Operazioni su Immagini e su matrici

- Un’immagine digitale raster può essere rappresentata da una matrice;
- Su una immagine possono essere fatte tutte le operazioni che si possono fare sulle matrici.
- Non è detto che tali operazioni abbiano un senso logico. Ad esempio che vuol dire moltiplicare due immagini da un punto di vista visivo?
- Qual è il range dei valori dopo tali operazioni?

Prodotto

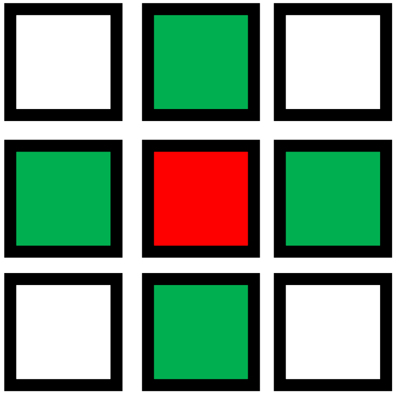
- ATTENZIONE: per le matrici vale la regola del prodotto riga per colonna, mentre nell'image processing si usa fare il prodotto puntuale tra due matrici, cioè il prodotto punto a punto degli elementi corrispondenti.

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} (1\times 3+0\times 2+2\times 1) & (1\times 1+0\times 1+2\times 0) \\ (-1\times 3+3\times 2+1\times 1) & (-1\times 1+3\times 1+1\times 0) \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$$

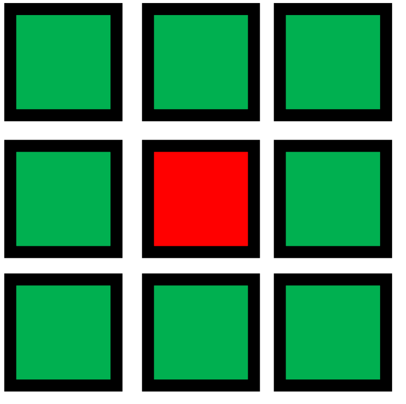
$$\begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix} \cdot \begin{bmatrix} -3 & 0 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} -3 & 0 \\ 3 & -4 \end{bmatrix}$$

Neighborhood N_p

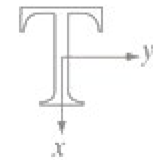
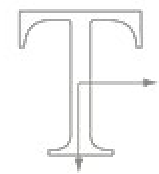
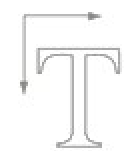
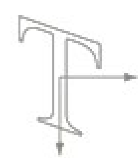
- I vicini 4 connessi di un dato pixel sono quelli alla sua destra e sinistra e quelli sopra e sotto.



- I vicini 8 connessi sono quelli 4 connessi a cui si aggiungono i 4 pixel in diagonale.



Operazioni affini

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

Forward mapping

- Dove (v,w) è il pixel di input, (x,y) quello di output e T la matrice affine.
- Per ottenere il valore

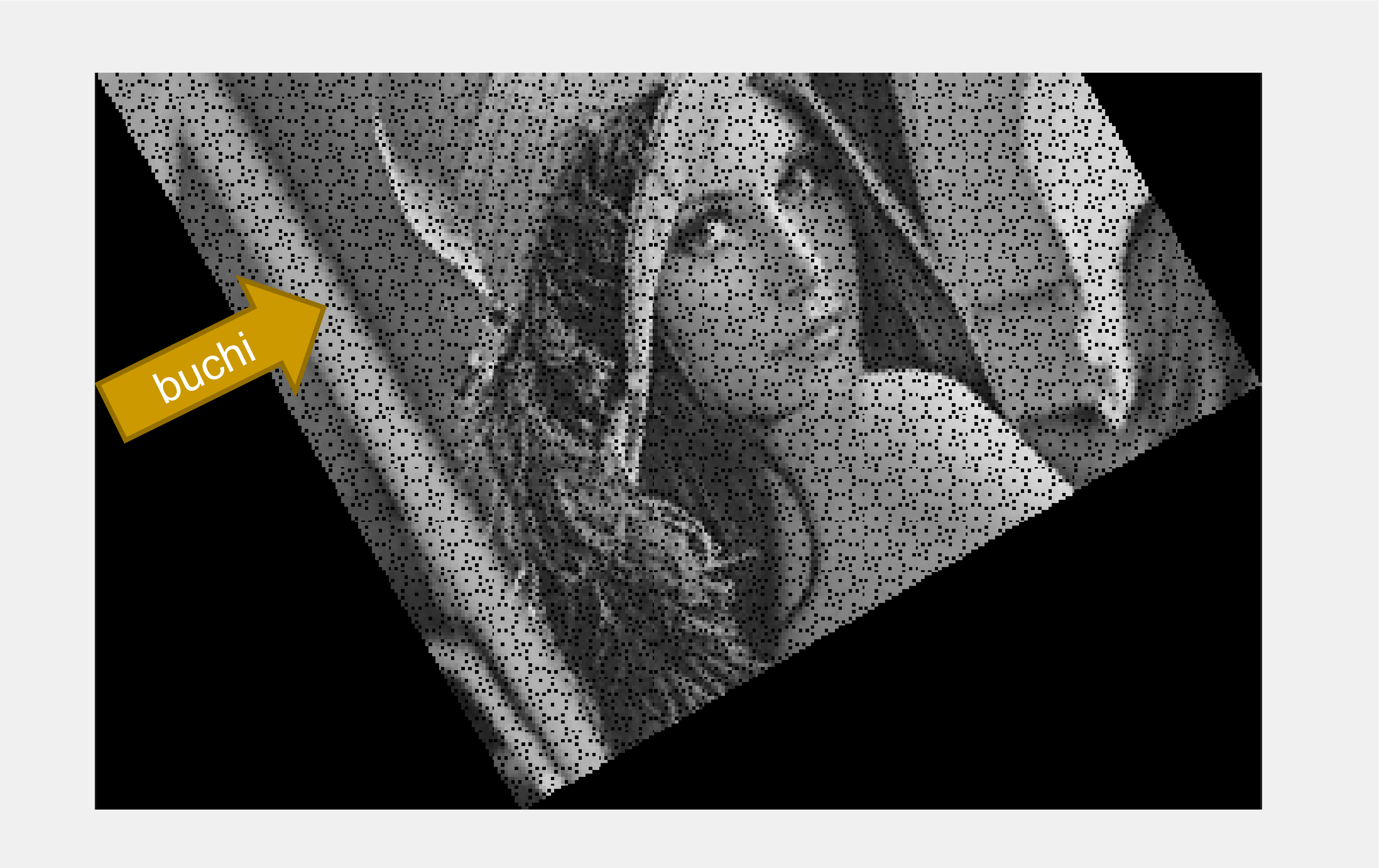
$$[x\ y\ 1] = [v\ w\ 1] * T$$

- In questo caso si fa scorrere l'immagine di input e per ogni pixel (v,w) si calcola la posizione della nuova immagine (x,y)

Forward mapping

```
A=rgb2gray(imread('lena.jpg'));
A=double(A);
figure,imshow(uint8(A));
[m,n]=size(A);
theta=-45;

B=zeros(size(A));
T=[cosd(theta) sind(theta) 0; -sind(theta) cosd(theta) 0; 0 0 1];
 %scorre l'immagine di input e si stabilisce in quale punto finiranno i
 %nostri pixel in output

for v=1:m
    for w=1:n
        vett=round([v w 1]*T);
        x=vett(1);
        y=vett(2);
        if (x>0 & x<=m) & (y>0 & y<=n)
            B(x,y)=A(v,w);
        end
    end
end

figure,imshow(uint8(B));
```

Forward mapping (rotazione)



Forward mapping (scaling)

- **Forward mapping**
- In questo caso si fa scorrere l’immagine di input e per ogni pixel (v,w) si calcola la posizione della nuova immagine (x,y)
- **Inverse mapping**
- Visita le posizioni spaziali dei pixel di output (x,y) e per ciascuna di esse calcola le corrispondenti coordinate nell’immagine di input (si ha una formula inversa)

Inverse mapping

- Visita le posizioni spaziali dei pixel di output (x,y) e per ciascuna di esse calcola le corrispondenti coordinate nell'immagine di input (si ha una formula inversa)

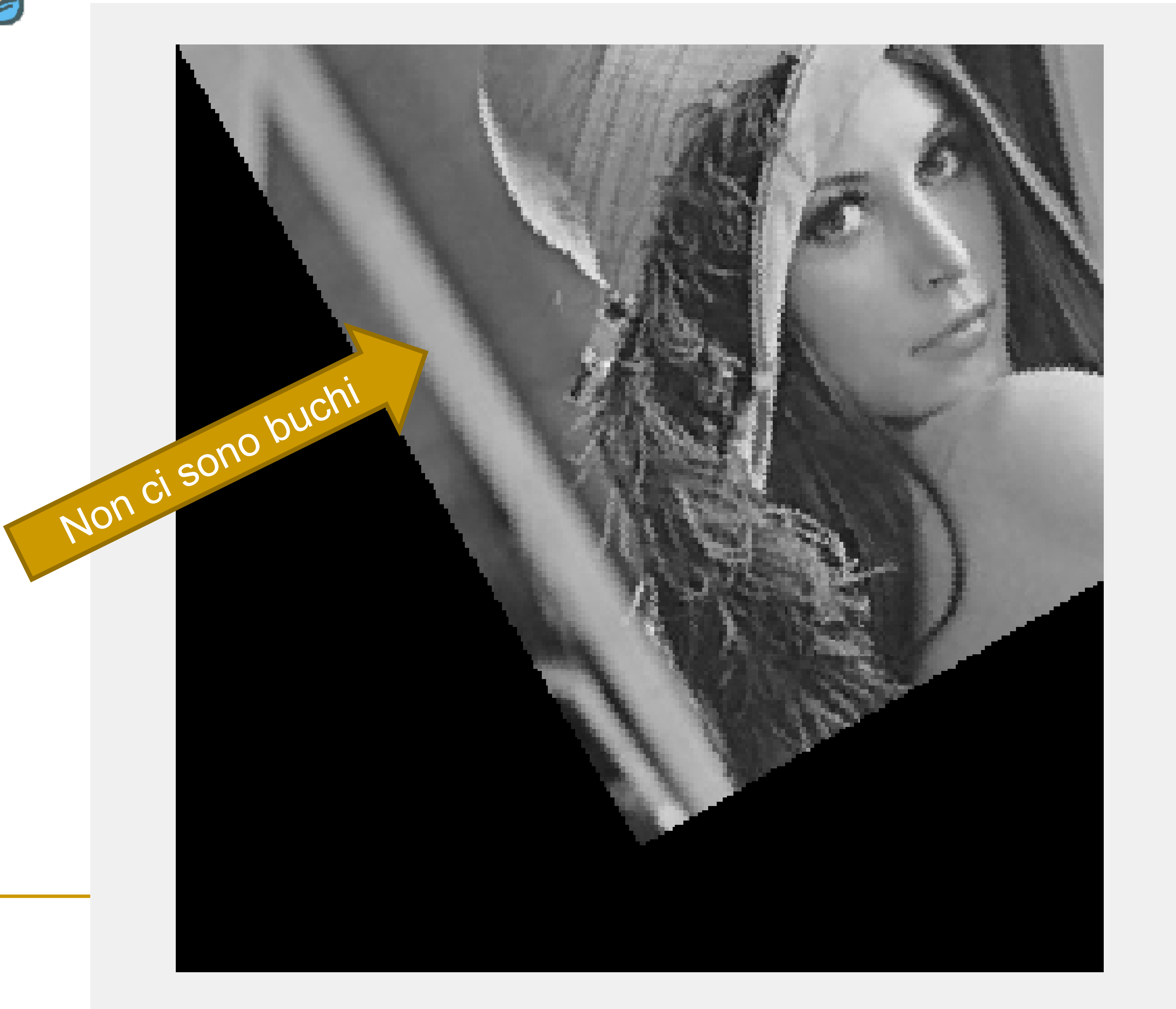
- Ovviamente

$$[v\ w\ 1] = [x\ y\ 1] * \text{inversa}(T)$$

Inverse mapping (usato da Matlab)

```
A=rgb2gray(imread('lena.jpg'));
A=double(A);
figure,imshow(uint8(A));
[m,n]=size(A);
theta=38;

B=zeros(size(A));
T=[cosd(theta) sind(theta) 0; -sind(theta) cosd(theta) 0; 0 0 1];

%scorre l'immagine di output

for x=1:m
    for y=1:n
        vett=round([x y 1]*inv(T));
        v=vett(1);
        w=vett(2);
        if (v>0 & v<=m) & (w>0 & w<=n)
            B(x,y)=A(v,w);
        end
    end
end

figure,imshow(uint8(B));
```

Inverse mapping (rotazione)



Inverse mapping (scaling)

Non ci sono buchi

Combinazioni

- Inoltre le trasformazioni affini si possono combinare tra di loro semplicemente moltiplicando le corrispondenti matrici T.

```
theta=38;
tx=40;
ty=23;
cx=2;
cy=2;

Tr=[cosd(theta) sind(theta) 0; -sind(theta) cosd(theta) 0; 0 0 1];
Tt=[1 0 0; 0 1 0; tx ty 1];
Ts=[cx 0 0 ; 0 cy 0; 0 0 1];
T=Tr*Tt*Ts;
```

L’interpolazione

Valori non assegnati

- Nel corso delle trasformazioni, potrebbero esserci dei valori di pixel che non sono mai individuati dalle formule.

- Per essi si applica un processo di interpolazione.

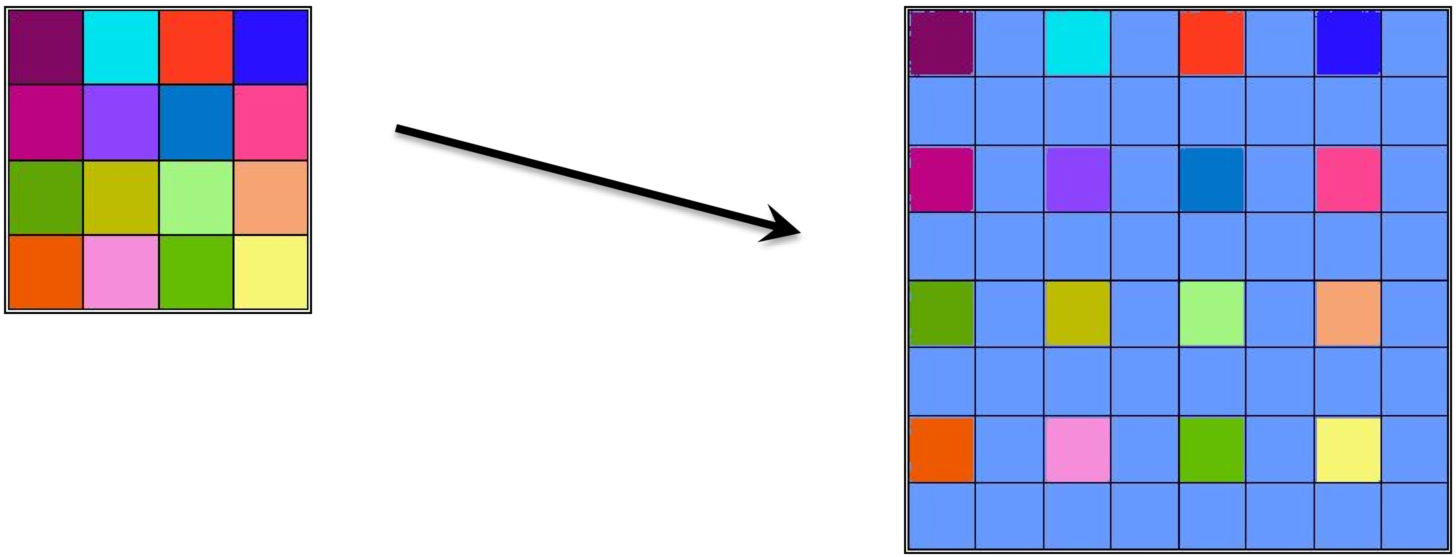
Interpolazione

- In generale, l'interpolazione è il processo che partendo da dati reali stima i dati non conosciuti.
- L'interpolazione non comporta un miglioramento della qualità dell'immagine come se "riacquisisse" i valori mancanti ma effettua solo una stima dei valori ignoti.

Zooming in

- L'interpolazione è effettuata anche nei processi di zooming.
- Uno zooming 2× vuol dire che le dimensioni dell'immagine sono raddoppiate. Se ho una immagine $m \times n$ essa diverrà $2m \times 2n$. Che vuol dire che il numero totale di pixel sarà quadruplicato!
- Lo zooming può essere effettuato in maniera differente per le singole dimensioni. Ad esempio si potrebbe raddoppiare il numero di righe e triplicare il numero di colonne. In questo caso, i nostri algoritmi di interpolazione saranno più complicati.

Zooming in (2x)



Dopo aver posizionato i valori già noti, occorre stimare i valori nelle zone vuote.

Vari tipi di interpolazione

Esistono diversi tipi di interpolazione:

- Nearest neighbor (o replication)
- Bilinear
- Bicubic
- Altri…

Replication

(a)

(b)

Replication o nearest neighbor

- Questo metodo assegna a ogni nuova posizione l'intensità del pixel più prossimo nell'immagine originale.
- Questo approccio è molto semplice ma introduce artefatti come distorsioni lungo i lati degli oggetti rappresentati nell'immagine.

Bilinear

(a)

(b)

Bilinear

- Nell'interpolazione bilineare si utilizzano i quattro pixel più vicini per stimare l'intensità da assegnare a ciascuna nuova posizione. Supponiamo che (x, y) siano le coordinate della posizione cui si deve assegnare un valore di intensità e che v(x, y) equivalga al valore dell'intensità. Per l'interpolazione bilineare il valore assegnato si ottiene mediante l'equazione

  $$\mathbf{v(x, y) = ax + by + cxy + d}$$

  Dove i quattro coefficienti sono determinati a partire dalle quattro equazioni nelle quattro incognite ottenibili utilizzando i quattro pixel più vicini al punto (x, y).
- L'interpolazione bilineare produce dei risultati migliori rispetto alla replication con un incremento modesto nella complessità di calcolo.

Bicubic

(a)

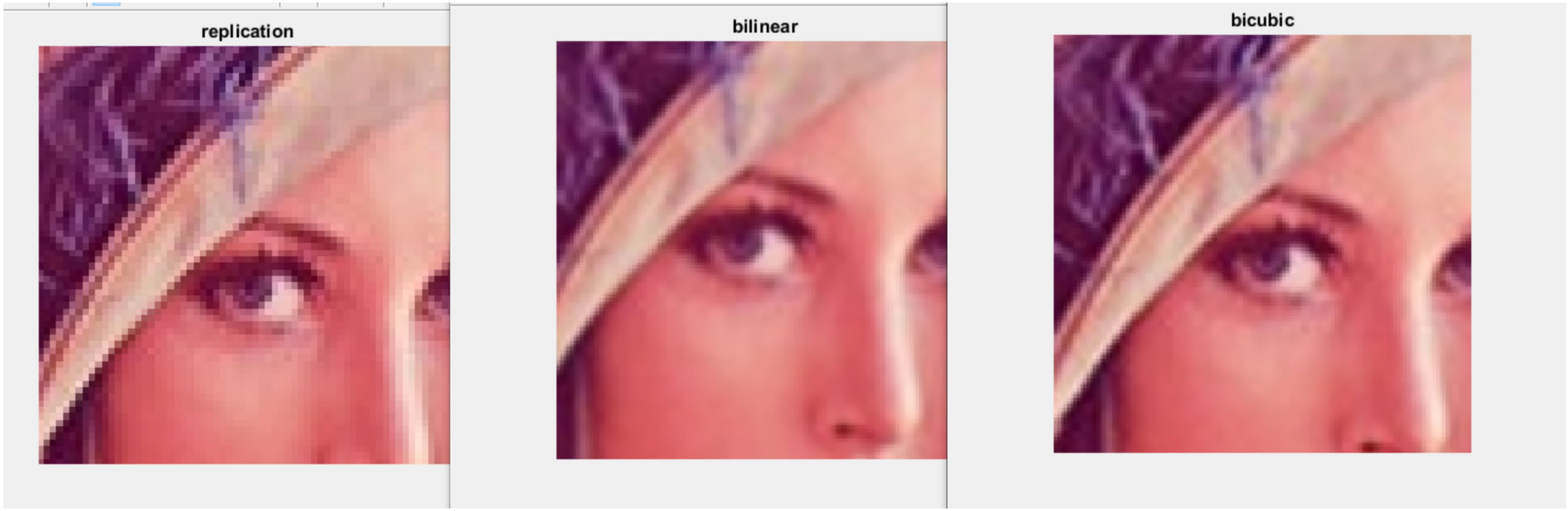
(b)

Bicubic

- L’interpolazione bicubica utilizza i sedici pixel più vicini al punto. Il valore di intensità assegnato al punto (x, y) si ottiene attraverso l’equazione

$$v(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} x^i y^j$$

- Dove i sedici coefficienti sono determinati a partire da sedici equazioni in sedici incognite che possono essere scritte utilizzando i sedici punti più vicini a (x, y).
- Generalmente l’interpolazione bicubica preserva meglio i dettagli rispetto all’interpolazione bilineare. L’interpolazione bicubica è la tecnica standard utilizzata nei programmi commerciali di editing come Adobe Photoshop e Corel Photopaint.


replication
bilinear
bicubic

Cosa fare ai bordi?

Un problema è quello dei bordi: come fare l'interpolazione ai bordi?

POSSIBILI SOLUZIONI:

- Non fare nulla
- Interpolare con i valori presenti anche se in numero minore di quelli usati per altri pixel.

Considerare solo le zone centrali dell’immagine.

- In questo caso non si calcolano i valori ai bordi

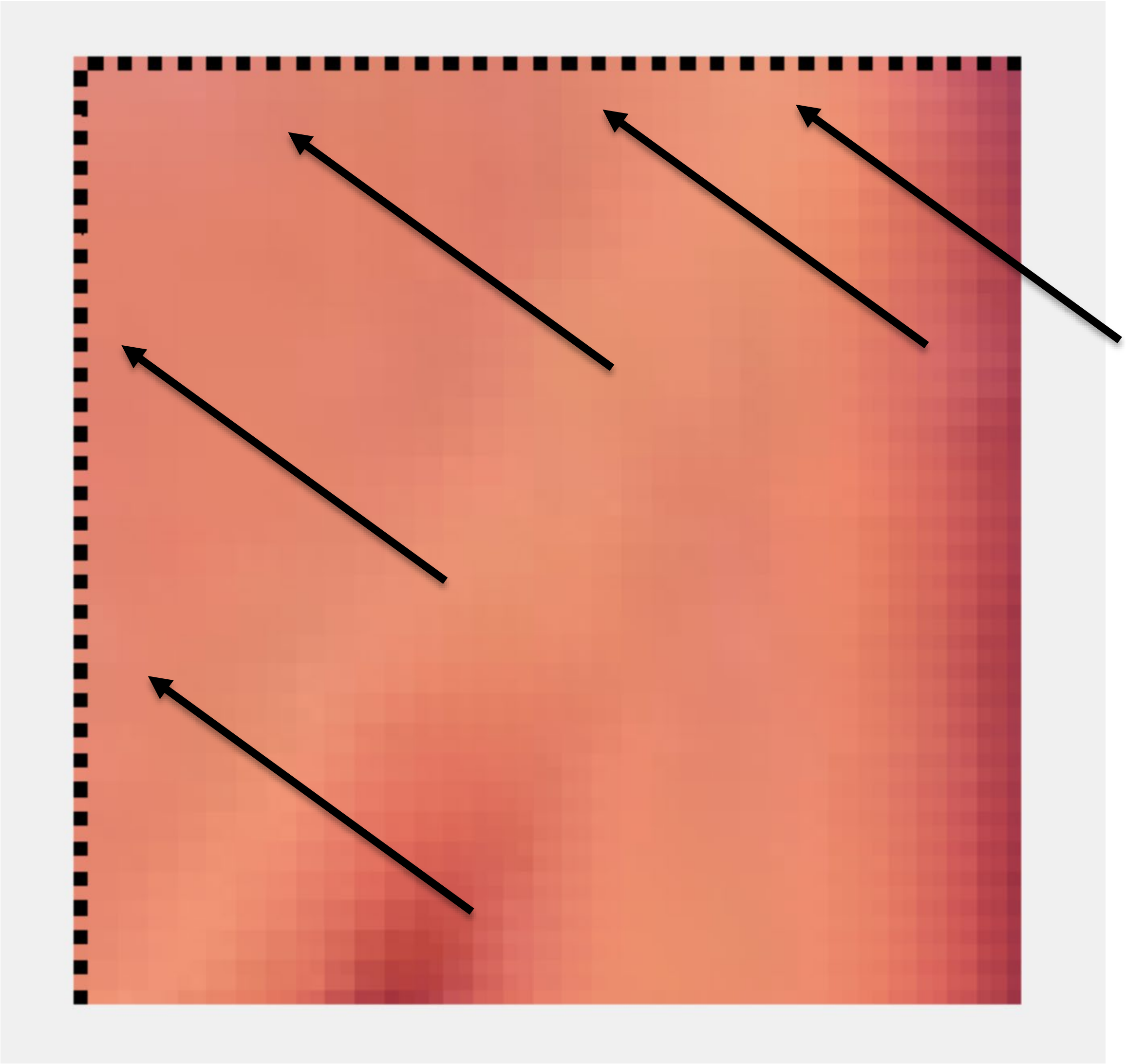
input

1	2	3
4	5	6
7	8	9

output

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

31



Fare i calcoli solo con i valori presenti nell'intorno.

- In questo caso per fare i calcoli si usano solo i valori che ci sono (valori in nero e in rosso).

input

1	2	3
4	5	6
7	8	9

output

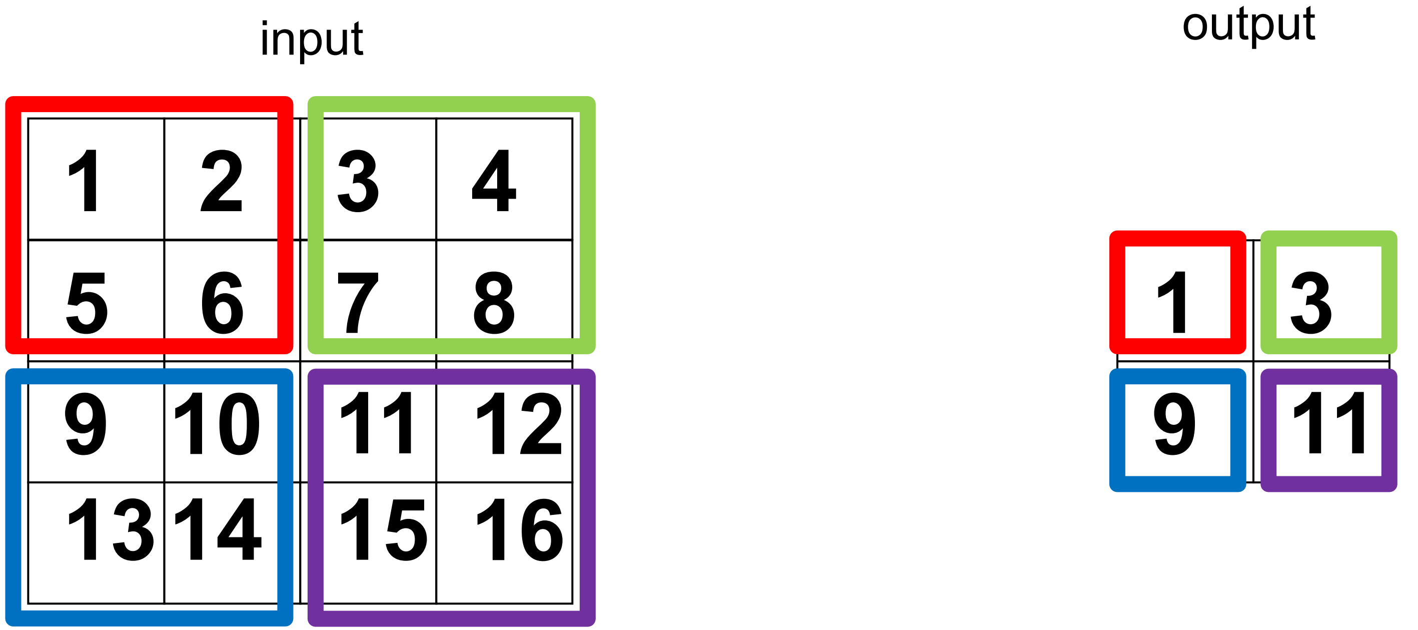
1	2	2	3	3	3
3	3	3	4	4	4
4	4	5	5	6	6
6	6	6	7	7	7
7	7	8	8	9	9

Zooming out

- Se lo zooming è fatto con un numero inferiore ad 1, si ottiene una immagine più piccola dell’originale.
- Se riduco una immagine, ho un processo detto di «decimazione».
- Data una immagine mxn con uno zooming out di 0,5 si otterrà una immagine $m/2 \times n/2$.

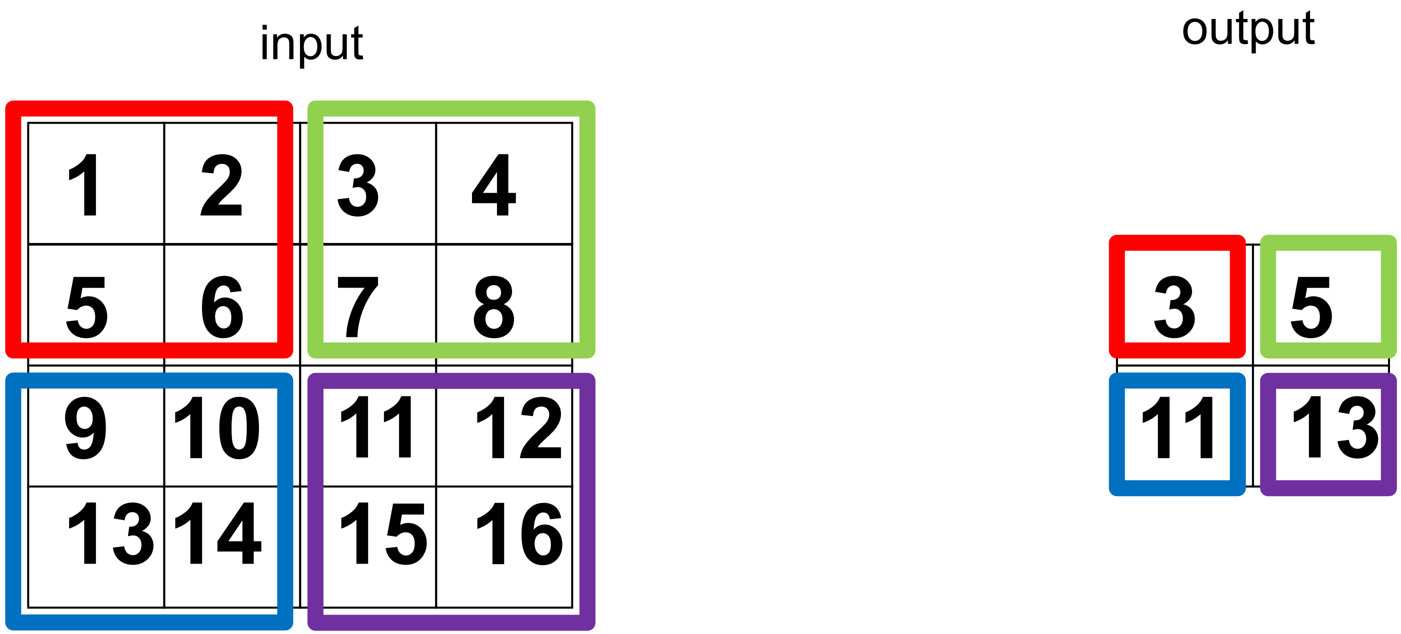
Decimazione: metodo 1

- Ogni quattro pixel se ne sceglie uno.



Decimazione: metodo 2

- Di quattro pixel se ne calcola il valore medio.



Stima della qualità di un algoritmo

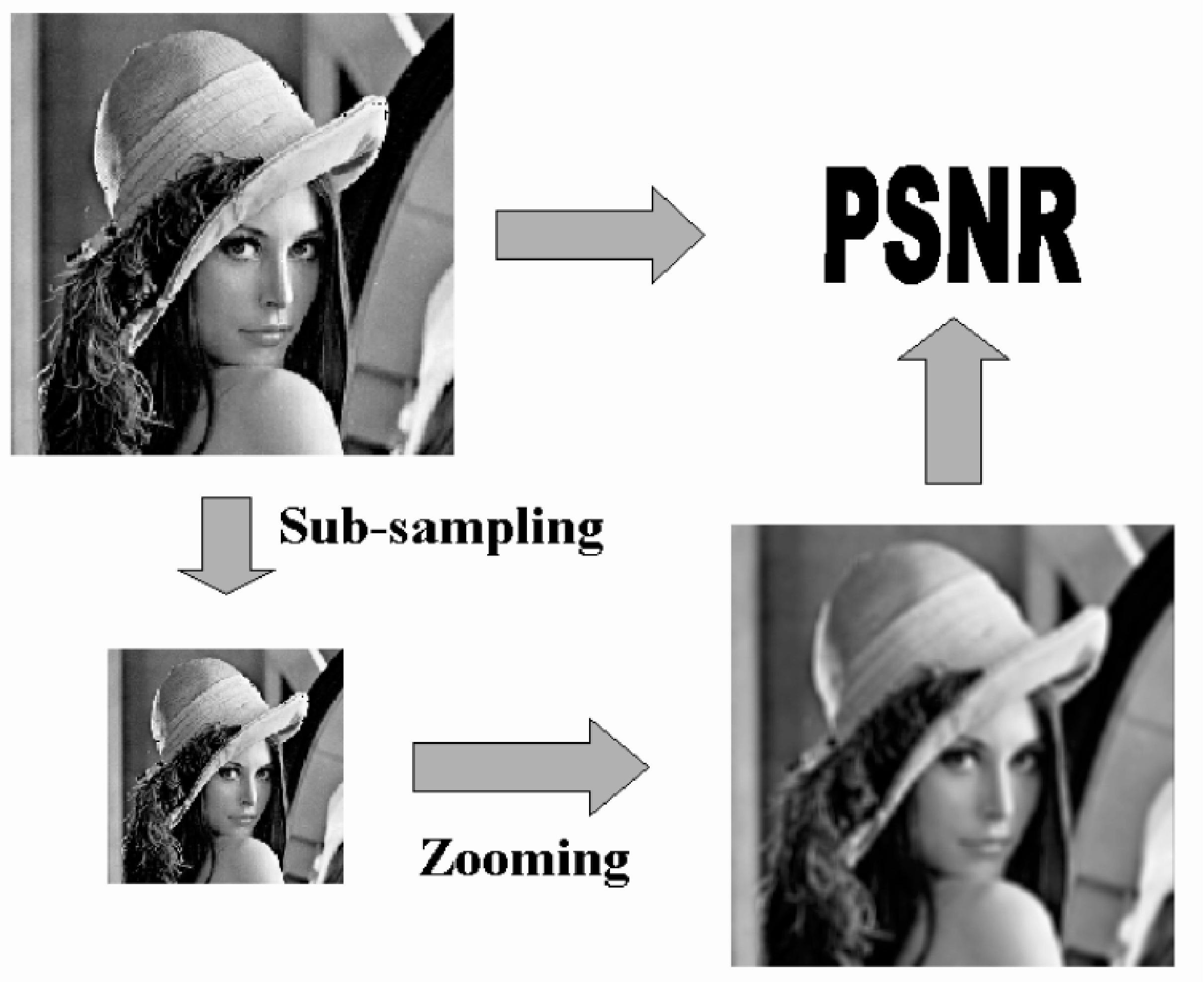
MSE: (*Mean Square Error*) tale parametro serve a stimare l'errore quadratico medio tra due immagini; più tale indice è basso minore è la differenza tra le immagini.

PSNR: (*Peak Signal to NoiseRatio*) parametro per misurare la qualità di un immagine compressa rispetto all'originale, dipende dalla differenza tra l’immagine codificata e quella originale. Maggiore è il suo valore maggiore sarà la “somiglianza” con l'originale.

PSNR (Peak Signal to Noise Ratio)

- È una tecnica Full reference.
- Per calcolarlo è necessario avere sia l'immagine da valutare I' (MxN) che una sua versione ottimale I (MxN).
- Il PSNR non è il migliore parametro per valutare la qualità di un algoritmo di interpolazione, ma è il più diffuso.

Schema per il calcolo del PSNR in caso di zooming



PSNR: formule

- Per calcolare il PSNR abbiamo bisogno dell'MSE (Mean Square Error):

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

- Il PSNR è calcolato con una delle seguenti formule (sono equivalenti):

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

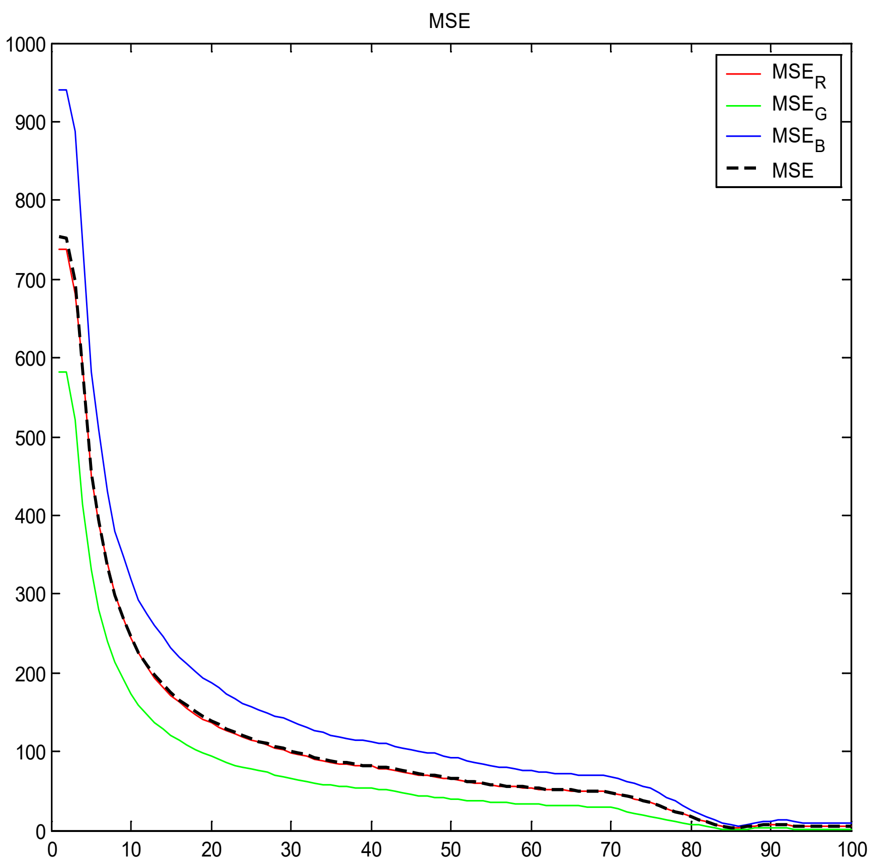
S is the maximum pixel value (usually 255),

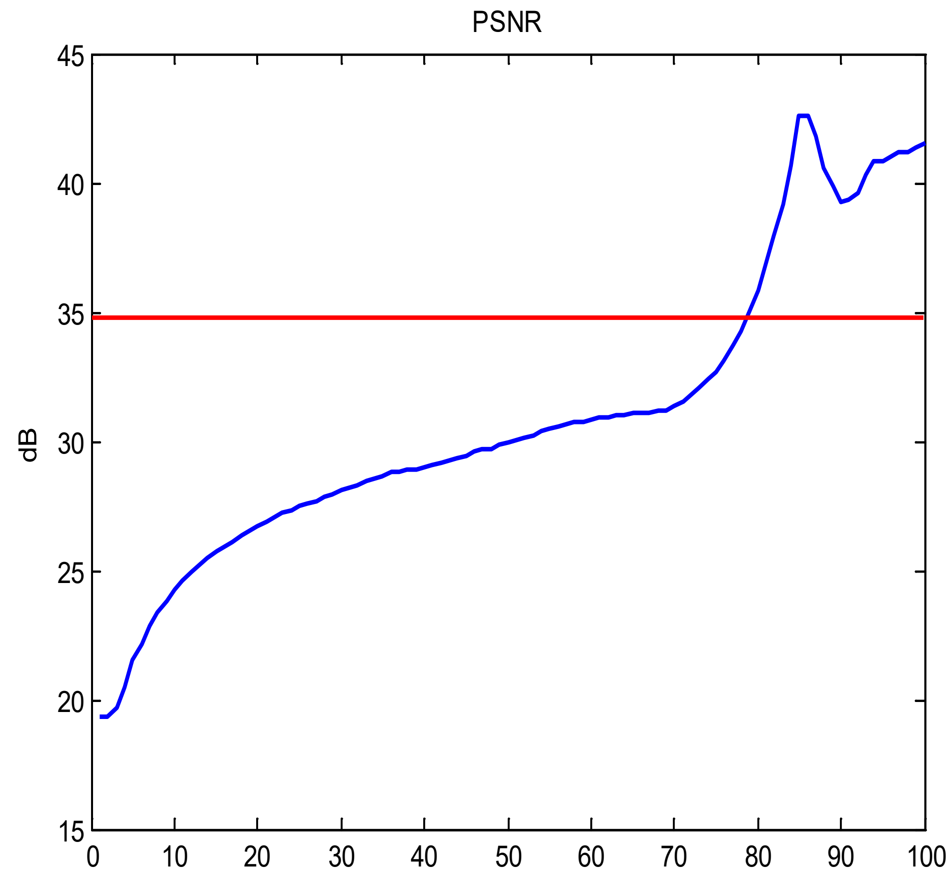
MSE e PSNR

MSE e PSNR sono molto usati perché semplici da calcolare, però non sempre danno un risultato fedele a quello dato dal sistema visivo umano. Infatti:

- La sensibilità del sistema HVS agli errori può essere diversa per diversi tipi di errori, e può variare anche in base al contesto visuale. Tale differenza non può essere colta adeguatamente dall'MSE.
- Due immagini distorte possono avere tipi molto diversi di errori pur avendo lo stesso MSE.
- Entrambe le metriche sono fortemente influenzate anche da "impercettibili" movimenti spaziali (traslazioni,rotazioni, flipping di righe e/colonne)

PSNR vs MSE

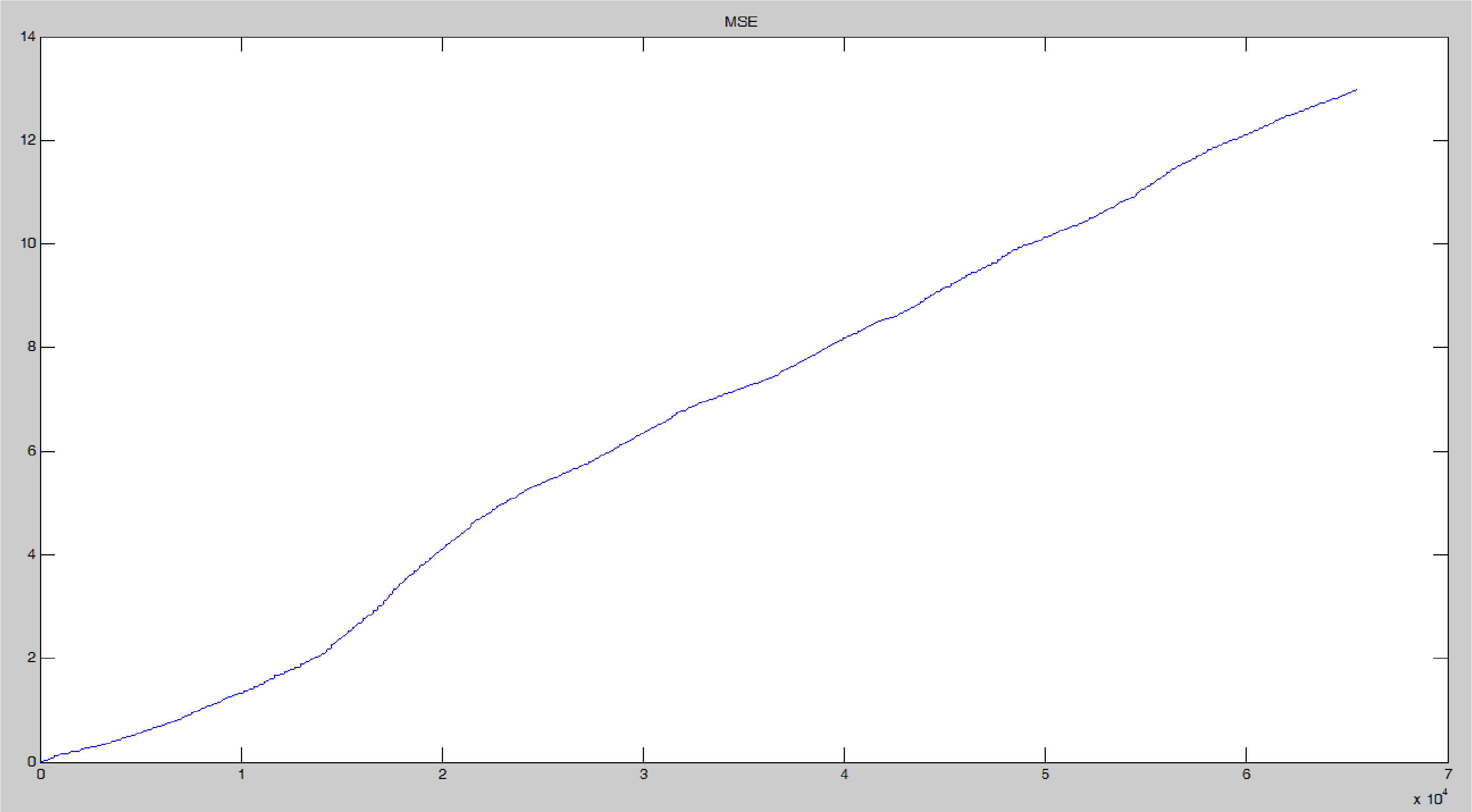

MSE
1000
900
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
MSE_R
MSE_G
MSE_B
MSE


PSNR
dB
45
40
35
30
25
20
15
0
10
20
30
40
50
60
70
80
90
100

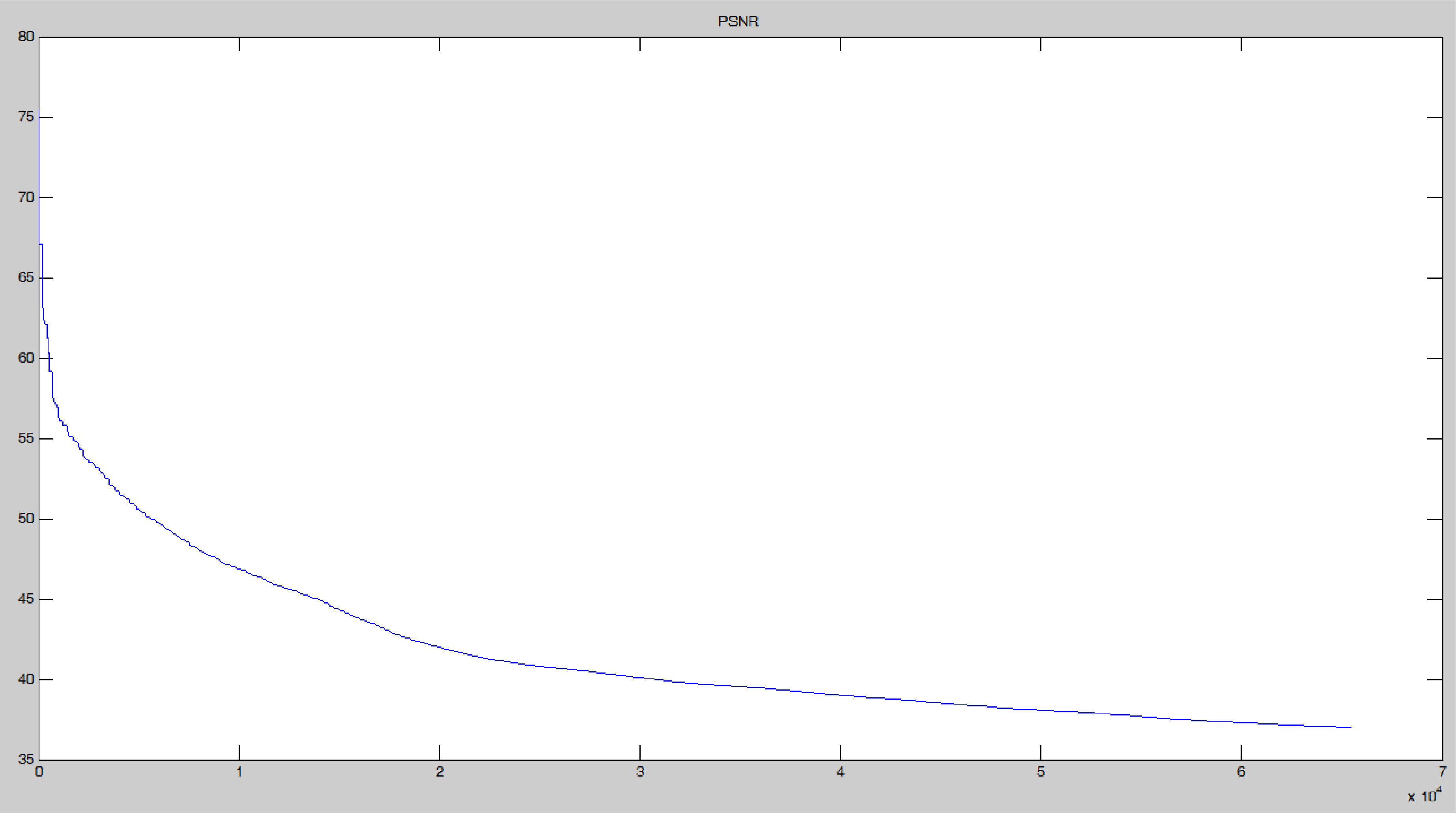
Andamento grafico

- Ecco come varia il PSNR e MSE se facciamo variare in maniera random i pixel di una immagine.
- Ogni volta che varia un pixel calcolo il PSNR e MSE.

MSE



PSNR



MSE e PSNR per immagini RGB

- Solitamente si usa una delle seguenti soluzioni
  - la semplice media dei valori MSE (o PSNR) sui 3 canali
  - Una combinazione lineare che pesa maggiormente la componente verde
  - …