale è un passo preparatorio e non un metodo completo. JPEG la usa per i coefficienti DC dei blocchi (capitolo 29).

La compressione lossy

Quando si accetta di perdere informazione si può guadagnare molto di più, sfruttando il fatto che gli esseri umani interpretano correttamente segnali rumorosi o incompleti. Un esempio classico è la lingua scritta senza vocali:

Nl mzz dl cmmn d nstr vt / M rtrv pr n slv scr

Chi conosce la lingua riconosce senza sforzo i versi di Dante («Nel mezzo del cammin di nostra vita / mi ritrovai per una selva oscura»). Alcune lingue non prevedono nemmeno di scrivere le vocali. Se si ha una competenza, spesso si completano da soli i dettagli non trasmessi: è un esempio di **ridondanza semantica**. Alcuni telefoni, componendo un SMS, scelgono la sequenza di caratteri più probabile nella lingua: è un esempio di **ridondanza statistica**.

Il criterio per giudicare un metodo lossy è duplice. Fissata la massima **distorsione** accettabile, l'algoritmo deve trovare la rappresentazione con il più basso numero di bit. Viceversa, fissato il massimo numero di bit accettabile, occorre trovare l'algoritmo che a parità di bit dà la minima distorsione. La distorsione si misura con il PSNR (capitolo 13) oppure, meglio, con il giudizio dell'osservatore.

L'idea di fondo è: **se percettivamente non è importante, si butta via**. L'MP3 la applica al suono e alla musica, il JPEG alle immagini fisse, MPEG e simili alle sequenze di immagini. Nel corso si studiano soltanto due esempi: un algoritmo di requantization e lo standard JPEG (capitolo 29).

REQUANTIZATION

È il modo più semplice per comprimere con perdita: si **riduce il numero di livelli** disponibili, e quindi i bit per pixel, «dimenticando» gli n bit meno significativi di ogni canale. È la quantizzazione uniforme del capitolo 9. Con k bit conservati su 8, i livelli diventano 2^k , il rapporto di compressione è $8 / k$ e per la ricostruzione si usa il centro di ciascun intervallo.



Figura 4. La fotografia a 8 bit e dopo avere conservato soltanto i 4, 2 e 1 bit più significativi. Con 4 bit il difetto è lieve; con 2 e 1 bit compaiono bande di grigio uniforme e i dettagli sfumati scompaiono.

Bit k conservati	Livelli	Rapporto $8 / k$	PSNR (dB)	Entropia dell'immagine a k bit
1	2	8,00	17,2	0,94
2	4	4,00	23,6	1,82
3	8	2,67	28,7	2,51
4	16	2,00	35,0	3,39
5	32	1,60	40,8	4,31
6	64	1,33	46,3	5,26
7	128	1,14	51,1	6,24
8	256	1,00	∞	7,23

Tabella 5. Requantization della fotografia 512×512 : conservando i k bit più significativi, il rapporto di compressione è $8 / k$, ma il PSNR scende rapidamente. L'ultima colonna mostra che con meno livelli l'entropia, e quindi lo spazio dopo una codifica di entropia, scende ancora.

Con 4 bit (rapporto 2) il PSNR è circa 35 dB e la perdita è appena visibile; con 2 bit (rapporto 4) il PSNR scende a 23,6 dB e i grigi diventano bande. L'ultima colonna della tabella spiega perché la requantization si abbina bene a metodi lossless: «se ci sono meno simboli, la compressione LZW o Huffman è più efficiente».

Esempio. Le slide propongono di conservare, di un pixel a 24 bit, 4 bit del rosso, 6 del verde e 2 del blu. Dato che l'occhio è più sensibile al verde, ha senso dare al verde più livelli: i bit scendono da 24 a 12, un risparmio del 50%. Il costo è la qualità, come mostra la figura 5.

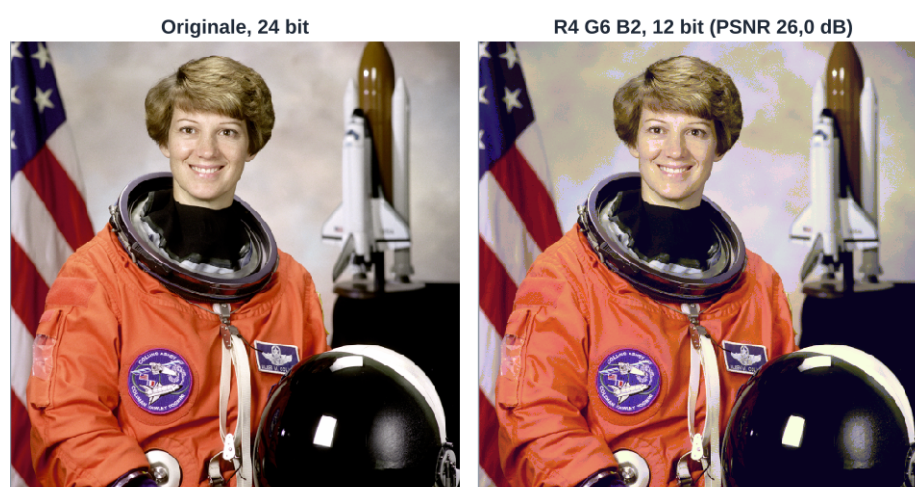


Figura 5. L'esempio delle slide applicato a una immagine a colori: rosso a 4 bit, verde a 6 bit, blu a 2 bit. Il numero di bit per pixel si dimezza (da 24 a 12) ma il blu, ridotto a soli 4 livelli, produce evidenti bande e dominanti di colore.

Il PSNR calcolato sui tre canali è 26,0 dB. Il blu, con soli $2^2 = 4$ livelli, è il canale che si degrada di più.

DA RICORDARE

Comprimere vuol dire eliminare la ridondanza: di codifica (codeword più corte per simboli più frequenti), spaziale e temporale (pixel simili ai vicini) e percettiva (dettagli che non si vedono). L'entropia $E = -\sum f \log_2 f$ dà il minimo di bit per simbolo; il teorema di Shannon dice che servono almeno $N \cdot E$ bit senza perdita, ma non come codificare. RLE e codifica differenziale preparano i dati; Huffman (capitolo 28) li codifica; JPEG (capitolo 29) mette insieme tutto, con una perdita controllata.

La codifica di Huffman

L'algoritmo che costruisce un codice a lunghezza variabile quasi ottimale: idea, proprietà, costruzione dell'albero, esempi svolti, confronto con il limite di Shannon e uso pratico.

Idea e proprietà

Il teorema di Shannon (capitolo 27) dice quanti bit servono al minimo, ma non come ottenerli. L'**algoritmo di Huffman**, proposto da David Huffman nel 1953, è un semplice algoritmo **greedy** («goloso»: a ogni passo fa la scelta localmente migliore) che costruisce un «dizionario», cioè una tabella simbolo → codeword binaria, per una compressione **quasi ottimale**: pari al limite di Shannon con un eccesso di al più qualche bit.

Le proprietà fondamentali sono tre.

- È una codifica a **lunghezza variabile**: assegna codeword corte ai simboli frequenti e lunghe ai simboli rari.
- È una codifica **prefissa**: nessuna codeword è il prefisso di un'altra. Ne segue che la sequenza codificata si decodifica senza ambiguità e senza separatori.
- È **ottimale** tra i codici che assegnano una codeword intera a ciascun simbolo: nessun altro codice di questo tipo ha una lunghezza media minore.

PERCHÉ IL PREFISSO CONTA

Se $A = 0$ e $B = 01$, la sequenza 001 si può leggere come A A seguita da un bit pendente, oppure come A B: il decodificatore non sa decidere. Con $A = 0$ e $B = 10$ nessuna codeword inizia con un'altra, e ogni sequenza ha una sola lettura. Un codice prefisso si rappresenta con un albero binario in cui i simboli sono solo foglie.

Il confronto con la lunghezza fissa

La codifica a **lunghezza fissa** usa lo stesso numero di bit per ogni simbolo. È semplice e veloce da implementare ma spreca spazio, perché non tiene conto delle frequenze. La codifica di Huffman è in generale migliore, ma richiede di conoscere in anticipo le frequenze (o di calcolarle con una prima lettura dei dati). Non esiste una scelta sempre migliore: dipende dalla situazione.

Esempio: la stringa «pelliccia». I simboli distinti sono 6 (p, e, l, i, c, a). Con una codifica a lunghezza fissa il numero di bit per simbolo è il minimo n tale che $2^n \geq 6$, cioè $n = 3$. Si numerano i simboli nell'ordine in cui compaiono e si scrive il numero in binario a 3 cifre:

Simbolo	p	e	l	l	i	c	c	i	a
Numero	0	1	2	2	3	4	4	3	5
Codice a 3 bit	000	001	010	010	011	100	100	011	101

Tabella 1. La stringa «pelliccia» con una codifica a lunghezza fissa di 3 bit per simbolo.

Il totale è 9 simboli $\times$ 3 bit = **27 bit**. Più avanti si vedrà che con Huffman bastano 23 bit.

L'algoritmo

Si parte da una **foresta** di alberi, uno per simbolo, ciascuno costituito da un solo nodo (la foglia) con peso uguale alla frequenza del simbolo (si possono usare indifferentemente le frequenze relative, come $1/8$, o i conteggi, come 2). L'algoritmo è iterativo:

1. si scelgono i due alberi della foresta con **peso minore** e li si collega con un nuovo nodo padre, il cui peso è la somma dei pesi dei due;
2. se più nodi hanno il peso minimo se ne scelgono due qualsiasi; se il minimo è unico si prende quel nodo e poi uno qualsiasi tra quelli con il secondo peso più basso;
3. si ripete finché rimane un solo albero: la radice ha peso 1 (o N , usando i conteggi);
4. si etichettano i rami: per convenzione, 1 per il ramo sinistro e 0 per il destro;
5. la codeword di un simbolo è la sequenza delle etichette lette dalla radice alla sua foglia.

A parità di pesi la scelta è libera: scelte diverse producono alberi e codeword diversi ma con la **stessa lunghezza totale**, perché tutti sono ottimali.

Esempio dalle slide: AABABCAACAAADDDD

La sequenza ha 16 caratteri. Le frequenze sono A: $8/16 = 1/2$, B: $2/16 = 1/8$, C: $2/16 = 1/8$ e D: $4/16 = 1/4$. Nella figura 1 si usano i conteggi (8, 2, 2, 4) al posto delle frequenze.



Figura 1. Costruzione dell'albero. B e C (peso 2 ciascuno) sono i più leggeri e si fondono in un nodo di peso 4; poi (BC) e D, entrambi di peso 4, si fondono in un nodo di peso 8; infine A (8) e (BCD) (8) formano la radice, di peso 16. I rami sinistri sono etichettati con 1, i destri con 0.

Leggendo le etichette dalla radice a ogni foglia si ottiene il dizionario della tabella 2. Il simbolo più frequente, A, ha la codeword più corta (1 bit); i meno frequenti, B e C, la più lunga (3 bit). Nessuna codeword è prefisso di un'altra.

Simbolo	Occorrenze	Frazione	f	Codeword	Lunghezza	Bit totali
A	8	$8/16$	0,500	1	1	8
D	4	$4/16$	0,250	00	2	8
B	2	$2/16$	0,125	011	3	6
C	2	$2/16$	0,125	010	3	6
Totale	16	1				28

Tabella 2. Il dizionario di Huffman per AABABCAACAAADDDD.

La stringa codificata, scrivendo le codeword una dopo l'altra (A A B A B C ...), è:

1 1 011 1 011 010 1 1 010 1 1 1 00 00 00 00 → 1101110110101101011100000000

Gli spazi sono del tutto superflui: dato che nessuna codeword è prefisso di un'altra, la sequenza di 28 bit si decodifica senza errori anche senza separatori. Il decodificatore parte dalla radice, segue un ramo per ogni bit letto

e, quando arriva a una foglia, emette il simbolo e riparte dalla radice. Per esempio, nella sequenza 1 1 0 1 1 1 0 1 1 ... il primo 1 porta subito alla foglia A, il secondo ancora ad A, poi 0 1 1 conduce a B, e così via.

Verifica con Shannon. Il limite è $16 \times (1/2 \cdot 1 + 1/8 \cdot 3 + 1/8 \cdot 3 + 1/4 \cdot 2) = 16 \times (0,5 + 0,375 + 0,375 + 0,5) = 16 \times 1,75 = 28$ bit. La codifica di Huffman usa esattamente 28 bit: è **ottimale**. Si ricorda che con 4 simboli e lunghezza fissa i bit sarebbero $16 \times 2 = 32$. Il risultato è esatto perché tutte le frequenze sono potenze di $1/2$: la lunghezza ideale $-\log_2 f$ di ogni simbolo è un numero intero (1, 3, 3, 2) e coincide con la lunghezza della codeword.

Esempio con la stringa «pelliccia»

Le occorrenze sono l: 2, i: 2, c: 2, p: 1, e: 1, a: 1, per un totale di 9 simboli. I tre simboli con una sola occorrenza sono i più leggeri. Si procede così:

Passo	Nodi fusi (peso)	Peso del nuovo nodo	Nome
1	a (1) + e (1)	2	(ea)
2	p (1) + c (2)	3	(cp)
3	i (2) + l (2)	4	(li)
4	(ea) (2) + (cp) (3)	5	((cp)(ea))
5	(li) (4) + ((cp)(ea)) (5)	9	(((cp)(ea))(li))

Tabella 3. I passi di Huffman per «pelliccia». Nei casi di parità si prende il nodo creato prima.

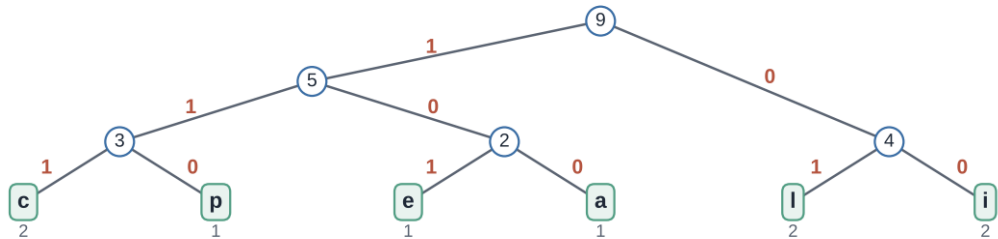


Figura 2. L'albero di Huffman per «pelliccia». Nelle foglie, il simbolo e il numero di occorrenze; nei cerchi, il peso del nodo.

Simbolo	Occorrenze	Frazione	f	Codeword	Lunghezza	Bit totali
i	2	2/9	0,222	00	2	4
l	2	2/9	0,222	01	2	4
a	1	1/9	0,111	100	3	3
c	2	2/9	0,222	111	3	6
e	1	1/9	0,111	101	3	3
p	1	1/9	0,111	110	3	3
Totale	9	1				23

Tabella 4. Il dizionario di Huffman per «pelliccia».

La stringa si codifica come 11010101010011111100100: **23 bit**, contro i 27 della lunghezza fissa. Il limite di Shannon è $9 \times 2,503 = 22,53$ bit; dato che non si possono usare frazioni di bit, il minimo realizzabile è 23, e Huffman lo raggiunge.

ATTENZIONE AI PESI UGUALI

Con pesi uguali si possono ottenere alberi diversi, e quindi codeword diverse (anche di lunghezza diversa per gli stessi simboli), ma il totale è sempre 23 bit. Il risultato di un esercizio va dunque confrontato con il totale dei bit, non con le singole codeword.

Quanto si avvicina al limite

Se L è la lunghezza media delle codeword in bit per simbolo ed E l'entropia, per la codifica di Huffman vale:

$$E \leq L < E + 1$$

Il codice non può essere migliore dell'entropia, ed è peggiore di meno di un bit per simbolo. Il caso peggiore è quello di una sorgente molto sbilanciata. Con due simboli di probabilità 0,9 e 0,1 l'entropia è 0,469 bit, ma ogni codeword deve avere almeno 1 bit: $L = 1$, più del doppio del limite. Il rimedio è codificare **blocchi** di più simboli alla volta: le probabilità dei blocchi sono più varie e la perdita per simbolo scende.

Simboli per blocco	Blocchi possibili	L per blocco (bit)	L per simbolo (bit)	Eccesso su $E = 0,469$
1	2	1,000	1,000	0,531
2	4	1,290	0,645	0,176
3	8	1,598	0,533	0,064

Tabella 5. Sorgente con probabilità 0,9 e 0,1 senza memoria. Codificando blocchi di 1, 2 e 3 simboli la lunghezza media per simbolo scende da 1 verso l'entropia.

Per questo gli standard reali non applicano Huffman ai singoli simboli grezzi, ma a simboli già preparati da altri passi (come i coefficienti quantizzati di JPEG, capitolo 29) o a coppie di valori.

SU UNA IMMAGINE

Applicato ai livelli di grigio della fotografia 512×512 (capitolo 27), Huffman dà una lunghezza media di 7,26 bit per pixel contro un'entropia di 7,23 e gli 8 bit della lunghezza fissa: il rapporto di compressione è 1,10. Applicato alle differenze tra pixel adiacenti, la lunghezza media è 4,72 bit per pixel contro un'entropia di 4,70: il rapporto sale a 1,69. Come previsto da $E \leq L < E + 1$, in entrambi i casi l'eccesso è minimo (0,030 e 0,018 bit). Si noti che i dati reali hanno memoria: la codifica differenziale ne sfrutta una parte e permette di scendere sotto l'entropia dei singoli valori.

Esercizio svolto: «gazzella»

Applicare la codifica di Huffman alla stringa «gazzella». Quanti bit sono effettivamente usati? Quanti bit si dovrebbero usare secondo il teorema di Shannon?

Occorrenze: a: 2, z: 2, l: 2, g: 1, e: 1, su 8 simboli. Con lunghezza fissa servirebbero 3 bit per ciascuno dei 5 simboli ($2^3 \geq 5$), quindi $8 \times 3 = 24$ bit. Con Huffman:

I passi sono quattro: $e + g \rightarrow (ge)$ (peso 2); $a + l \rightarrow (la)$ (peso 4); $z + (ge) \rightarrow ((ge)z)$ (peso 4); $(la) + ((ge)z) \rightarrow (((ge)z)(la))$ (peso 8). Il risultato è l'albero della figura 3.

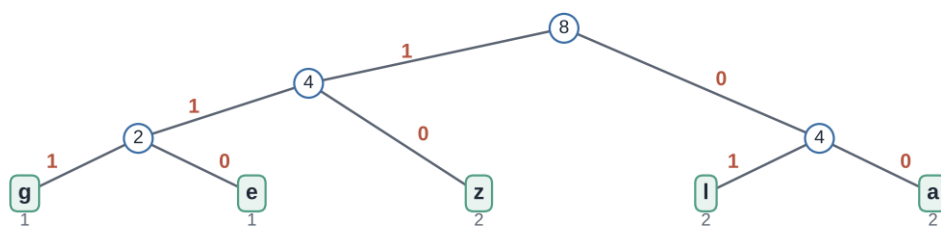


Figura 3. L'albero di Huffman per «gazzella».

Simbolo	Occorrenze	Frazione	f	Codeword	Lunghezza	Bit totali
a	2	2/8	0,250	00	2	4
l	2	2/8	0,250	01	2	4
z	2	2/8	0,250	10	2	4
e	1	1/8	0,125	110	3	3
g	1	1/8	0,125	111	3	3
Totale	8	1				18

Tabella 6. Il dizionario di Huffman per «gazzella».

La stringa si codifica come 111001010110010100, cioè **18 bit**. Secondo Shannon il minimo è $8 \times E$, con $E = 3 \cdot (2/8 \cdot 2) + 2 \cdot (1/8 \cdot 3) = 1,5 + 0,75 = 2,25$ bit per simbolo, cioè $8 \times 2,25 = 18$ bit. Anche qui Huffman raggiunge esattamente il limite, perché le frequenze ($1/4$ e $1/8$) sono potenze di $1/2$.

Huffman in pratica

- **Costo della tabella.** Per decodificare serve il dizionario simbolo $\rightarrow$ codeword, che va memorizzato insieme ai dati. Se i simboli sono pochi il costo è trascurabile; se sono molti può essere rilevante, e su dati molto brevi può annullare il risparmio. Per gli esempi di questo capitolo la tabella non è stata contata.
- **Conoscere le frequenze.** Servono in partenza: o si leggono i dati una prima volta per contarle, o si usano tabelle standard predefinite (come in JPEG) o si adotta una versione adattiva che le aggiorna durante la lettura.
- **Uso negli standard.** Huffman è usato nella fase finale della codifica JPEG, dopo che i dati sono stati ridotti con altre tecniche, e negli standard CCITT, JBIG2, JPEG e MPEG 1, 2 e 4, di solito in combinazione con la RLE.

DA RICORDARE

Huffman costruisce un codice prefisso fondendo ripetutamente i due nodi di peso minore. I simboli frequenti ottengono codeword corte. La lunghezza media L soddisfa $E \leq L < E + 1$: coincide con l'entropia quando le frequenze sono potenze di $1/2$ (28 bit per AABABCAACAAADDDDD, 18 per «gazzella»), altrimenti la supera di poco (23 bit contro 22,53 per «pelliccia»).

Lo standard JPEG

La compressione con perdita delle immagini fisse: trasformazione di colore, sottocampionamento, blocchi 8×8 , trasformata del coseno, quantizzazione, ordinamento a serpentina e codifica di entropia, con un blocco calcolato passo per passo.

Che cos'è il JPEG

JPEG è l'acronimo di **Joint Photographic Experts Group**, il comitato che ha sviluppato lo standard per la compressione delle immagini fisse. Nel 1988 il gruppo scelse uno schema di codifica adattativo basato sulla trasformata discreta del coseno (DCT); nel 1991 fu presentata una proposta di standard, approvata dai membri del consorzio e divenuta standard ISO nel 1992.

Il JPEG è una compressione **lossy**: sfrutta la ridondanza spaziale (il valore di un pixel è prevedibile dai vicini) e le debolezze del sistema visivo umano, che nota poco le variazioni fini di luminosità e ancora meno quelle di colore. L'idea è quella del capitolo 27: ciò che percettivamente non conta si scarta, in modo irreversibile; ciò che resta si codifica senza perdita con RLE, codifica differenziale e Huffman.

I passi della codifica

La codifica avviene in tre fasi: **pre elaborazione**, **trasformazione** e **codifica**.



Figura 1. Pre elaborazione.

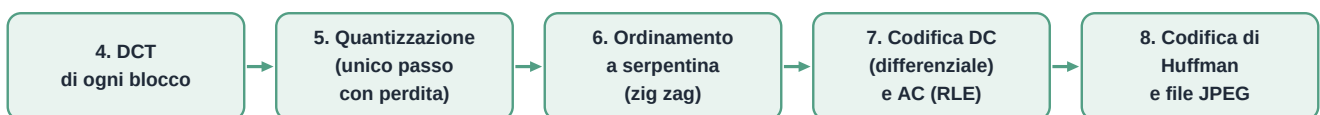


Figura 2. Trasformazione (passi 4 e 5) e codifica (passi 6, 7 e 8). Solo la quantizzazione e il sottocampionamento perdono informazione: tutti gli altri passi sono reversibili.

DOVE SI PERDE L'INFORMAZIONE

La DCT, di per sé, non perde nulla (si può invertire esattamente, a parte l'arrotondamento). Le perdite del JPEG sono due: il **sottocampionamento della cromaticanza** e la **quantizzazione dei coefficienti**. La seconda è la più importante, ed è la manopola con cui si regola la qualità.

Pre elaborazione

TRASFORMAZIONE DI COLORE, DA RGB A YCBCR

Il primo passo converte ogni pixel da RGB a **YCbCr** (capitolo 17): Y è la luminanza, Cb e Cr codificano il colore. La trasformazione è **reversibile** (a meno degli arrotondamenti) e permette di sfruttare il fatto che l'occhio è più sensibile alla luminanza che alla cromaticità. Le formule sono:

$$Y = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B$$

$$C_b = -0,169 \cdot R - 0,331 \cdot G + 0,5 \cdot B + 128 \quad C_r = 0,5 \cdot R - 0,419 \cdot G - 0,081 \cdot B + 128$$

CORREZIONE

Le slide riportano per questa trasformazione la matrice con righe (0,299; 0,587; 0,114), (0,596; -0,275; -0,321) e (0,212; -0,523; 0,311). Quelle ultime due righe non sono Cb e Cr: sono i coefficienti dello spazio **YIQ** (usato dal sistema televisivo NTSC). Le formule giuste del JPEG sono quelle qui sopra.

SOTTOCAMPIONAMENTO DELLA CROMINANZA

L'occhio umano è più sensibile alla luminanza che alla cromaticità. Il JPEG conserva **tutti** i campioni di Y ma, per Cb e Cr, solo un campione ogni quattro (un pixel su due in orizzontale e uno su due in verticale: schema 4:2:0). Si scarta così metà dei dati: per ogni quattro pixel, invece di $4 + 4 + 4 = 12$ campioni se ne tengono $4 + 1 + 1 = 6$. Questo passo è con perdita ed è irreversibile; il capitolo 17 ne mostra l'effetto, quasi invisibile su Cb e Cr.

DIVISIONE IN BLOCCHI 8×8

Per sfruttare al meglio la ridondanza, che localmente è maggiore che sull'immagine intera, e per semplificare i calcoli, il JPEG divide ogni canale in «quadrotti» 8×8 di 64 pixel, non sovrapposti. Ogni quadrotto è elaborato in modo indipendente dagli altri: è l'origine della «quadrettatura» visibile nelle immagini molto compresse (figura 10). Se le dimensioni dell'immagine non sono multipli di 8, i blocchi ai bordi si completano ripetendo i pixel del bordo.

SHIFT DEI LIVELLI DI GRIGIO

Prima della DCT a ogni pixel si sottrae 2^{n-1} , con 2^n numero di livelli dell'immagine: per 256 livelli si sottrae $2^7 = 128$. In questo modo il grigio medio (128) diventa 0 e i valori sono centrati intorno allo zero. Alla decodifica si somma 128 e si limita il risultato a 0...255.

La trasformata discreta del coseno (DCT)

Il JPEG trasforma ogni blocco 8×8 con la **DCT** (Discrete Cosine Transform), un algoritmo della famiglia delle trasformate di Fourier (capitolo 26). Si dimostra che, statisticamente, la DCT **decorrela** al massimo i dati, e questo permette rapporti di compressione maggiori nella fase di codifica. Per un blocco di $N \times N$ pixel ($N = 8$ nel JPEG) la trasformata diretta e l'inversa sono:

$$F(u, v) = (2 / N) \cdot \sum_x \sum_y C(u) C(v) f(x, y) \cos[(2x + 1)u\pi / 2N] \cos[(2y + 1)v\pi / 2N]$$

$$f(x, y) = (2 / N) \cdot \sum_u \sum_v C(u) C(v) F(u, v) \cos[(2x + 1)u\pi / 2N] \cos[(2y + 1)v\pi / 2N]$$

con $C(k) = 1 / \sqrt{2}$ per $k = 0$ e $C(k) = 1$ negli altri casi. Un calcolo diretto richiede un numero di operazioni proporzionale a N^2 per coefficiente; esistono algoritmi «veloci» derivati dalla trasformata di Fourier veloce.

Il nome non deve confondere: la DCT non cambia i dati, cambia la **base** in cui sono scritti. Un blocco 8×8 è un vettore in uno spazio a 64 dimensioni. Nella **base impulsiva** (canonica) è la somma pesata di 64 immagini nere con un solo pixel pari a 1, e i pesi sono i valori dei pixel. La DCT lo riscrive come somma pesata di 64 **immagini di base** fatte di coseni di frequenza crescente (figura 3): i pesi sono i coefficienti $F(u, v)$.

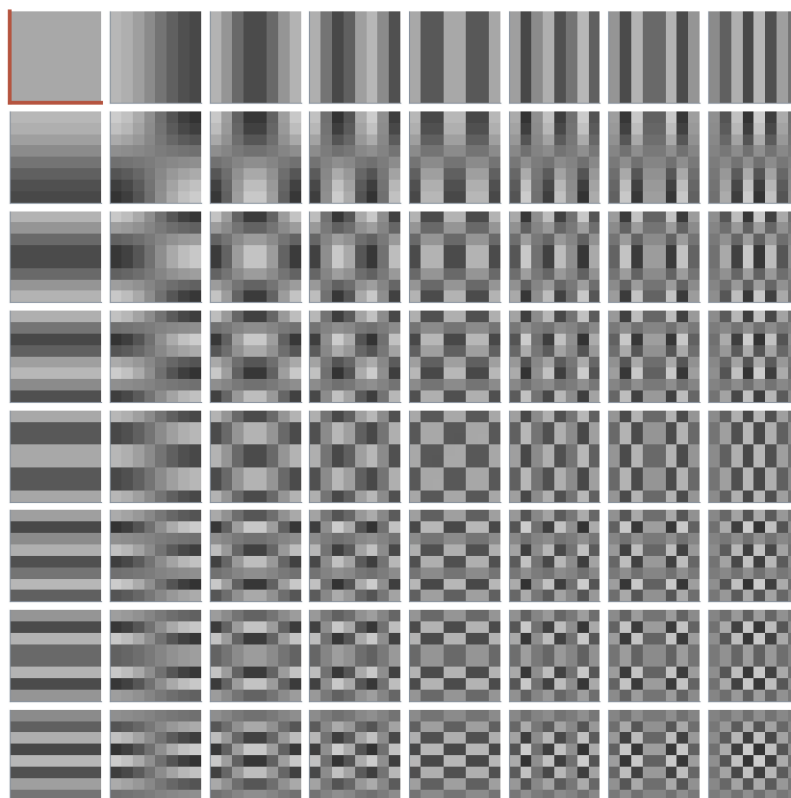


Figura 3. Le 64 immagini di base della DCT 8×8 . In alto a sinistra (bordo rosso) la base costante; procedendo verso destra aumenta la frequenza orizzontale, verso il basso quella verticale. Grigio medio = 0, chiaro = positivo, scuro = negativo.

- Il coefficiente in alto a sinistra, $F(0, 0)$, è proporzionale al **valore medio** della luminanza del blocco: è il **coefficiente DC** (dal nome della corrente continua). Con la normalizzazione usata qui vale 8 volte la media dei 64 pixel.
- Gli altri 63 sono i **coefficienti AC**: misurano quanto del blocco assomiglia a ciascuna onda di frequenza maggiore. Nelle immagini naturali i coefficienti a bassa frequenza (in alto a sinistra) sono grandi, quelli ad alta frequenza (in basso a destra) piccoli, che è proprio ciò che serve alla compressione.

UN BLOCCO DI ESEMPIO

Si usa il blocco 8×8 delle slide. Sottraendo 128 a ogni pixel e applicando la DCT si ottengono i coefficienti della figura 4 (arrotondati all'intero).

Blocco originale								Dopo lo shift								Coefficienti DCT							
52	55	61	66	70	61	64	73	-76	-73	-67	-62	-58	-67	-64	-55	-414	-29	-62	25	55	-20	-1	2
63	59	66	90	109	85	69	72	-65	-69	-62	-38	-19	-43	-59	-56	6	-21	-62	8	12	-7	-6	7
62	59	68	113	144	104	66	73	-66	-69	-60	-15	16	-24	-62	-55	-46	8	77	-26	-30	10	6	-5
63	58	71	122	154	106	70	69	-65	-70	-57	-6	26	-22	-58	-59	-49	12	34	-14	-10	6	1	1
67	61	68	104	126	88	68	70	-61	-67	-60	-24	-2	-40	-60	-58	11	-8	-12	-2	0	1	-5	2
79	65	60	70	77	68	58	75	-49	-63	-68	-58	-51	-60	-70	-53	-10	1	3	-3	0	0	2	0
85	71	64	59	55	61	65	83	-43	-57	-64	-69	-73	-67	-63	-45	-3	-1	1	0	1	-4	2	-3
87	79	69	68	65	76	78	94	-41	-49	-59	-60	-63	-52	-50	-34	-1	-1	0	-3	0	0	-1	0

Figura 4. Il blocco di esempio, la versione senza media (sottratto 128) e i suoi coefficienti DCT. Il coefficiente DC, in alto a sinistra, vale -414: grande e negativo perché il blocco è più scuro di 128.

Il coefficiente DC è -414 e i coefficienti più grandi sono tutti vicini all'angolo in alto a sinistra: l'energia del blocco si è concentrata nelle basse frequenze. Questa concentrazione è il motivo per cui la DCT è utile alla compressione.

CORREZIONE

Nelle slide la matrice dei coefficienti DCT è «-415, -29, -62, 25, 55, -20, -1, 3» in prima riga e così via. Ricalcolando la DCT esatta del blocco mostrato nelle slide si ottiene «-414, -29, -62, 25, 55, -20, -1, 2», e differenze di circa un'unità in altre posizioni (per esempio -48,9 al posto di -50 nella quarta riga). Le slide usano quindi valori approssimati o calcolati su un blocco leggermente diverso. In questa dispensa tutti i numeri sono ricalcolati in modo esatto.

La quantizzazione

Si riduce il numero di valori che i coefficienti possono assumere, e quindi i bit necessari a memorizzarli, riducendone la **precisione**. Dato un fattore di quantizzazione Q e un coefficiente F :

$$F_q = \text{round}(F / Q) \quad \text{valore ricostruito} = F_q \cdot Q$$

È il passo con perdita: la divisione e l'arrotondamento non si possono invertire. Più grande è Q , più coefficienti diventano 0 (e nulla costa codificare uno 0) e maggiore è la perdita.

Non conviene usare un solo Q per tutti i 64 coefficienti. Si usa una **tabella di quantizzazione** $Q(i, j)$, uno per ogni coefficiente $F(i, j)$: le alte frequenze, a cui l'occhio è poco sensibile, sono divise per numeri grandi; le basse per numeri piccoli. Lo standard fornisce due tabelle, una per la luminanza e una per la cromaticanza (la cromaticanza è quantizzata in modo più pesante); l'utente può scegliere tabelle proprie, ma in quel caso vanno trasmesse insieme ai dati.

Luminanza								Cromaticanza							
16	11	10	16	24	40	51	61	17	18	24	47	99	99	99	99
12	12	14	19	26	58	60	55	18	21	26	66	99	99	99	99
14	13	16	24	40	57	69	56	24	26	56	99	99	99	99	99
14	17	22	29	51	87	80	62	47	66	99	99	99	99	99	99
18	22	37	56	68	109	103	77	99	99	99	99	99	99	99	99
24	35	55	64	81	104	113	92	99	99	99	99	99	99	99	99
49	64	78	87	103	121	120	101	99	99	99	99	99	99	99	99
72	92	95	98	112	100	103	99	99	99	99	99	99	99	99	99

Figura 5. Le tabelle di quantizzazione standard del JPEG, per luminanza e cromaticanza (qualità 50). I valori crescono verso il basso a destra, dove stanno le alte frequenze.

IL QUALITY FACTOR

L'utente sceglie il **grado di quantizzazione** con un *quality factor* QF da 1 a 100: più è alto, più i fattori di quantizzazione sono piccoli e minore è la perdita. Le slide dicono che la tabella adottata è «una copia delle tabelle in cui gli elementi sono divisi per QF». In realtà la scalatura non è una semplice divisione. Nell'implementazione più diffusa (libjpeg) si calcola un fattore $S = 5000 / QF$ se $QF < 50$, e $S = 200 - 2 \cdot QF$ se $QF \geq 50$, e ogni elemento diventa $\lfloor (Q \cdot S + 50) / 100 \rfloor$, limitato tra 1 e 255.

Con $QF = 50$ le tabelle sono quelle di figura 5 ($S = 100$); con $QF = 100$ tutti i fattori valgono 1; con $QF = 10$ i fattori sono 5 volte più grandi di quelli a 50 ($S = 500$).

IL BLOCCO DI ESEMPIO, QUANTIZZATO

Coefficienti DCT									Tabella luminanza									Quantizzati							
-414	-29	-62	25	55	-20	-1	2	÷	16	11	10	16	24	40	51	61	=	-26	-3	-6	2	2	0	0	0
6	-21	-62	8	12	-7	-6	7		12	12	14	19	26	58	60	55		1	-2	-4	0	0	0	0	0
-46	8	77	-26	-30	10	6	-5		14	13	16	24	40	57	69	56		-3	1	5	-1	-1	0	0	0
-49	12	34	-14	-10	6	1	1		14	17	22	29	51	87	80	62		-3	1	2	0	0	0	0	0
11	-8	-12	-2	0	1	-5	2		18	22	37	56	68	109	103	77		1	0	0	0	0	0	0	0
-10	1	3	-3	0	0	2	0		24	35	55	64	81	104	113	92		0	0	0	0	0	0	0	0
-3	-1	1	0	1	-4	2	-3		49	64	78	87	103	121	120	101		0	0	0	0	0	0	0	0
-1	-1	0	-3	0	0	-1	0		72	92	95	98	112	100	103	99		0	0	0	0	0	0	0	0

Figura 6. Divisione di ciascun coefficiente per il corrispondente fattore e arrotondamento.

Dopo la quantizzazione restano 17 coefficienti diversi da zero su 64; gli altri 47 sono nulli e quasi tutti nella zona in basso a destra. Si noti il vantaggio per la codifica: con pochi livelli e tanti zeri i simboli sono pochi e frequenti, quindi i codici di Huffman sono brevi.

Ordinamento a serpentina

I 64 coefficienti quantizzati si riordinano in un vettore con il percorso a **serpentina** (zig zag) della figura 7, che parte dal DC e percorre le diagonali dalle basse alle alte frequenze. Lo scopo è creare **lunghe sequenze di zeri** verso la fine del vettore, che la codifica successiva comprime molto bene.

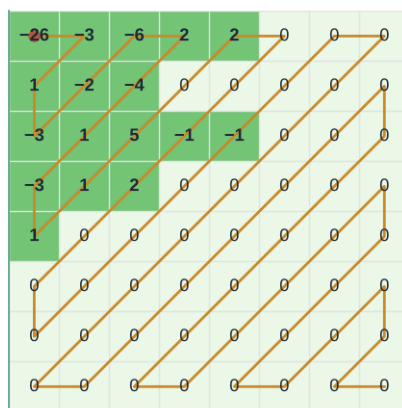


Figura 7. Il percorso a serpentina sul blocco quantizzato (le celle diverse da zero sono più scure). I coefficienti diversi da zero stanno quasi tutti all'inizio.

Per il blocco di esempio il vettore, in cui l'ultimo coefficiente non nullo è alla posizione 25 (contando da 0), è:

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

dove **EOB** (end of block) indica che tutti i coefficienti rimanenti sono zero.

Codifica dei coefficienti DC e AC

A questo punto i coefficienti seguono due strade diverse.

- I **coefficienti DC** (il primo di ogni blocco) sono codificati con una **codifica differenziale** (capitolo 27): blocchi vicini hanno medie simili, quindi si memorizza la differenza dal DC del blocco precedente.
- I **coefficienti AC** (gli altri 63) sono codificati con una **codifica a lunghezza di sequenza** (RLE): data la sequenza, si memorizza il numero di zeri che precedono ogni valore diverso da zero.

In entrambi i casi il risultato è poi codificato con codici di Huffman le cui tabelle, ricavate da calcoli statistici preventivi, sono fornite dallo standard (esistono tabelle diverse per luminanza e cromaticità, e per DC e AC).

CATEGORIE E BIT AGGIUNTIVI

Un valore v non si codifica direttamente: si individua la sua **categoria** n (chiamata anche SSSS), cioè il numero di bit necessari per rappresentare il suo valore assoluto. La categoria si codifica con Huffman; seguono poi n bit aggiuntivi che dicono quale valore della categoria è, con questa regola:

- se $v > 0$, i bit aggiunti sono le n cifre di v in binario;
- se $v < 0$, sono le n cifre di $(v - 1)$ in complemento a due, ovvero le n cifre di v in complemento a due diminuite di 1 (equivale a invertire i bit di $|v|$);
- se $v = 0$ la categoria è 0 e non si aggiunge nessun bit.

Categoria n	Valori	Bit aggiuntivi
0	0	0
1	-1, 1	1
2	-3, -2, 2, 3	2
3	da -7 a -4 e da 4 a 7	3
4	da -15 a -8 e da 8 a 15	4
5	da -31 a -16 e da 16 a 31	5

Tabella 1. Categorie dei coefficienti JPEG (le categorie continuano con 6 per $|v|$ fino a 63, 7 fino a 127, e così via).

Esempio. Per $v = -9$ la categoria è 4 ($8 \leq 9 \leq 15$). Le quattro cifre di 9 sono 1001; per un valore negativo si invertono, e si ottiene 0110. Per $v = -3$ la categoria è 2: $3 = 11$ diventa 00. Per $v = 5$ la categoria è 3 e i bit sono 101.

IL COEFFICIENTE DC

Se i DC di due blocchi consecutivi sono -26 (primo blocco) e -17 (secondo), l'evento da codificare per il secondo è la **differenza** $\Delta = DC_{\text{corrente}} - DC_{\text{precedente}} = -17 - (-26) = +9$. La categoria di 9 è 4. Nella tabella dei codici DC di default per la luminanza la categoria 4 ha il codice di base 101:

Categoria	0	1	2	3	4	5	6	7	8
Codice di base	00	010	011	100	101	110	1110	11110	111110

Tabella 2. Codici di base DC di default per la luminanza (categorie da 0 a 8).

Il codice completo è il codice di base seguito dai 4 bit di 9 (1001): **1011001**, in 7 bit.

CORREZIONE

Le slide definiscono la differenza al contrario, come $\Delta = -26 - (-17) = -9$, cioè «precedente meno successivo», e ottengono il codice 1010110 (101 seguito da 0110). Lo standard prevede «corrente meno precedente»: il decodificatore ricostruisce il DC sommando Δ al DC precedente, e quindi serve questa convenzione. Il numero di bit (7) è lo stesso.

I COEFFICIENTI AC

Il vettore dei 63 AC si trasforma in una lista di coppie (*zeri precedenti*, *valore*), dette skip value. Per esempio la sequenza 0, 0, 0, 0, 11, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 12, 17 diventa (4, 11), (3, 3), (8, 12), (0, 17). Nel blocco di esempio si ottiene:

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

L'evento da codificare con Huffman è la coppia (*run*, *categoria*) del valore, seguita dai bit aggiuntivi come per il DC. Per la prima coppia, (0, -3), il valore -3 ha categoria 2, quindi l'evento è (0, 2): nella tabella AC di default per la luminanza ha il codice 01. I bit aggiuntivi per -3 sono 00 (come visto sopra), e il codice completo è **0100**. Se una sequenza di zeri è più lunga di 15, si inserisce un simbolo speciale (ZRL) per ogni gruppo di 16 zeri; se gli zeri arrivano fino alla fine del blocco, basta il simbolo EOB.

Evento (run / categoria)	0 / 1	0 / 2	0 / 3	1 / 2	2 / 1	6 / 1	EOB (0 / 0)
Codice di base	00	01	100	11011	11100	1111011	1010

Tabella 3. Le righe della tabella AC di default per la luminanza necessarie a questo esempio (le tabelle complete hanno 162 eventi).

LA SEQUENZA FINALE DEL BLOCCO

Tipo	Evento	Run / categoria	Codice di base	Bit aggiuntivi	Bit totali
DC	$\Delta = +9$	categoria 4	101	1001	7
AC	(0, -3)	0 / 2	01	00	4
AC	(0, 1)	0 / 1	00	1	3
AC	(0, -3)	0 / 2	01	00	4
AC	(0, -2)	0 / 2	01	01	4
AC	(0, -6)	0 / 3	100	001	6
AC	(0, 2)	0 / 2	01	10	4
AC	(0, -4)	0 / 3	100	011	6
AC	(0, 1)	0 / 1	00	1	3
AC	(0, -3)	0 / 2	01	00	4
AC	(0, 1)	0 / 1	00	1	3
AC	(0, 1)	0 / 1	00	1	3
AC	(0, 5)	0 / 3	100	101	6
AC	(1, 2)	1 / 2	11011	10	7
AC	(2, -1)	2 / 1	11100	0	6
AC	(0, 2)	0 / 2	01	10	4
AC	(6, -1)	6 / 1	1111011	0	8
AC	EOB	0 / 0	1010	nessuno	4

Tabella 4. Codifica del blocco di esempio, con il DC come differenza dal blocco precedente ($\Delta = +9$) e gli AC come coppie (*run*, *valore*). L'ultima riga è il simbolo di fine blocco.

Scrivendo di seguito i codici si ottiene la sequenza finale, di **86 bit**:

1011001010000101000101100001011010001100101000
0100110010111011101110000110111101101010

Il blocco originale occupava $8 \times 8 \times 8 = 512$ bit: il rapporto di compressione per questo blocco è $512 / 86 = 6,0$. La riduzione è ottenuta sia scartando informazione (quantizzazione), sia sfruttando la distribuzione dei simboli.

CORREZIONE

Le slide indicano una sequenza finale di 72 bit, ma la stringa che riportano, applicando le tabelle standard alla loro lista di coppie, ne ha 91, e al rapporto di compressione citato ($512 / 72$) corrisponderebbe quindi un valore sbagliato. Le due liste di coppie differiscono inoltre da quella qui ricavata (le slide hanno -4 invece di -3 in una posizione, per il motivo spiegato nella correzione sulla DCT). Il procedimento di codifica è comunque identico.

La decodifica e la perdita di informazione

Il decodificatore esegue i passi inversi: legge i codici e ricostruisce i coefficienti quantizzati, li **moltiplica per i fattori di quantizzazione** (ottenendo coefficienti approssimati), applica la DCT inversa, somma 128 e arrotonda, limitando i valori a $0 \dots 255$. Poiché la quantizzazione ha scartato informazione, il blocco ricostruito **non coincide** con l'originale.

Originale								Ricostruito								Differenza							
52	55	61	66	70	61	64	73	65	65	64	63	65	70	73	75	13	10	3	-3	-5	9	9	2
63	59	66	90	109	85	69	72	55	55	68	89	97	86	74	69	-8	-4	2	-1	-12	1	5	-3
62	59	68	113	144	104	66	73	52	49	75	121	135	106	76	67	-10	-10	7	8	-9	2	10	-6
63	58	71	122	154	106	70	69	64	50	74	129	146	109	75	70	1	-8	3	7	-8	3	5	1
67	61	68	104	126	88	68	70	79	54	62	105	119	90	67	70	12	-7	-6	1	-7	2	-1	0
79	65	60	70	77	68	58	75	84	58	52	72	81	67	61	70	5	-7	-8	2	4	-1	3	-5
85	71	64	59	55	61	65	83	85	69	58	59	63	63	68	77	0	-2	-6	0	8	2	3	-6
87	79	69	68	65	76	78	94	86	80	71	63	64	72	81	87	-1	1	2	-5	-1	-4	3	-7

Figura 8. Il blocco originale, quello ricostruito dopo la decodifica e la differenza. L'errore massimo è 13 livelli, il PSNR del blocco 32,7 dB: a occhio i due blocchi sono molto simili.

Il JPEG in pratica

Il vero strumento di controllo è il quality factor. Le immagini seguenti mostrano la fotografia a diversi livelli di qualità.

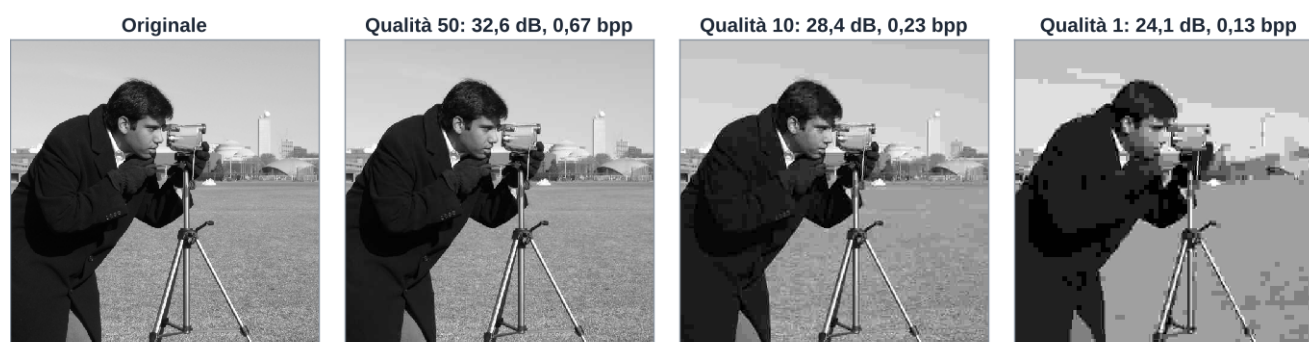


Figura 9. La fotografia compressa con JPEG a qualità 50, 10 e 1, con PSNR e bit per pixel (bpp). A qualità 50 le differenze si notano solo da vicino; a qualità 1 l'immagine è ridotta a blocchi uniformi.

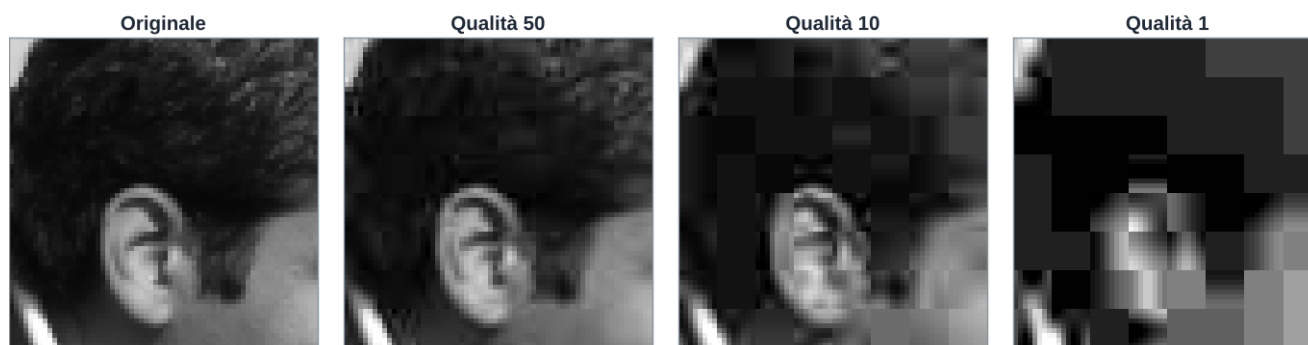


Figura 10. Ritaglio di 64×64 pixel, ingrandito 4 volte. Abbassando la qualità i blocchi 8×8 diventano visibili, perché ognuno è elaborato per conto suo: è la «quadrettatura» tipica delle forti compressioni.

Qualità	Byte	Bit per pixel	Rapporto (da 8 bpp)	PSNR (dB)
95	85 033	2,60	3,1	45,1
75	34 472	1,05	7,6	35,1
50	22 050	0,67	11,9	32,6
25	13 915	0,42	18,8	30,8
10	7 496	0,23	35,0	28,4
5	5 164	0,16	50,8	26,3
1	4 205	0,13	62,3	24,1

Tabella 5. File JPEG della fotografia 512×512 in scala di grigio (262 144 byte non compressi) al variare del quality factor. Dimensioni misurate con un codificatore JPEG standard; il rapporto è rispetto agli 8 bit per pixel originali.

A qualità 75 il file è circa 7,6 volte più piccolo dell'originale a 8 bit con un PSNR di 35 dB, e a qualità 50 il rapporto sale a circa 12; sotto qualità 10 il risparmio aumenta ancora ma la perdita diventa evidente. Questi sono valori per una immagine in scala di grigio; per le immagini a colori si ottengono rapporti più alti perché la cromaticità, sottocampionata e quantizzata più duramente, costa pochissimo.

Esempio a colori. Una fotografia 512×512 a 24 bit occupa 786 432 byte. Salvata in JPEG a qualità 75 (con il sottocampionamento 4:2:0) occupa 40 240 byte, un rapporto di 19,5, con un PSNR di 34,0 dB.

- JPEG funziona bene sulle **foto**grafie, dove le variazioni sono gradualmente, e male sui disegni al tratto, sul testo e sui bordi netti, dove le alte frequenze importanti vengono scartate e compaiono aloni intorno ai contorni.
- Ogni salvataggio ripete la quantizzazione: aprire e salvare più volte lo stesso JPEG accumula perdite. Per modificare una immagine conviene lavorare su una copia senza perdita (per esempio PNG) e comprimere in JPEG solo alla fine.
- Per immagini che devono restare identiche all'originale si usa una compressione lossless (PNG, che non fa parte del programma di questo corso).

DA RICORDARE

Il JPEG separa luminanza e cromaticità (YCbCr), sottocampiona la cromaticità, divide in blocchi 8×8 , sottrae 128 e applica la DCT; quantizza i coefficienti con una tabella (qui avviene la perdita principale, controllata dal quality factor); li riordina a serpentina; codifica il DC come differenza dal blocco precedente e gli AC come coppie (run, valore); infine applica codici di Huffman standard. Per il blocco di esempio: 512 bit diventano 86.