

Prima di procedere oltre...

Bit, Byte, KiloByte, MegaByte,...

- Bit = '0' oppure '1'
- Byte = 8 bit = 2^3
- KiloByte (KB) = 2^{10} byte = 1.024 byte
- MegaByte (MB) = 2^{20} byte = 1.048.576 byte
- GigaByte (GB) = 2^{30} byte = 1.073.741.824 byte
- TeraByte (TB) = 2^{40} byte = ...
- PetaByte (PB) = 2^{50} byte = ...
- ExaByte (EB) = 2^{60} byte = ...

Tipologie di codici

Nel seguito vedremo tipologie di rappresentazioni diverse:

- Senza assumere limitazioni sul numero di bit a disposizione:
per numeri [notazione binaria, ovvero posizionale con base 2]
- Disponendo di un numero di bit limitato:
 - numeri naturali
 - interi relativi [valore assoluto e segno, complemento a due]
 - “reali” [virgola fissa e virgola mobile]
 - valori logici, caratteri alfabetici, testi
 - suoni, immagini e sequenze video
 - codici per la rilevazione e correzione di errori
- Codici di compressione (senza | con perdita)

CODIFICA BINARIA DI CARATTERI

Quanti oggetti dobbiamo rappresentare?

- 10 cifre
 - 26 lettere minuscole + 26 lettere maiuscole = 52
 - ~30 segni di interpunzione
 - ~30 caratteri di controllo (LF, CR, EOF, ...)
- } ~ 120

➡ $k = \lceil \log_2 120 \rceil = 7$: 7 bit sono sufficienti

CODIFICA BINARIA DI CARATTERI

Quanti oggetti dobbiamo rappresentare?

- 10 cifre
 - 26 lettere minuscole + 26 lettere maiuscole = 52
 - ~30 segni di interpunzione
 - ~30 caratteri di controllo (LF, CR, EOF, ...)
- } ~ 120

 $k = \lceil \log_2 120 \rceil = 7$: 7 bit sono sufficienti

- Codice **ASCII**: utilizza **7 bit** \Rightarrow al massimo **128 caratteri**
- Codice **ASCII esteso**: utilizza **8 bit** \Rightarrow 256 caratteri
- Codice **UNICODE**: utilizza **16 bit** \Rightarrow 65536 caratteri
(anche quelli delle lingue orientali... ma non tutti!)
- I caratteri utili sono circa 200 000! ...e aumentano!
 \Rightarrow Estensione a 21 bit del codice UNICODE

ASCII Table

Dec Hx Oct Char

0	0	000	NUL	(null)
1	1	001	SOH	(start of heading)
2	2	002	STX	(start of text)
3	3	003	ETX	(end of text)
4	4	004	EOT	(end of transmission)
5	5	005	ENQ	(enquiry)
6	6	006	ACK	(acknowledge)
7	7	007	BEL	(bell)
8	8	010	BS	(backspace)
9	9	011	TAB	(horizontal tab)
10	A	012	LF	(NL line feed, new line)
11	B	013	VT	(vertical tab)
12	C	014	FF	(NP form feed, new page)
13	D	015	CR	(carriage return)
14	E	016	SO	(shift out)
15	F	017	SI	(shift in)
16	10	020	DLE	(data link escape)
17	11	021	DC1	(device control 1)
18	12	022	DC2	(device control 2)
19	13	023	DC3	(device control 3)
20	14	024	DC4	(device control 4)
21	15	025	NAK	(negative acknowledge)
22	16	026	SYN	(synchronous idle)
23	17	027	ETB	(end of trans. block)
24	18	030	CAN	(cancel)
25	19	031	EM	(end of medium)
26	1A	032	SUB	(substitute)
27	1B	033	ESC	(escape)
28	1C	034	FS	(file separator)
29	1D	035	GS	(group separator)
30	1E	036	RS	(record separator)
31	1F	037	US	(unit separator)

Dec Hx Oct Char

32	20	040	SPACE	
33	21	041	!	
34	22	042	"	
35	23	043	#	
36	24	044	\$	
37	25	045	%	
38	26	046	&	
39	27	047	'	
40	28	050	(
41	29	051)	
42	2A	052	*	
43	2B	053	+	
44	2C	054	,	
45	2D	055	-	
46	2E	056	.	
47	2F	057	/	
48	30	060	0	
49	31	061	1	
50	32	062	2	
51	33	063	3	
52	34	064	4	
53	35	065	5	
54	36	066	6	
55	37	067	7	
56	38	070	8	
57	39	071	9	
58	3A	072	:	
59	3B	073	;	
60	3C	074	<	
61	3D	075	=	
62	3E	076	>	
63	3F	077	?	

Dec Hx Oct Char

64	40	100	@	
65	41	101	A	
66	42	102	B	
67	43	103	C	
68	44	104	D	
69	45	105	E	
70	46	106	F	
71	47	107	G	
72	48	110	H	
73	49	111	I	
74	4A	112	J	
75	4B	113	K	
76	4C	114	L	
77	4D	115	M	
78	4E	116	N	
79	4F	117	O	
80	50	120	P	
81	51	121	Q	
82	52	122	R	
83	53	123	S	
84	54	124	T	
85	55	125	U	
86	56	126	V	
87	57	127	W	
88	58	130	X	
89	59	131	Y	
90	5A	132	Z	
91	5B	133	[
92	5C	134	\	
93	5D	135]	
94	5E	136	^	
95	5F	137	_	

Dec Hx Oct Char

96	60	140	`	
97	61	141	a	
98	62	142	b	
99	63	143	c	
100	64	144	d	
101	65	145	e	
102	66	146	f	
103	67	147	g	
104	68	150	h	
105	69	151	i	
106	6A	152	j	
107	6B	153	k	
108	6C	154	l	
109	6D	155	m	
110	6E	156	n	
111	6F	157	o	
112	70	160	p	
113	71	161	q	
114	72	162	r	
115	73	163	s	
116	74	164	t	
117	75	165	u	
118	76	166	v	
119	77	167	w	
120	78	170	x	
121	79	171	y	
122	7A	172	z	
123	7B	173	{	
124	7C	174		
125	7D	175	}	
126	7E	176	~	
127	7F	177	DEL	

CODIFICA BINARIA DI TESTI

Cosa dobbiamo rappresentare?

- Caratteri che compongono il testo
- Caratteristiche di formattazione:
 - stile di scrittura (**grassetto**, *corsivo*, sottolineato,...)
 - dimensione del carattere
 - tipo o “font” (times new roman, arial, courier,...) del carattere
 - l’ampiezza dei margini della pagina
 - ...

➡ Esistono diversi codici (comunemente chiamati *formati*) capaci di rappresentare un insieme più o meno ampio di caratteristiche

CODIFICA BINARIA DI TESTI

Cosa dobbiamo rappresentare?

- Caratteri che compongono il testo
- Caratteristiche di formattazione:
 - stile di scrittura (**grassetto**, *corsivo*, sottolineato,...)
 - dimensione del carattere
 - tipo o “font” (times new roman, **arial**, *courier*,...) del carattere
 - l’ampiezza dei margini della pagina
 - ...

➡ Esistono diversi codici (comunemente chiamati *formati*) capaci di rappresentare un insieme più o meno ampio di caratteristiche

-*Testo semplice* (text o txt): corrisponde a una sequenza di caratteri ASCII senza caratteristiche di formattazione

-*Testo arricchito* (rich text format o rtf): in grado di rappresentare un ristretto insieme di formattazioni (stile, dimensione e colore dei caratteri)

-*Testo Word* (document o doc): rappresenta insieme ampio di formattazioni

Formati dei documenti per stampa e visualizzazione

- Formati standardizzati che sono orientati alla produzione di documenti destinati a *stampa* e *visualizzazione*:
 - Documenti in formato **PDF** (*Portable Document Format*, formato di documento portabile)
 - Documenti in formato **PS** (*Postscript*) - l'applicazione GSview (Ghostview) legge il formato postscript
- Non limitati alla rappresentazione di testo ma possono includere anche immagini e disegni

Tipologie di codici

Nel seguito vedremo tipologie di rappresentazioni diverse:

- Senza assumere limitazioni sul numero di bit a disposizione:
per numeri [notazione binaria, ovvero posizionale con base 2]
- Disponendo di un numero di bit limitato:
 - numeri naturali
 - interi relativi [valore assoluto e segno, complemento a due]
 - “reali” [virgola fissa e virgola mobile]
 - valori logici, caratteri alfabetici, testi
 - suoni, immagini e sequenze video
 - codici per la rilevazione e correzione di errori
- Codici di compressione (senza | con perdita)

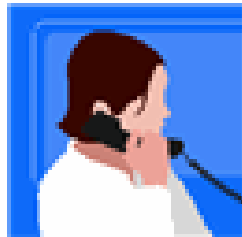
Rappresentazione di informazioni multimediali: concetti generali

- Con appositi metodi e codifiche si possono rappresentare informazioni complesse con sequenze di bit:
 - Suoni
 - Immagini
 - Filmati
- I problemi collegati alla gestione della rappresentazione, memorizzazione, trasmissione e elaborazione di queste informazioni sono:
 - occupazione elevata di memoria: per questo si utilizzano tecniche di compressione dei dati che consentono di ridurre la richiesta di memoria
 - la richiesta di elevate capacità di elaborazione (es: calcoli necessari per riprodurre un video codificato in divx)

La digitalizzazione

rappresentazione anche approssimata
tramite sequenze di bit

- Voce



- Suoni



- Immagini

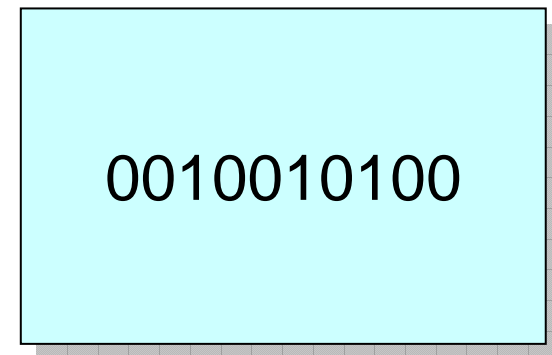


- Filmati



Informazione
“continua”

Codifica digitale



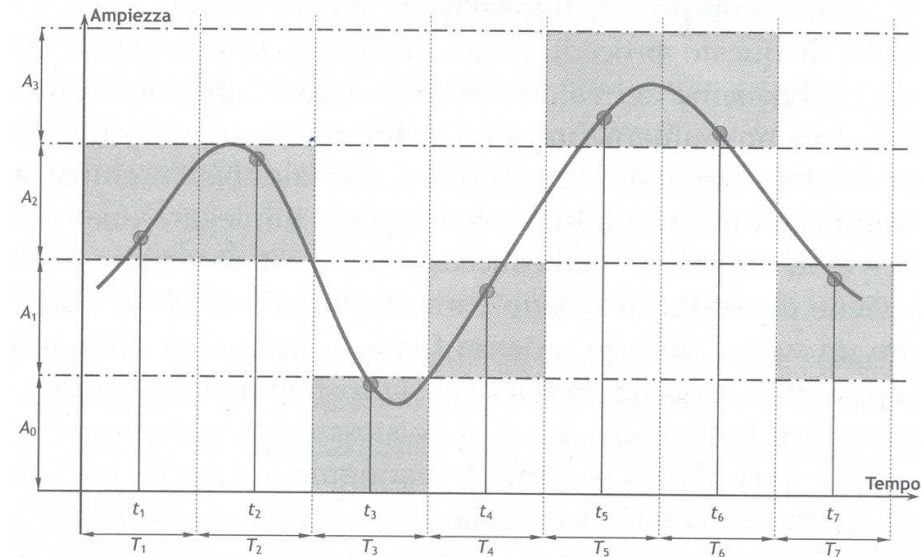
Nel calcolatore

Note al lucido precedente

- Problema: codificare in modo digitale una grandezza fisica (es. volume di un suono) i cui valori variano in un intervallo continuo e possono essere infiniti.
- Soluzione: processo di *digitalizzazione*, o *conversione analogico-digitale*. Comprende due attività:
 - *Quantizzazione*: discretizzazione dei valori attraverso approssimazione con uno dei valori compresi fra quelli previsti nella codifica
 - *Campionamento*: se la grandezza inoltre varia anche nel tempo (es. un suono) o nello spazio (es. colore di un'immagine) occorre selezionare un insieme finito di valori, ad intervalli (nel tempo o nello spazio) costanti

CODIFICA DI SUONI

- ... in maniera digitale attraverso *campionamento* (nel tempo) e *quantizzazione* (in ampiezza)
- **Frequenza di campionamento:** numero di campioni acquisiti nell'unità di tempo (es. numero di campioni al secondo, misurato in Hertz) - Es. frequenza di 5 Hz significa acquisizione di 5 campioni al secondo
- Il **numero di bit** necessari per codificare ogni valore è il parametro che qualifica la quantizzazione - Es. con 8 bit posso codificare 256 valori diversi



Per ciascun intervallo di tempo viene scelto l'istante di campionamento t_i in cui viene rilevato il valore della grandezza

La quantizzazione è su 4 livelli, quindi il risultato del campionamento è $A_2 A_2 A_0 A_1 A_3 A_3 A_1$

La successione (codifica digitale) potrebbe essere
10 10 00 01 11 11 01

Rappresentazione dei suoni nei CD audio

- Segnali audio: onde analogiche

→ campionamento + quantizzazione → valori digitali



Frequenza di campionamento $f_c = 44.1$ kHz (44.100 campioni) per i 2 canali
(frequenze udibili dall'orecchio umano fino a circa 22 kHz)

16 bit/campione (2 byte per campione) → 65536 livelli di quantizzazione
(65536 valori diversi per campione)

- Dimensione richieste

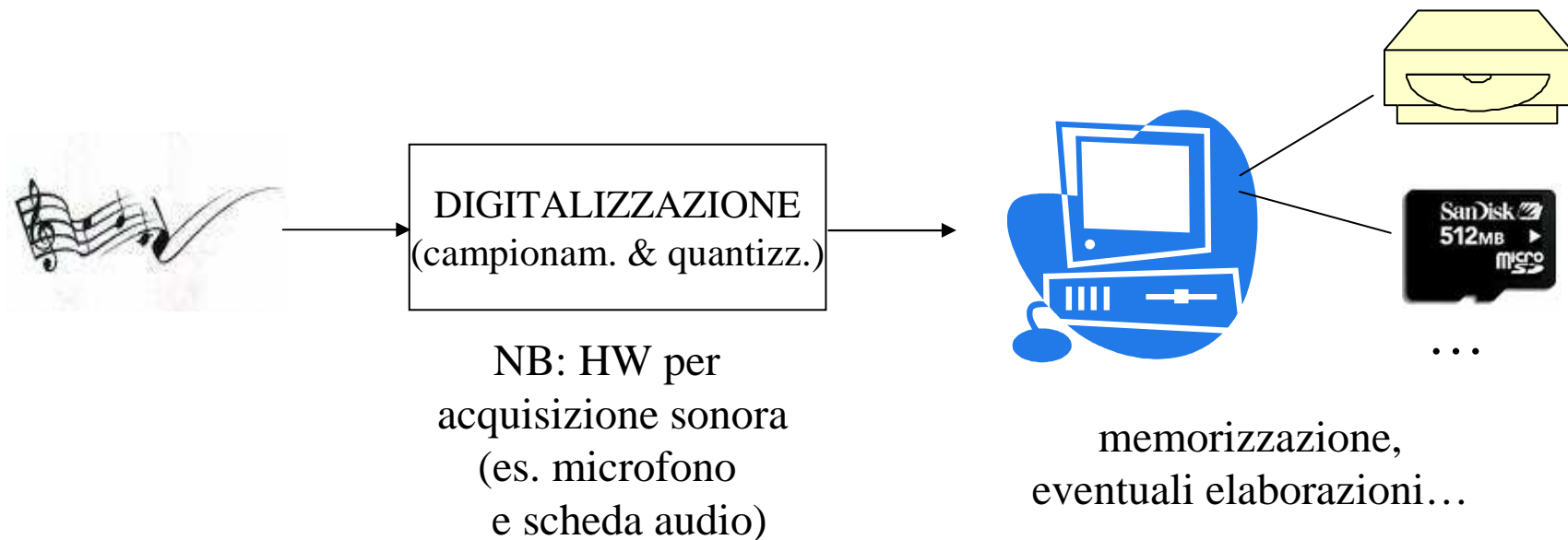
$$N_{\text{bit}} = F(\text{durata}, f_c, \text{bit/campione})$$

➡ Nello standard CDDA (CD Digital Audio)

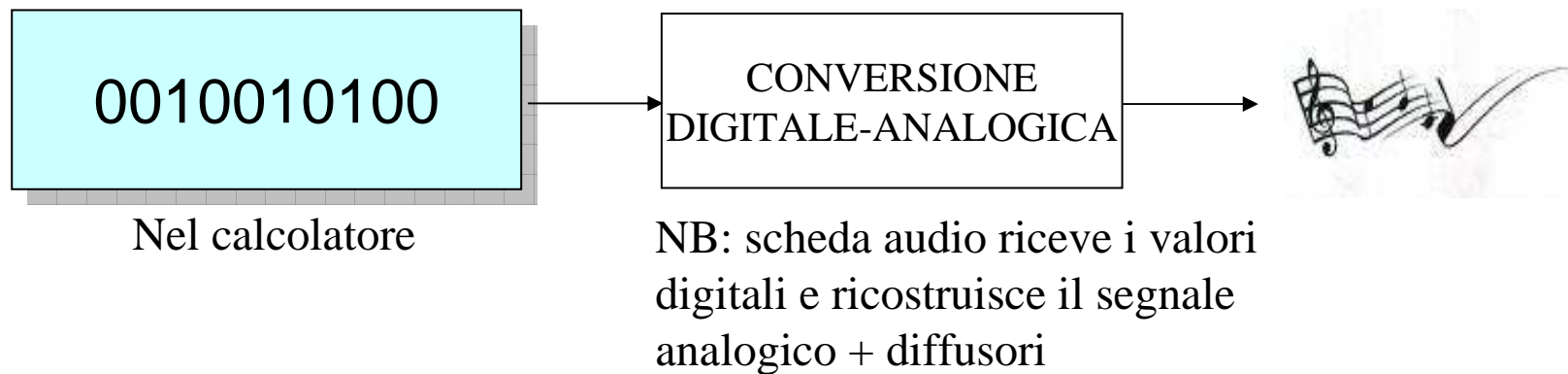
70 minuti di registrazione richiedono:

$$70 \times 60 \times 44.100 \times 2 \times 2 \text{ byte} \approx 750 \text{ MByte}$$

Dal mondo reale (analogico) al calcolatore (digitale)



Dal calcolatore (digitale) al mondo reale (analogico)



Alcuni formati audio

- **WAVE**: **.wav**, dimensioni elevate (corrisponde a quanto visto nei lucidi precedenti: ~1MB per minuto)
- **MP3** (MPEG-1 Layer 3): grande diffusione su Internet. Utilizza tecniche di compressione per ridurre le dimensioni (vedi poi).
MPEG nasce da un gruppo di lavoro di standardizzazione
- **MIDI** (*Musical Instrument Digital Interface*): **.mid**, i file memorizzano non suoni ma **comandi** (es. le note musicali di un particolare strumento) che vengono inviati ai dispositivi MIDI per riprodurre i suoni, occupano molto meno spazio dei **.wav**.

Es: tastiera musicale + calcolatore permettono di suonare un brano su tastiera che viene automaticamente trascritto in notazione musicale in un file midi che può poi essere anche modificato

CODIFICA DI IMMAGINI

- *Campionamento:*

- nel caso delle immagini è una discretizzazione nello spazio: si divide l'immagine in quadrati, per ognuno dei quali si preleva un campione che si considera rappresentativo di tutto il quadrato
- Questi quadrati sono trattati come elementi di base dell'immagine digitale: sono chiamati *pixel*
- Quanti più pixel vengono considerati per unità di superficie tanto più precisa sarà la riproduzione dell'immagine (*risoluzione*)

- *Quantizzazione:* codifica del colore associato a ogni pixel:

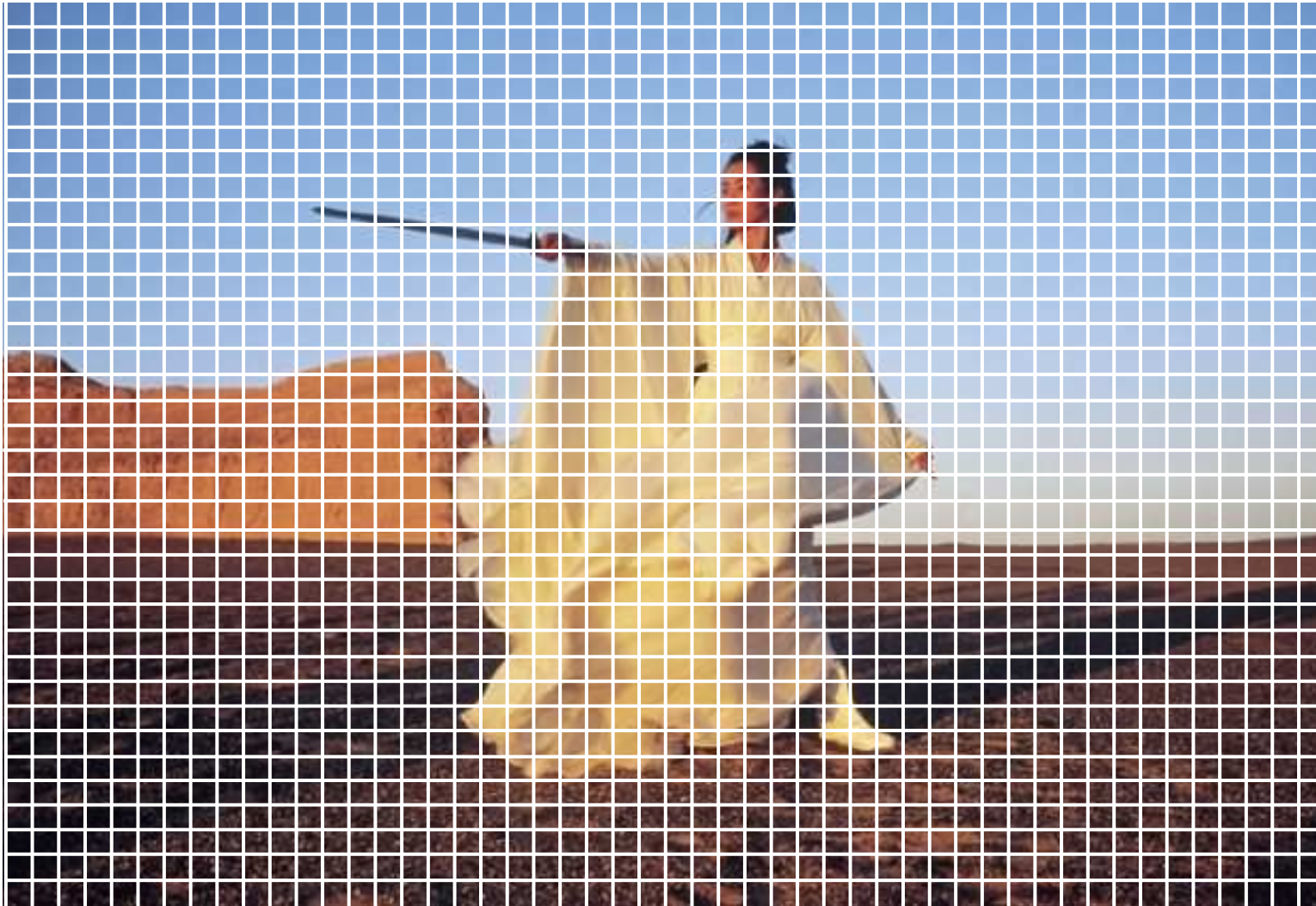
- Immagini in bianco e nero: 1 bit per pixel (b/n)
- Immagine a scala di grigi: es. un byte per pixel (256 livelli di grigio)
- Immagine a colori: es. con modello RGB (colore ottenuto per sovrapposizione di Rosso Verde e Blu) 3 byte per pixel, uno per ogni colore primario

Campionamento e quantizzazione: esempio

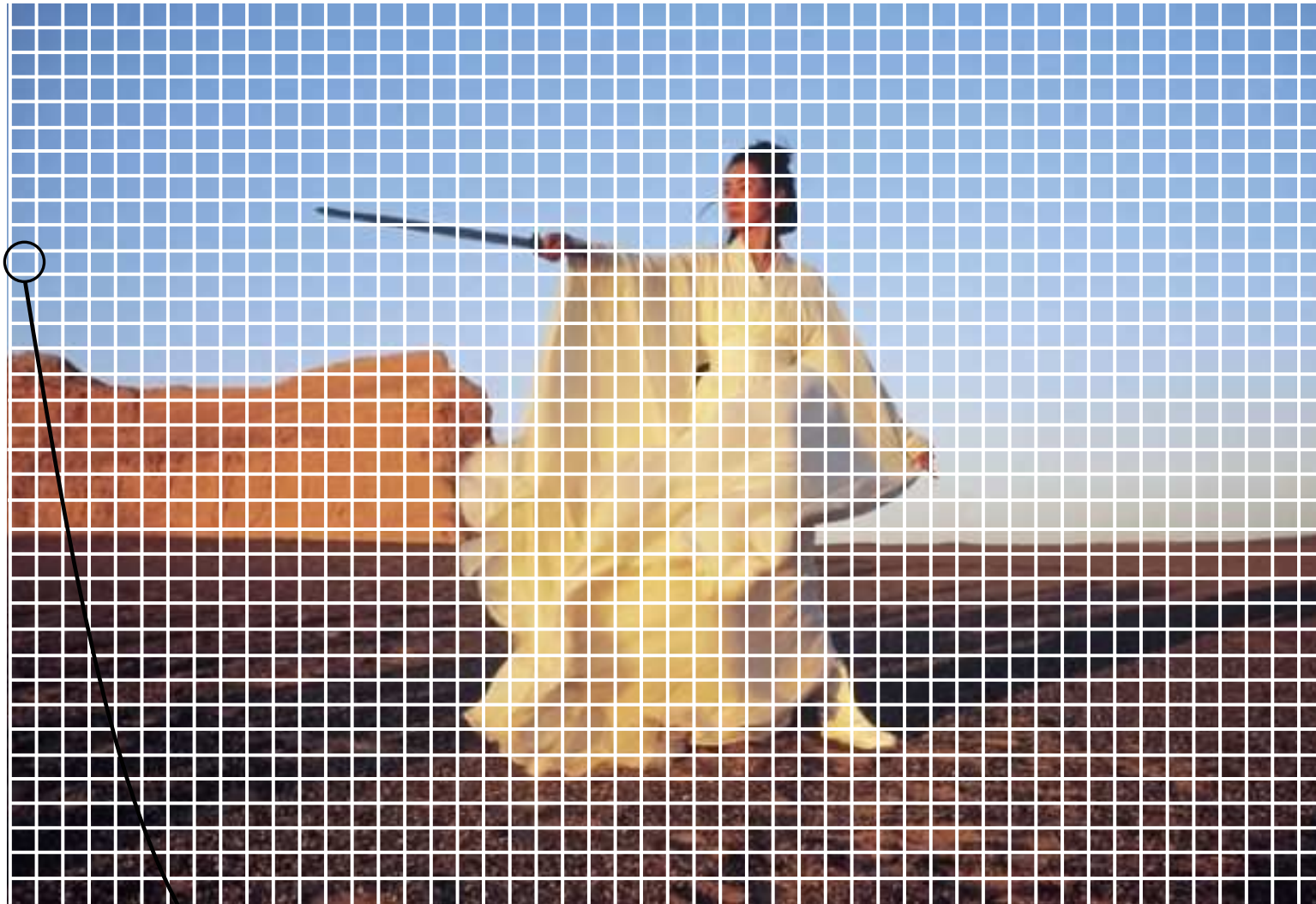


Maggie Cheung dal film “Hero” di Zhang Yimou (2002)

Campionamento



Quantizzazione



Per ogni pixel un certo numero di byte
es. 3 byte (colori) o 1 byte (b/n)

Codifica bitmap (raster) di immagini

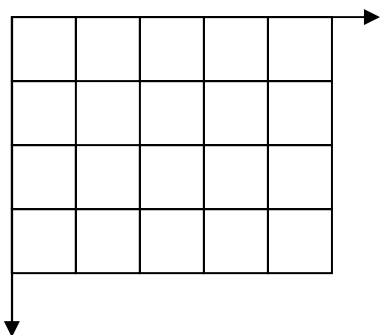


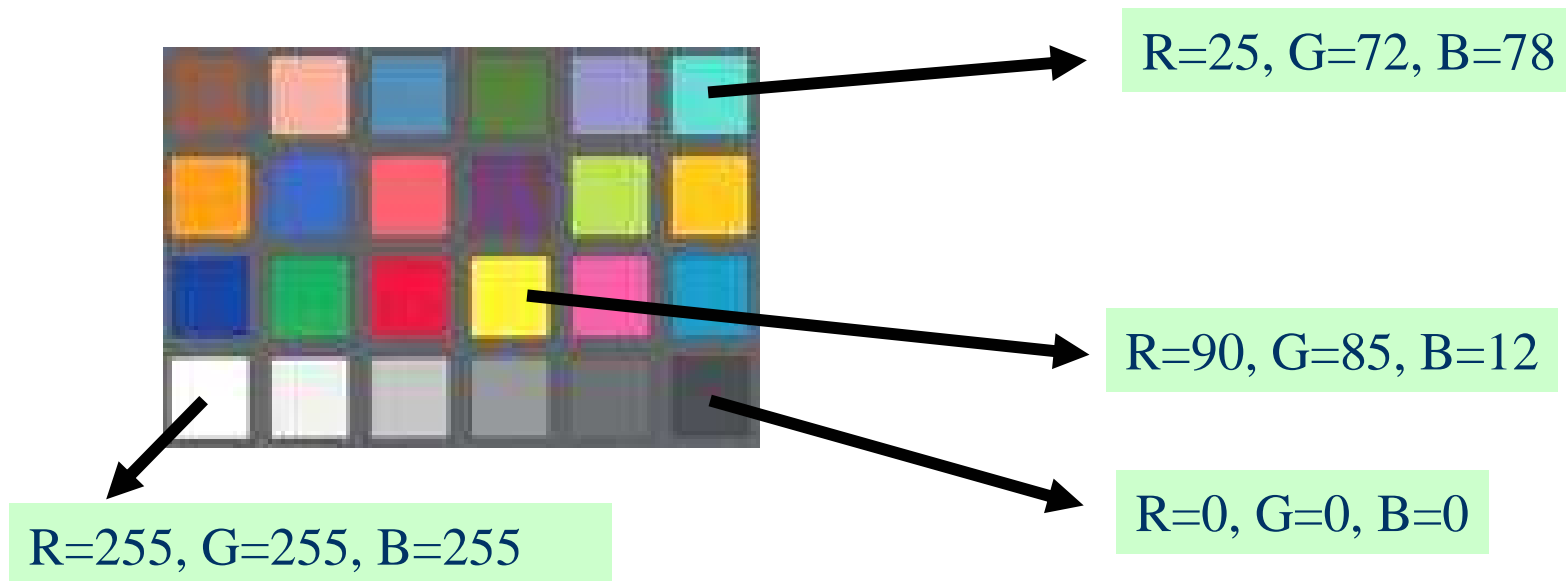
immagine *bitmap* (o *raster*):
matrice di *pixel*

- L'immagine è rappresentata direttamente come matrice di pixel
- **Risoluzione**: numero di pixel orizzontali X verticali
- Ad ogni pixel è riservato un certo numero di bit:

1 bit (per pixel)	→ 2 informazioni diverse (b/n)
4 bit (per pixel)	→ 16 colori o livelli di grigio
8 bit (per pixel)	→ 256 colori o livelli di grigio
16 bit (per pixel)	→ 64K colori ($2^{16} = 2^{10} \times 2^6$)
24 bit (per pixel)	→ 16M = 2^{24} colori (più di 16 milioni di colori)

Immagini bitmap RGB

- L'informazione sul colore di ogni pixel occupa 3 byte, uno per ogni colore primario (rosso, verde, blu)
- La quantità di colore primario è data da un valore tra 0 e 255 (o equivalentemente è rappresentato da una sequenza di bit a partire da 00000000 fino a 11111111)



Problemi con immagini bitmap (o raster):

- ogni ingrandimento fa perdere qualità (l'immagine si “sgrana”):
vedi esempio nel prossimo lucido
 - per evitare questo problema: **codifica vettoriale** (vedi poi)
- occupano molta memoria. Possibili soluzioni:
 - uso di tavolozze (palette)
Es. immagine a 256 colori:
uso $256 \times 3 = 768$ byte per codificare la tavolozza
e solo un byte per pixel (seleziona uno dei 256 colori)
 - formati compressi (trattati in una lezione successiva)
 - codifica vettoriale

Immagini bitmap: esempio



Un'immagine RGB occupa:

3 byte x risoluzione verticale x risoluzione orizzontale

Esempio: un'immagine RGB con risoluzione 1024x768 pixel necessita di più di 2 MB

... e se la
ingrandisco si
sgrana...



Una soluzione: la tavolozza (palette)

- Spesso ogni immagine utilizza un numero limitato di colori (scelti comunque tra l'insieme ampio di tutti i colori rappresentabili!)

Esempio banale: immagine RGB che usa due soli colori (qualsiasi)

- Dimensioni senza palette: *numero pixel * 3byte*
- Idea: il formato memorizza i colori usati in un'area specifica (palette) e per ogni pixel si indica quale dei due colori usato!

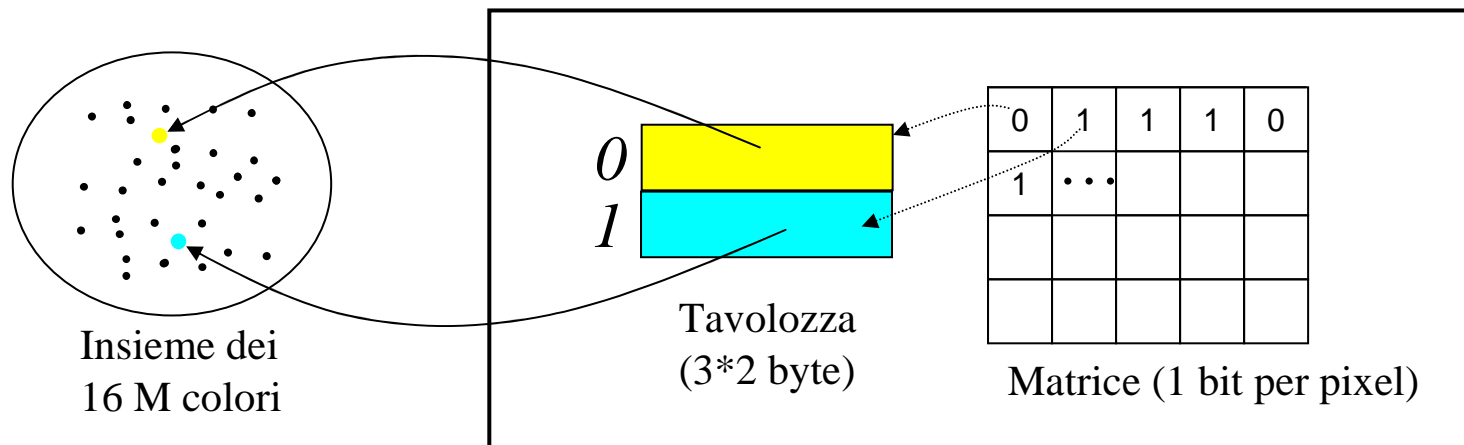


IMMAGINE CODIFICATA

Esercizio

Un'immagine di risoluzione 1024×768 utilizza 256 colori RGB.

Quali dimensioni occupa la sua codifica se si utilizza una palette?

E se non la si utilizza?

Esercizio

Un'immagine di risoluzione 1024*768 utilizza 256 colori RGB.

Quali dimensioni occupa la sua codifica se si utilizza una palette?

E se non la si utilizza?

Senza utilizzare palette:

$$\begin{aligned} 1024 * 768 * 3 \text{ bytes} &= 2^{10} * 3 * 2^8 * 3 \text{ bytes} \\ &= 2,25 * 2^{20} \text{ bytes} = 2,25 \text{ MB} \end{aligned}$$

Con utilizzo di palette:

$$\begin{aligned} 256 * 3 &= 768 \text{ bytes per codificare la tavolozza} + \\ 1024 * 768 * 1 &\text{ bytes per la matrice} \\ \Rightarrow \text{circa } 1024 * 768 &= 0,75 \text{ MB} \end{aligned}$$

I formati di file bitmap

- **BMP**: formato standard non compresso per MS Windows, 24 bit per pixel, **.bmp**. Gestisce palette a 2, 16, 256 colori + true color
- **TIFF** (*Tagged Image File Format*): alta qualità (32 bit per pixel), 16 milioni di colori (24 bit) + ulteriori proprietà, dimensioni file molto grandi, **.tif**, adottato da scanner e macchine fotografiche

FORMATI COMPRESSI:

- **GIF** (*Graphic Interchange Format*): formato compresso, 8 bit per pixel (256 colori), **.gif**
- **JPEG** (*Joint Picture Experts Group*): 16 milioni di colori, formato compresso (più del gif), **.jpg**
- **JPEG 2000**: formato ancor più compresso, **.j2k** o **.jp2**

Codifica vettoriale di immagini

- Le immagini sono rappresentate tramite un insieme di elementi grafici (linee, rettangoli, ellissi, archi e curve)
- Memorizzazione come *coordinate numeriche* o *formule matematiche* che specificano forma e posizione: occupa poca memoria (es. un cerchio solo centro e raggio, un segmento le coordinate degli estremi)
- Necessaria *un'operazione di rendering (rasterizzazione)* che, a partire dalla descrizione matematica, produca l'immagine raster
- *Programmi di tipo draw* (programmi di grafica vettoriale): es. Corel Draw, programmi di CAD, programmi di grafica tridimensionale
- Vantaggi:
 - Controllo accurato di linee e colori
 - Ingrandimento, riduzione, rotazione senza perdita
 - Possibilità inserimento testo attorno agli oggetti
- Alcuni formati: **DXF, DWG, CDR, AI, WMF**

Immagini vettoriali: esempio



CODIFICA DI SEQUENZE VIDEO

- In teoria, una sequenza video è semplicemente una sequenza di fotogrammi (immagini):
 - campionamento nel tempo (successione di fotogrammi)
 - ciascun fotogramma rappresentato normalmente:
campionamento nello spazio e quantizzazione
 - In pratica, le dimensioni risulterebbero troppo elevate. Facendo un conto “a spanne”, se un’immagine occupa 1 MB con 25 fotogrammi al secondo:
 - 25 MB al secondo
 - 25 x 60 MB al minuto = circa 1,5 GB al minuto!
- ➡ Tecniche di compressione specifiche per le sequenze video