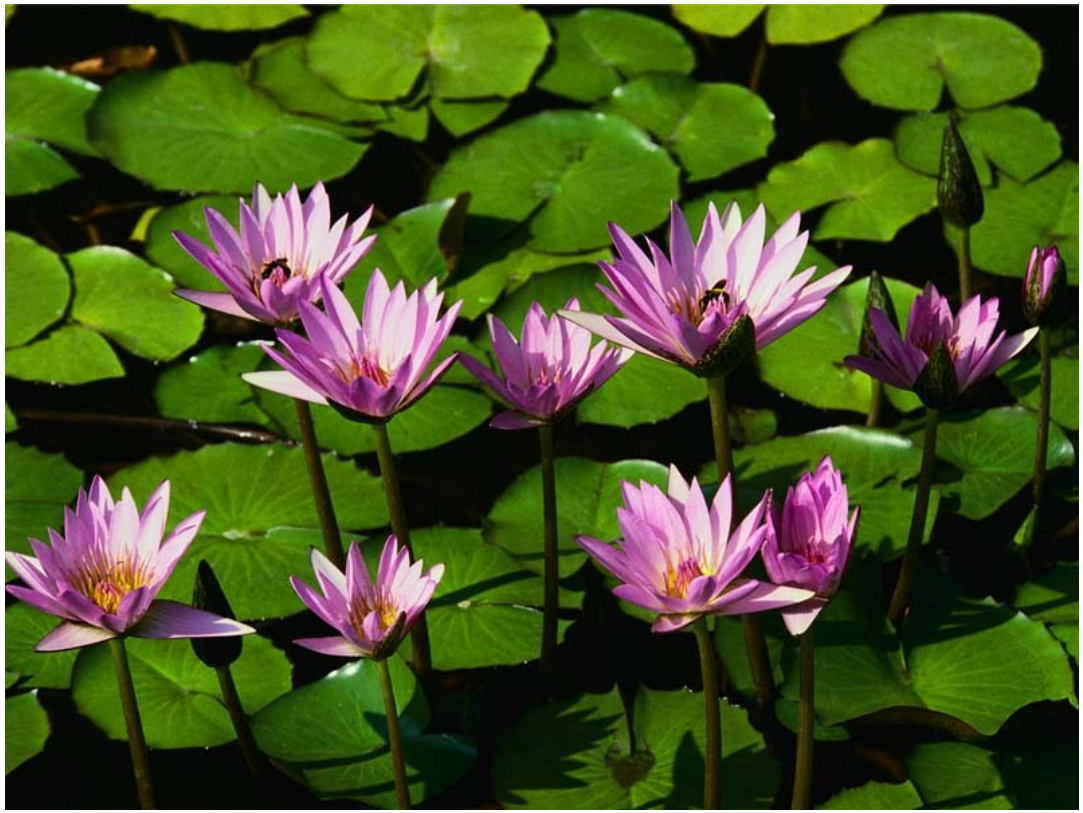
L1 Formato immagini, video e sistemi di acquisizione

Corso di Visione Artificiale
A.A. 2019/2020

Argomenti

- Formati di file per immagini
  - Formati raster: PNM, BMP, GIF, PNG, JPEG
  - Formati vettoriali: SVG
- Formati di file per video
- Esercitazione sul formato immagine

Formati raster



Formati raster

Formati raster

- Header
  - Formato
  - Dimensioni (WxHxD)
  - Colore (spazio e riferimenti colorimetrici)
  - Compressione o meno
  - Altre informazioni (Autore, TimeStamp, …)
- Bitmap
  - Valori numerici delle intensità luminose dei punti.
  - Codifica fortemente dipendente dal formato
  - Può essere compressa in vari modi (JPEG, ZIP, LZW, …)

PNM (PBM/PGM/PPM)

- Portable Bit/Grey/Pix Map Format
- Formato NON compresso (lossless)
- Adatto per applicazioni di visione artificiale
- Facilmente editabile
- Diffuso in ambiente UNIX (Linux)
- Si converte facilmente
- Header in formato ASCII:
  - Per incorporare nell'immagine informazioni aggiuntive
  - Facilmente modificabile con editor di testo
- Contenuto in formato RAW o ASCII
- Esiste anche la versione video (PVM)

PNM Header e Bitmap

P4 = PBM
P5 = PGM
P6 = PPM

```
P5
320 240
# bla bla
255
```

Dimensioni Immagine

Informazioni aggiuntive
Es. TimeStamp, velocita'

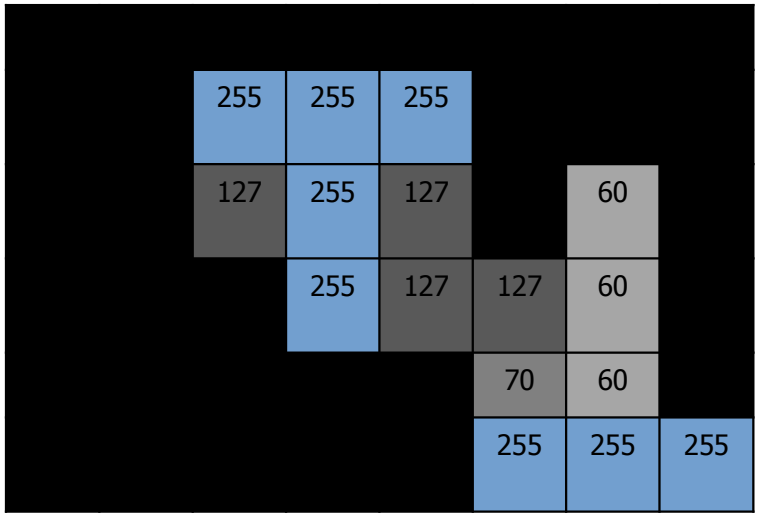
Valore massimo del colore

PBM, PGM: 1 byte per pixel
PPM: 3 byte per pixel

PGM Portable Grey Map

- Rappresentazione in memoria bitmap

0	0	0	0	0	0	0	0
0	0	255	255	255	0	0	0
0	0	127	255	127	0	60	0
0	0	0	255	127	127	60	0
0	0	0	0	0	70	60	0
0	0	0	0	0	255	255	255



Formati raster compressi

- Tipologia di compressione
- Lossless
  - Senza perdita di dati
  - TIFF, BMP, PNG, GIF, JPEG
- Lossy
  - Con perdita di dati
  - JPEG

Windows BMP Format

- Device Independent Bitmap (DIB)
- Struttura del file
- BITMAPFILEHEADER bmfh;
  - tipo, dimensione e layout (pixel ↔ lunghezza)
- BITMAPINFOHEADER bmih;
  - dimensione, tipo di compressione e formato del colore
- RGBQUAD aColors[];
  - Contiene tanti elementi quanti sono i colori nella bitmap
  - Non presente per bitmap a 24 bit di colore (24-bit red-green-blue (RGB) per rappresentare ciascun pixel)
  - I colori nella tabella sono in ordine di importanza (dithering)
- BYTE aBitmapBits[];
  - indici/intensita' di colore codificate run-length encoded (RLE)

GIF – Graphic Interchange Format

- Molto usato su web
- 8-bit (256 colori), trasparenza, animazioni
- Usa una color map (256 colori su 16M)
- Algoritmo di compressione brevettato
- Viene abbandonato in favore di PNG
- Portable Network Graphic
  - Colore a 24 bit + canale Alpha
  - Algoritmo di compressione non brevettato

PNG

- 8-bit (greyscale), 8-bit (palette), 24-bit RGB, etc.
- Gestione della trasparenza
- Header con una firma di 8-byte
  - 89 50 4E 47 0D 0A 1A 0A
- Dopo l'header è presente una serie di chunk ognuno dei quali contiene le informazioni sull'immagine

Length	Chunk type	Chunk data	CRC
4 bytes	4 bytes	Length bytes	4 bytes

JPEG File Interchange Format (JFIF)

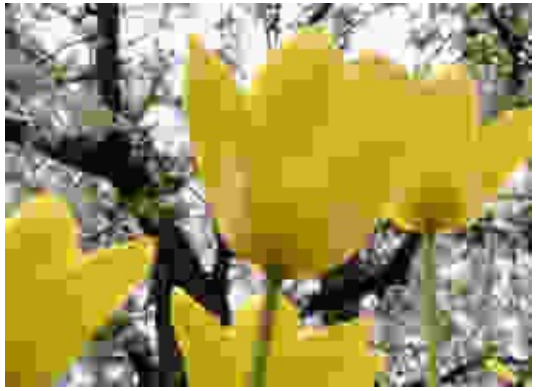
- JPEG: Joint Photographic Experts Group
- Nato alla fine degli anni 80
- E' un formato di file (diverso dall'algoritmo di compressione JPEG)
- Platform independent (PC, Mac …)
- Spazi di Colore: RGB, CMYK, YUV

Compressione JPEG

- La compressione basata su luminanza/crominanza
- I valori RGB o CMYK dei pixel vengono convertiti in Y'CbCr, uno spazio basato su luminanza/crominanza
- Compressioni separate dei due fattori
- Per il sistema visivo umano la luminanza è più importante della crominanza
- L'informazione sulla luminanza viene preservata più rispetto a quella di crominanza
- Compressione basata sulla quantizzazione della trasformata discreta di Fourier

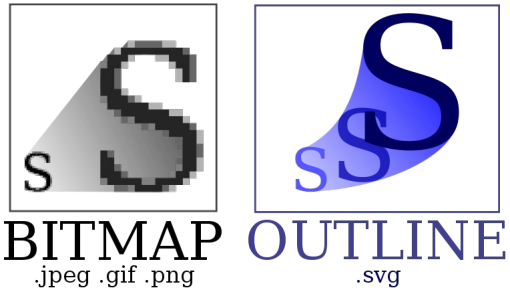
JPEG PRO e CONTRO

- Compressione Elevata:
  - $20 \rightarrow 1$ internet
  - $5 \rightarrow 1$ stampa
- Bene immagini a tono continuo
- Male immagini con pochi colori
- Poco adatto per visione (perdita, artefatti)



Scalable Vector Graphics

- Grafica vettoriale
- linguaggio basato su XML W3
- permette di avere 3 tipi di oggetti grafici:
  - forme geometriche, cioè linee costituite da segmenti di retta e curve e aree delimitate da linee chiuse;
  - immagini della grafica raster e immagini digitali;
  - testi esplicativi, eventualmente cliccabili.

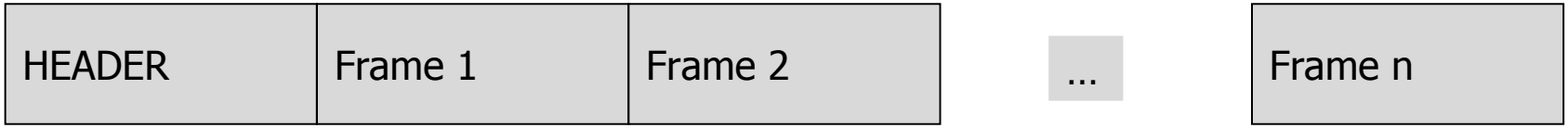


Formati video

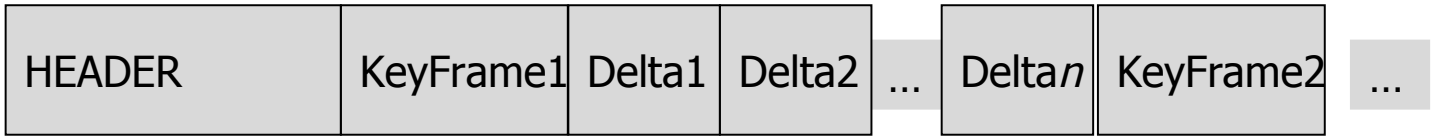
- Unico file una sequenza di immagini
- Header: informazioni sulla sequenza
  - Dimensione del frame e profondità di colore
  - Frame rate
  - Tipo di compressione
- Visione: meglio frame separati e non compressi.
  - Piu' facile operare su un singoli frame con diverse app.
  - Sequenze lunghe: File di dimensioni minori, molti file

Formati video

- Formati non compressi (PVM)



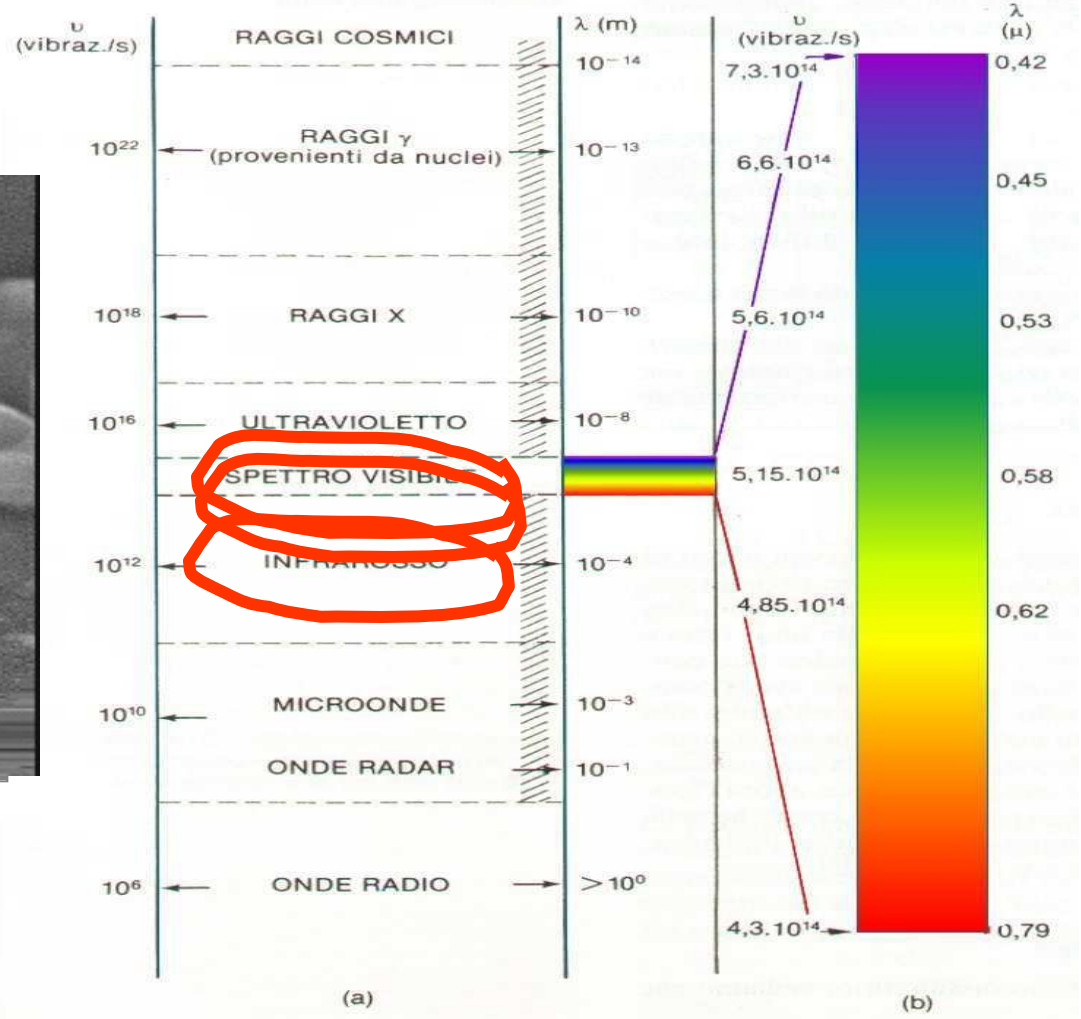
  - Possibile usare compressione intraframe
- Formati compressi interframe (MPEG)
  - Bitrate costante o variabile



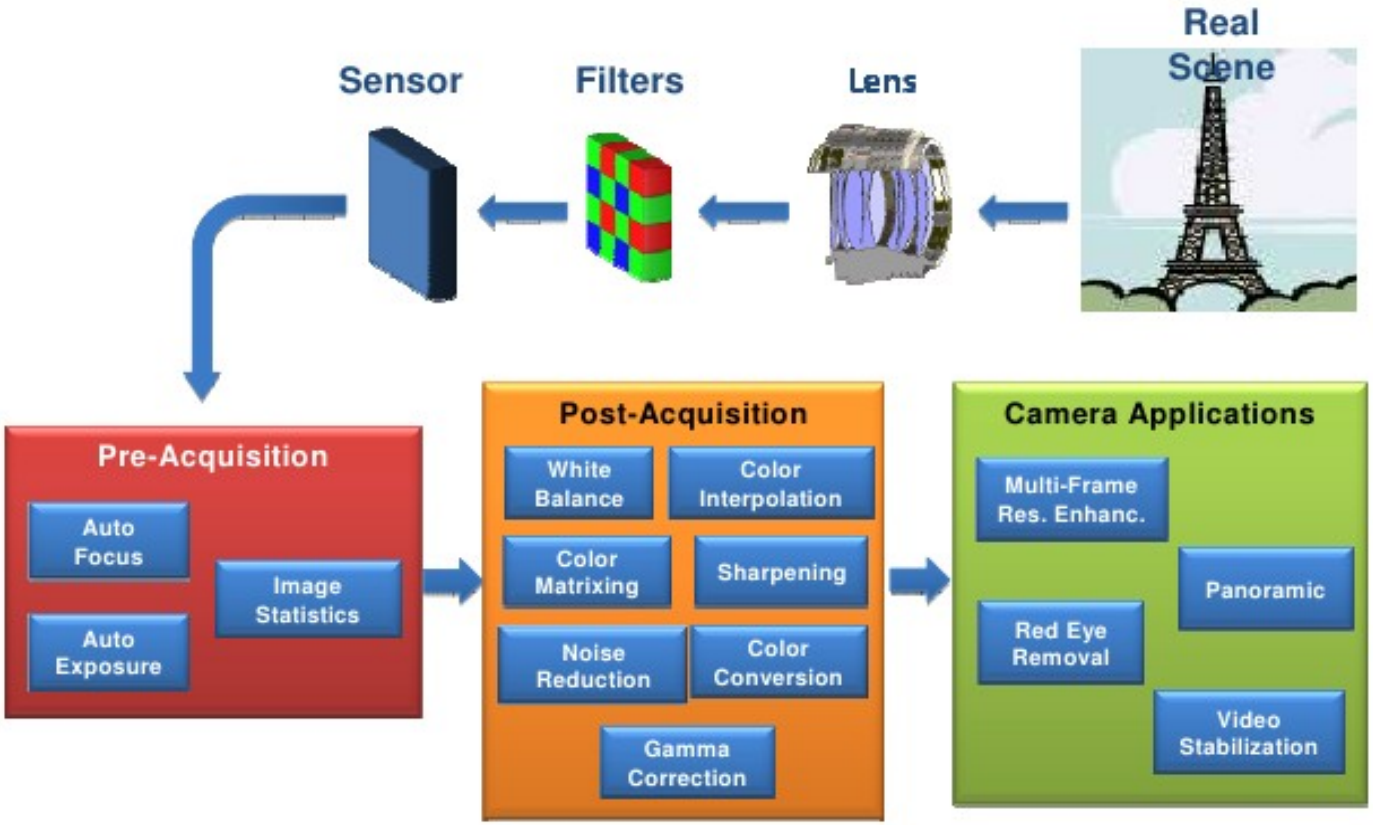
Acquisizione delle immagini

- Sistemi di acquisizione immagini
- Telecamere analogiche
- Telecamere digitali
- Smart cameras
- Schede di acquisizione video
- Interfacce di programmazione

Sensori

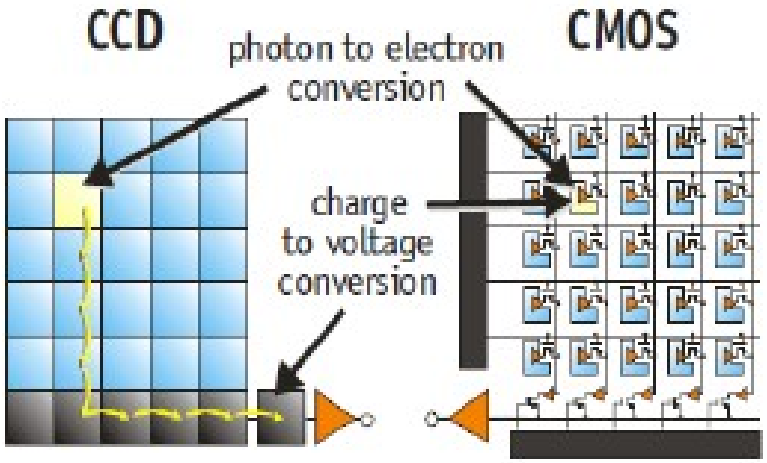

υ (vibraz./s)
λ (m)
RAGGI COSMICI
10⁻¹⁴
RAGGI γ (provenienti da nuclei)
10²²
10⁻¹³
RAGGI X
10¹⁸
10⁻¹⁰
ULTRAVIOLETTO
10¹⁶
10⁻⁸
SPETTRO VISIBILE
INFRAROSSO
10¹²
10⁻⁴
MICROONDE
10¹⁰
10⁻³
ONDE RADAR
10⁻¹
ONDE RADIO
10⁶
>10⁰
(a)
υ (vibraz./s)
λ (μ)
7,3.10¹⁴
0,42
6,6.10¹⁴
0,45
5,6.10¹⁴
0,53
5,15.10¹⁴
0,58
4,85.10¹⁴
0,62
4,3.10¹⁴
0,79
(b)

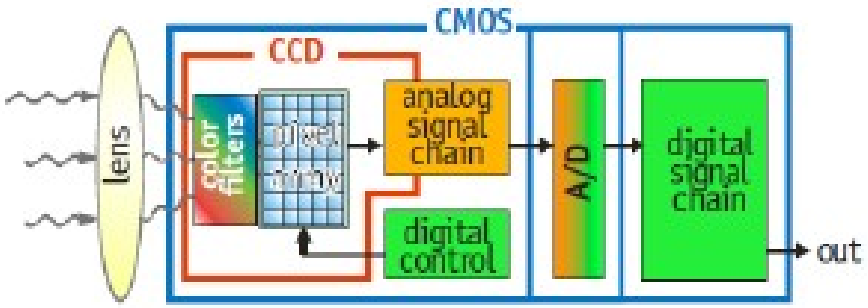
Tipica Pipeline Sensore



http://www.dmi.unict.it/~battiato/CVision1011/CVision1011.htm

CCD vs CMOS

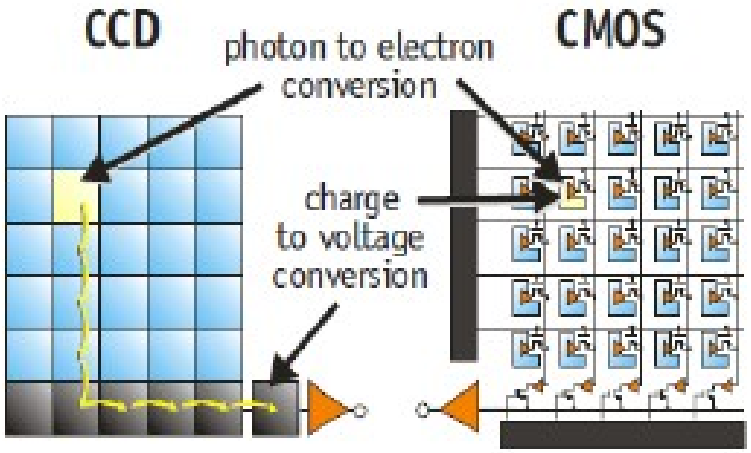


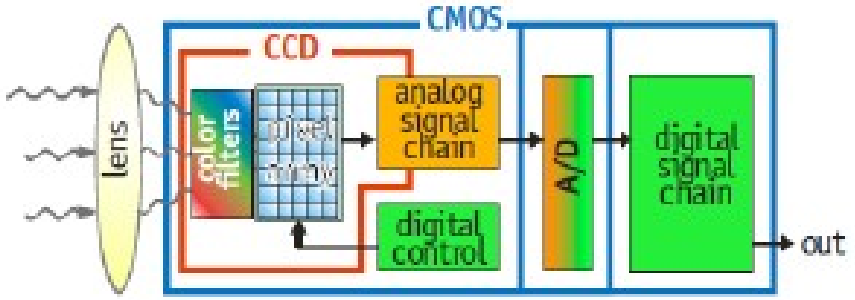


- Charge-Coupled Device:
  - Charge is actually transported across the chip and read at one corner of the array
  - Usage of a special manufacturing process to create the ability to transport charge across the chip without distortion.
  - Higher Fill Factor

http://www.dmi.unict.it/~battiato/CVision1011/CVision1011.htm

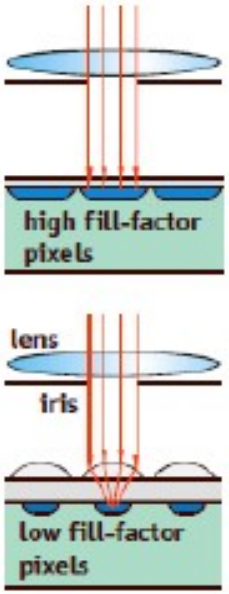
CCD vs CMOS

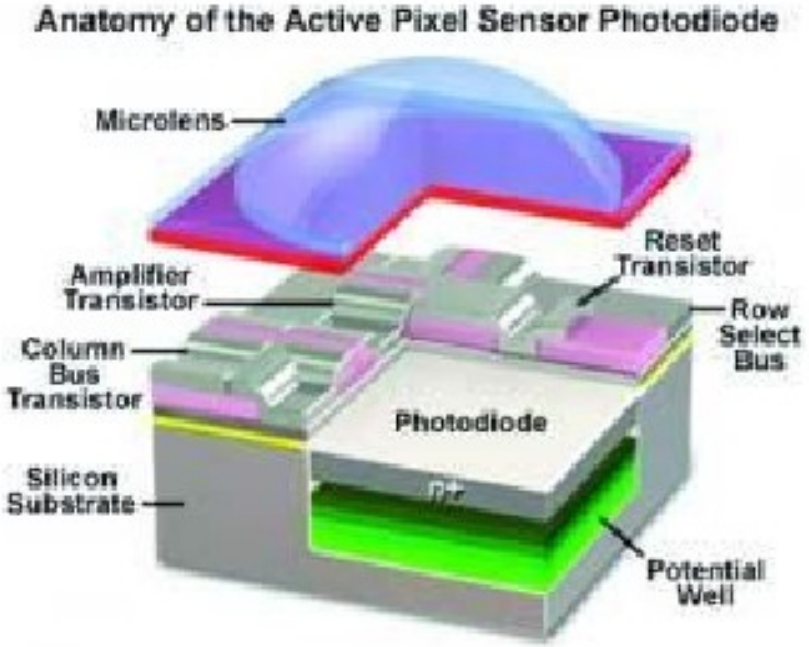




- Complimentary Metal-Oxide Semiconductor:
  - Several transistors at each pixel amplify and move the charge using more traditional wires
  - It is more flexible because each pixel can be read individually
  - Usage of the same traditional manufacturing processes to make most microprocessors.
  - Easy integration
  - Lower Fill Factor

CMOS microlenses

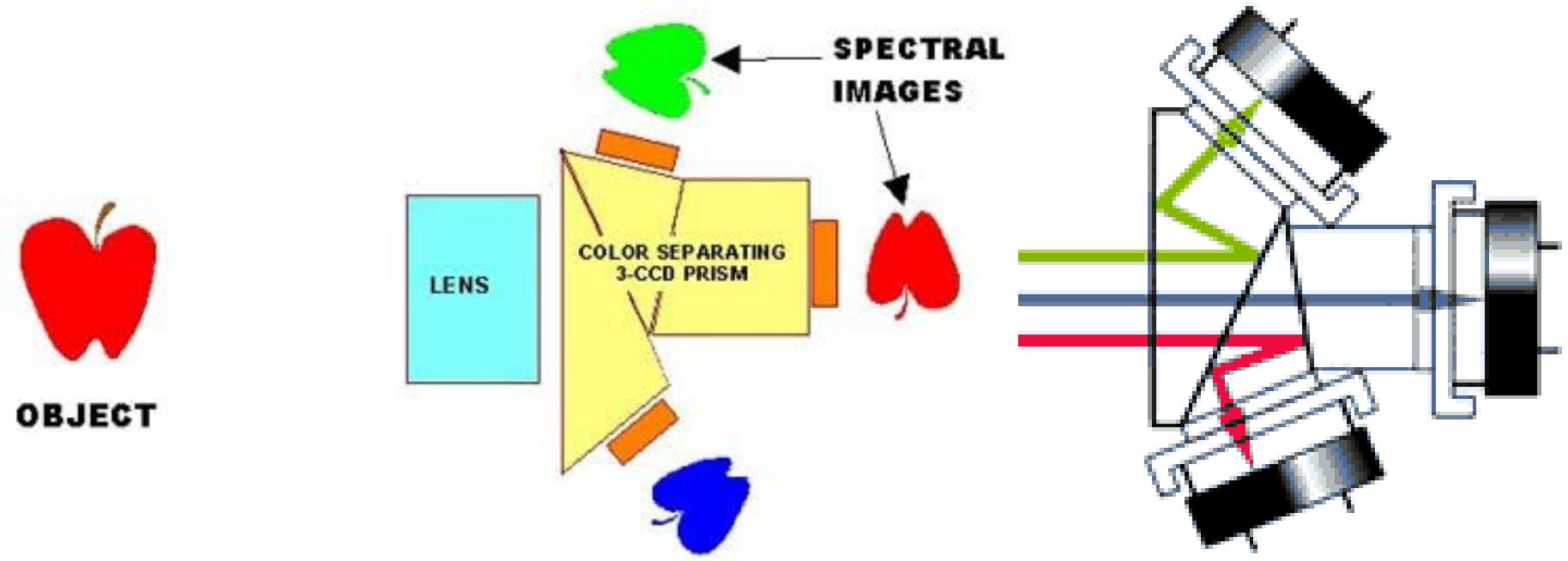




To compensate for lower fill factor (typically 30-50%), most CMOS sensors use microlenses, individual lenses deposited on the surface of each pixel to focus light on the photosensitive area. Microlenses can boost effective fill factor to approximately 70%, improving sensitivity (but not charge capacity) considerably.

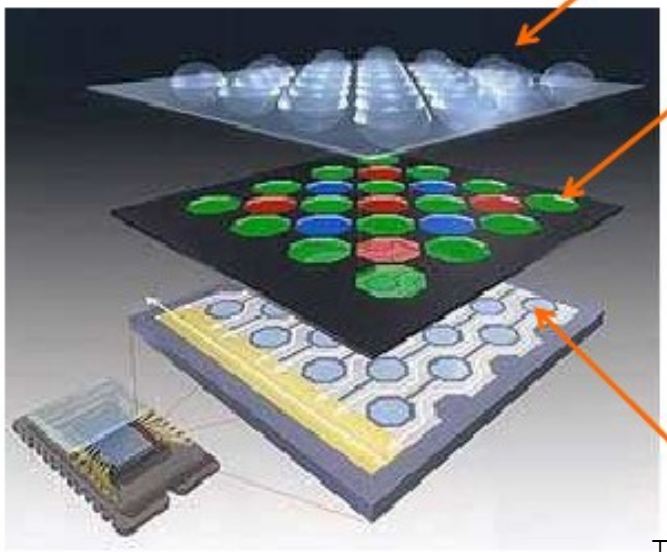
http://www.dmi.unict.it/~battiato/CVision1011/CVision1011.htm

Colore: Separazione Spettrale

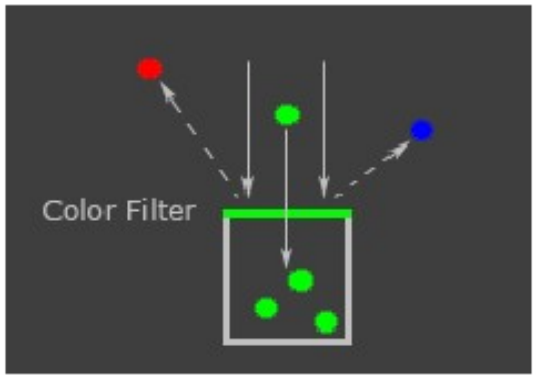


Color Field Array image sensor

Microlenti che focalizzano la luce dentro al filtro CFA



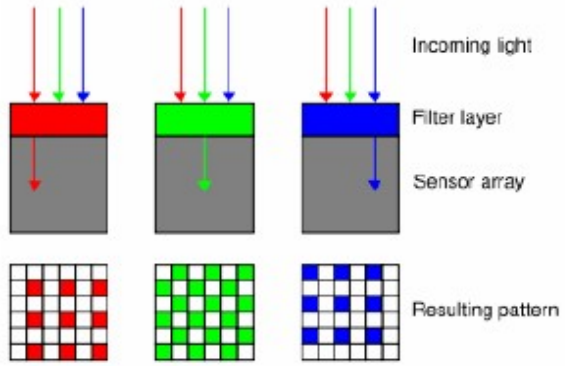
Il CFA permette il passaggio di un solo colore per volta

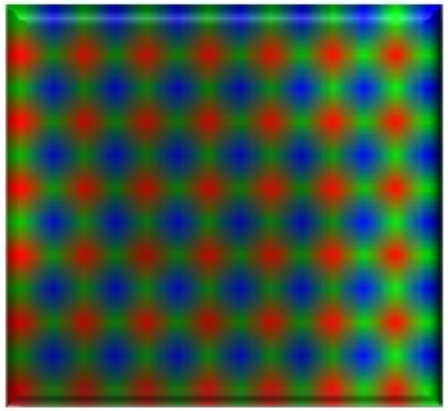


I fotorecettori accumulano gli elettroni ricevuti e il voltaggio viene trasformato in un valore numerico.

http://www.dmi.unict.it/~battiato/CVision1011/CVision1011.htm

CFA image sensor





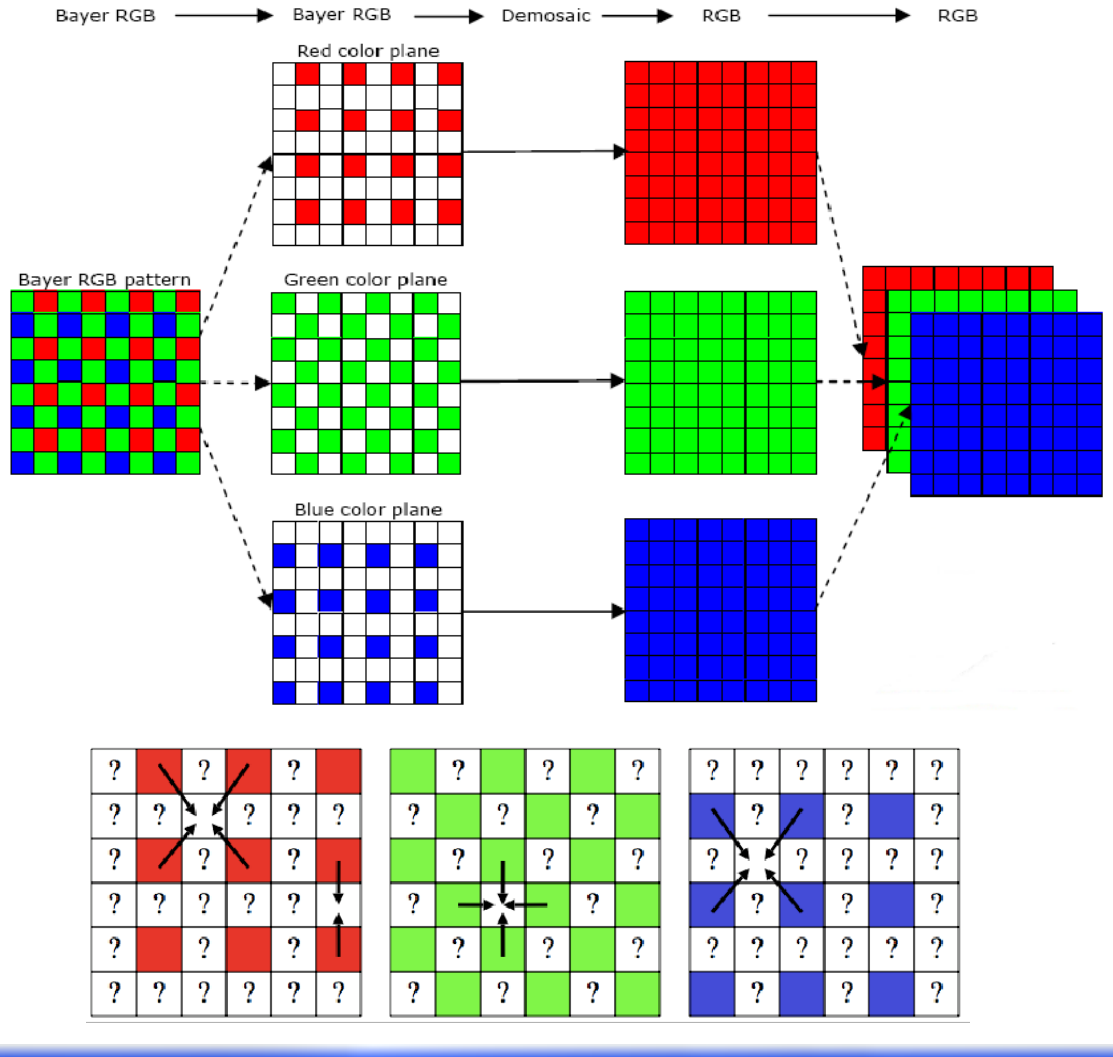
Real Scene... ...as seen by the sensor.

http://www.dmi.unict.it/~battiato/CVision1011/CVision1011.htm

Algoritmi di demosaicizzazione

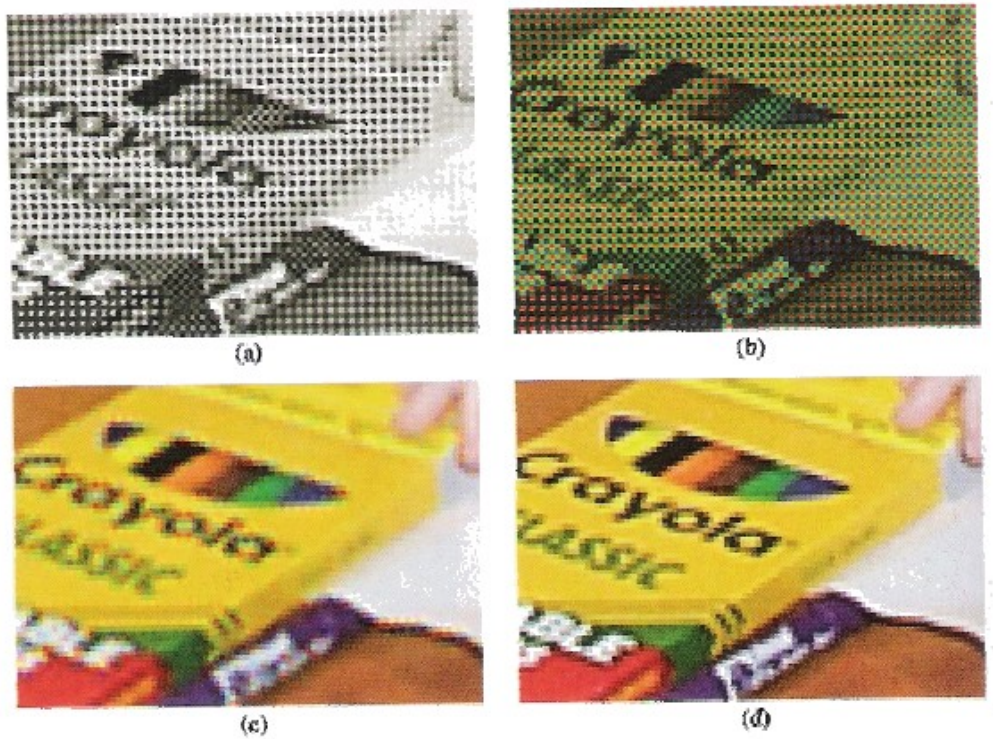
- Simple: vengono prese le coppie RGB limitrofe
- Downsample: dato un blocco 2x2 viene prodotto un singolo pixel RGB
- Edge Sensing (anche conosciuto come Edge Directed)
- …

Demosaicing linear interpolation (SIMPLE)



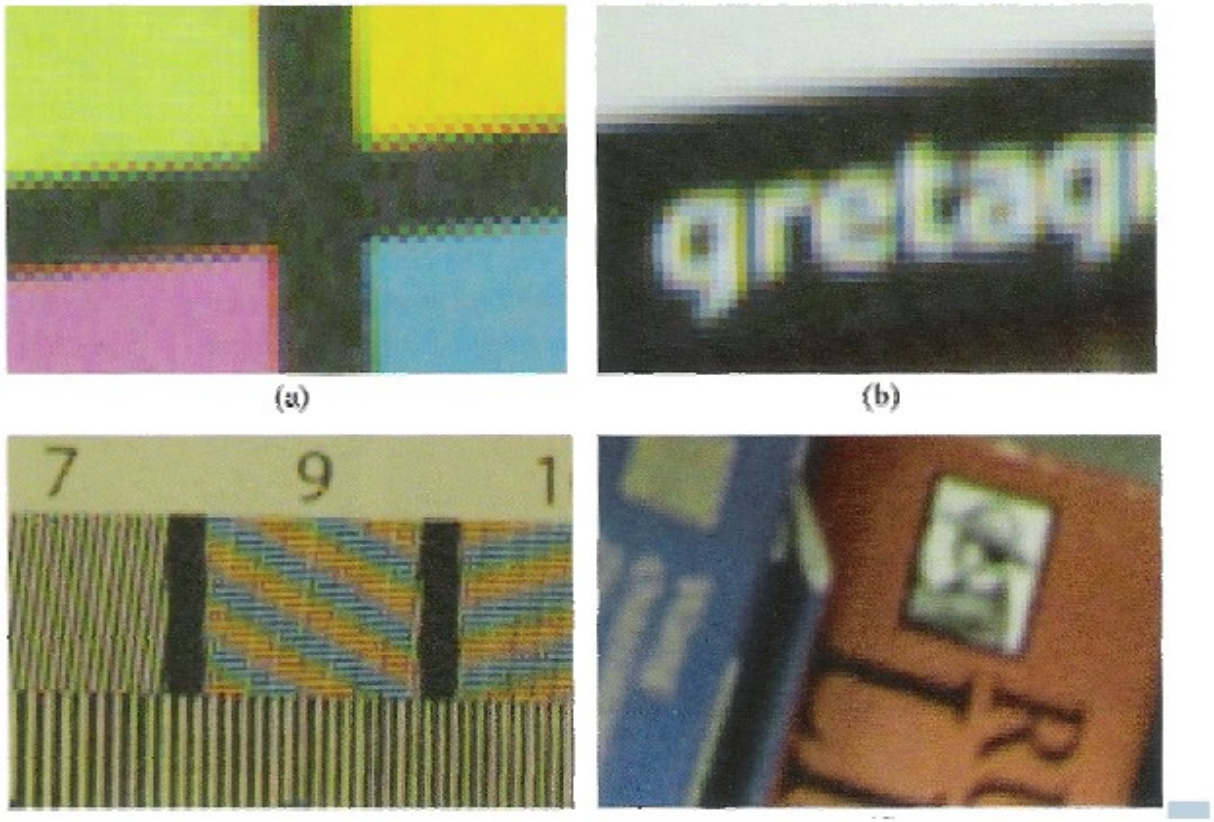
Riepilogo

a) L'immagine come esce dal sensore interpretata a toni di grigio
b) L'immagine assegnando il colore della microlente ad ogni pixel
c) Demosaicatura
d) Post Processing



http://www.dmi.unict.it/~battiato/CVision1011/CVision1011.htm

Defects



(a) zipper effects, (b) color shift, (c) aliasing artifacts and (d) blur effects.

http://www.dmi.unict.it/~battiato/CVision1011/CVision1011.htm

Pattern utilizzati

- Bayer (RGGB, BGGR, GRBG, GBRG)
- RGB+W (RGB + luminanza)
- CYGM
- RCCC (1 pixel rosso, 3 luminanza)
- RGB+NIR

Esercitazione sui formati immagine

OpenCV

OpenCV è una libreria software multipiattaforma nell'ambito della visione artificiale in tempo reale.

https://opencv.org/

Tonnellate di documentazione ed esempi online.

La useremo, con alcune limitazioni.

Da dove parto per creare una nuova applicazione?

“simple.cpp”

Semplicissimo main c++ testuale con un while che carica le immagini e le mostra

Eventuale interazione con il programma (play, stop, ecc.) da implementare.

Simple program

./simple -i image.pgm	singola immagine
./simple -i image_%03d.pgm	image_000.pgm image_001.pgm image_002.pgm
./simple -i %06d.pgm	000000.pgm 000001.pgm 000002.pgm
./simple -i %03d.pgm -t 500	attende 500ms tra 2 frame

CCMake

Configurazione manuale con ccmake (da terminale):

- portarsi nella directory di compilazione ***!= sorgente***
- *ccmake* <percorso al CMakeLists.txt principale applicazione>
- **c** per configurare (**e** per uscire dal report)
- **c** per configurare (**e** per uscire dal report), iterare fino a quando compare **g** tra i possibili comandi.
- **g** per generare il makefile ed uscire
- make

Esempio

//Dichiariamo una variabile immagine in OpenCv
cv::Mat M;

//Leggiamo un'immagine da file
M = cv::imread("image.png");

//Attenzione! Di default OpenCV apre le immagini in formato RGB!
//Se vogliamo aprire un'immagine gray scale, dobbiamo specificarlo:
M = cv::imread("image.pgm", CV_U8C1);

//Stampiamo sul terminale l'immagine
std::cout << "M = "<< std::endl << " " << M << std::endl << std::endl;

//Creiamo una finestra che chiamiamo "test"
cv::namedWindow("test",CV_WINDOW_NORMAL);

Esempio

//Creiamo una finestra che chiamiamo "test"
cv::namedWindow("test",CV_WINDOW_NORMAL);

//Visualizziamo nella finestra l'immagine
cv::imshow("test", cv::imread(M)); cv::imshow("test", M);

`cv::namedWindow("test",CV_WINDOW_NORMAL);`
`CV_WINDOW_NORMAL` = Permette di ridimensionare le finestre.
`CV_WINDOW_AUTOSIZE` = La finestra ha le dimensioni dell'immagine, ma non ne permette il ridimensionamento.

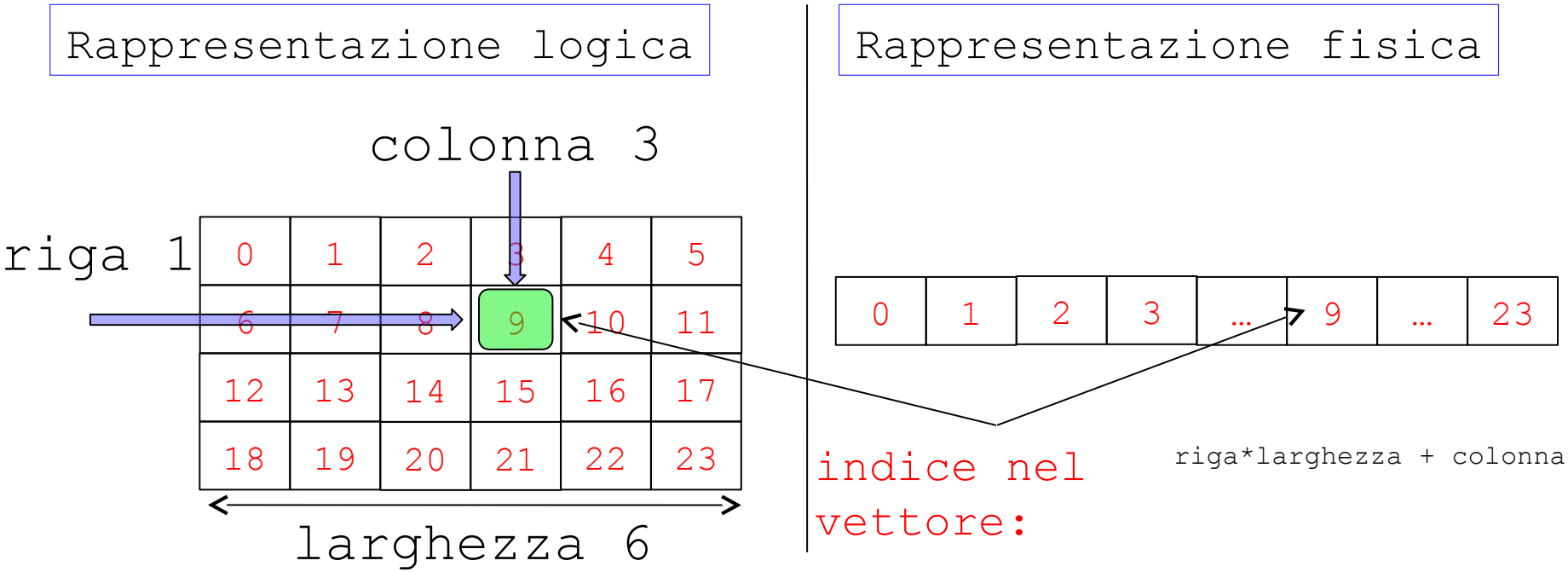
Introduzione ai buffer di memoria

Immagini - I

- Un immagine è una matrice di pixel.
- Il pixel codifica l'informazione in formato numerico (processo di campionamento).
- I tipi di pixel differiscono per:
  - numero di canali: mono(1), RGB(3), RGBA(4).
  - quantità di bit per ogni canale.

Immagini - II

- Tipicamente, per memorizzare i pixel di un immagine si usa un vettore **lineare** di pixel.



Accedere al buffer in OpenCV

```
uchar * cv::Mat::data
```

- Tramite il campo data e' possibile accedere al buffer di memoria che contiene fisicamente i pixel dell'immagine
- Restituisce il puntatore al primo byte dell'immagine

Accede al buffer in OpenCV

```
uchar * cv::Mat::ptr(int i)
```

- Tramite il metodo ptr() e' possibile accedere alla locazione di memoria che contiene la i-esima riga dell'immgine
- Restituisce il puntatore al primo byte della i-esima riga

Accede al buffer in OpenCV

```
T * cv::Mat::ptr<T>(int i)
```

- Tramite il metodo template ptr() e' possibile accedere alla locazione di memoria che contiene la i-esima riga dell'immgine qualunque sia il formato del pixel
- Restituisce il puntatore al primo pixel della i-esima riga

Esempio #1

Immagine come array semplice di unsigned char

```
for(int i =0;i<M.rows*M.cols*M.elemSize();++i)
 M.data[i] = i;
```

```
M.rows        numero di righe
M.cols        numero di colonne
M.elemSize()  dimensione in byte di ogni pixel
```

Esempio #2

Immagine come array di pixel a 3 canali di 1 byte ciascuno, RGB ad esempio

```
for(int i =0;i<M.rows*M.cols*M.elemSize();i+=M.elemSize())
 {
  M.data[i] = i;                                    //B
  M.data[i+M.elemSize1()] = i + 1;                  //G
  M.data[i+M.elemSize1()+M.elemSize1()] = i + 2; //R
 }
```

```
M.rows      numero di righe
M.cols      numero di colonne
M.elemSize  dimensione in byte di ogni pixel  (3 nel caso RGB)
M.elemSize1 dimensione in byte di ogni canale (1 nel caso RGB)
```

Esempio #4

Accesso riga/colonna per immagine a 3 canali di *1 byte ciascuno*, RGB ad esempio

```
for(int v =0;v<M.rows;++v)
  {
   for(int u=0;u<M.cols;++u)
      {
       M.data[ (u + v*M.cols)*3] = u;         //B
       M.data[ (u + v*M.cols)*3 + 1] = u+1; //G
       M.data[ (u + v*M.cols)*3 + 2] = u+2; //R
      }
  }
```

`M.rows` numero di righe

`M.cols` numero di colonne

Esempio #5

Accesso riga/colonna per immagine a multi-canale di 1 *byte ciascuno*

```
    for(int v =0;v<M.rows;++v)
      {
       for(int u=0;u<M.cols;++u)
          {
            for(int k=0;k<M.channels();++k)
              M.data[ (u + v*M.cols)*M.channels() + k] = u + k;
          }
      }
M.rows      numero di righe
M.cols      numero di colonne

M.channels() numero di canali
```

Accedere al buffer in OpenCV

```
M.at<type>(row,col)[channel]
```

- OpenCV mette a disposizione un metodo per accedere al pixel evitando l'aritmentica dei puntatori

```
M.at<cv::Vec3b>(r,c)[0] → canale B pixel r,c
```

- Nella prima parte del corso, fino al primo assegnemento, **non** la useremo.