

## 04 - Operazioni su immagini e su matrici

Considerando due immagini come matrici posso sommarle? Si ma la matrice risultante esisterà matematicamente ma dal punto di vista visivo non avrà alcun senso.

Si possono fare quindi le sottrazioni ma per quanto riguarda le moltiplicazioni vanno seguite le regole che si applicano sulle matrici, difatti le colonne della prima matrice devono essere dello stesso numero delle righe della seconda matrice, generalmente non lo sono quindi non si può fare solitamente, segue per la divisione.

### Prodotto puntuale

Un prodotto puntuale invece non è altro che un prodotto di ogni punto per un altro punto della matrice, sempre come prima dal punto di vista visivo non hanno alcun significato.

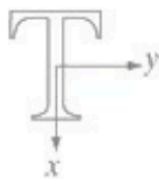
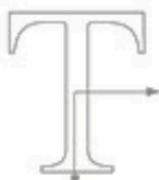
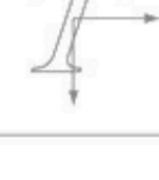
### Operazioni affini

Le operazioni sono delle operazioni che possono essere fatte sulle immagini e a differenza di quelle viste prima hanno un significato visivo, vediamo per esempio le operazioni:

- Traslazione
- Scaling
- Shear
- Identità
- Rotazione

In generale le trasformazioni affini sono le trasformazioni più generali che preservano i sottospazi affini.

Vediamo quindi uno schema riepilogativo:

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)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

## Mapping

Abbiamo visto però che con queste operazioni vengono creati dei buchi, dato che Matlab non sa che valori assegnare, perciò le matrici vengono riempite di 0, rappresentati come pixel neri, per risolvere questo problema possiamo fare alcune operazioni specifiche.

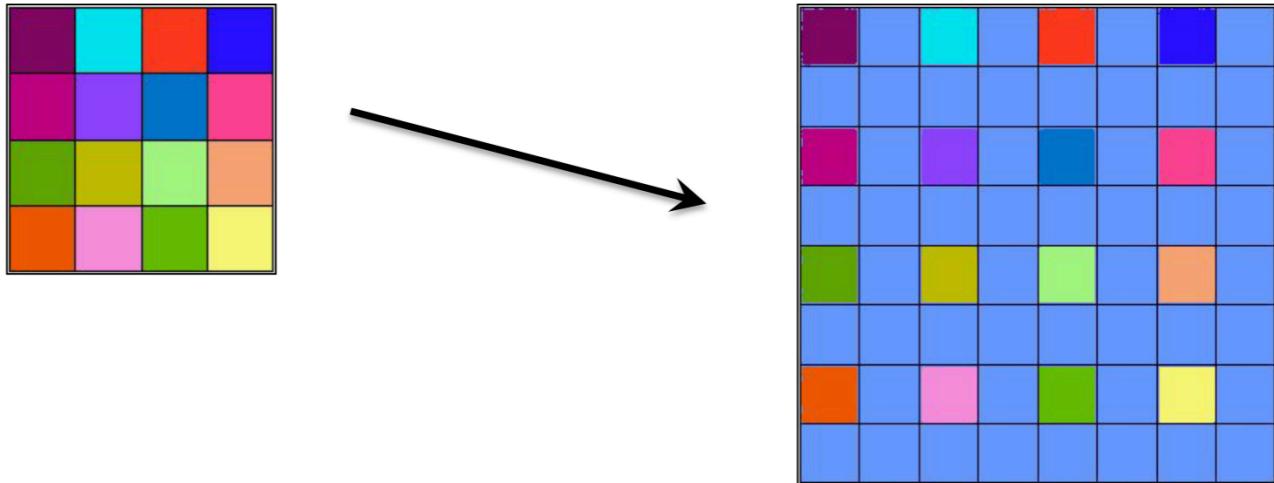
Quello fatto fin'ora è il **forward mapping** mentre tramite il **inverse mapping** possiamo risolvere questa problematica.

Vediamone le differenze:

- **Forward mapping:** 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).

## Interpolazione

Un'altra soluzione per riempire i buchi è quello dell'**interpolazione**, considerando l'operazione dello zooming in, la dimensione della matrice da  $n \times m$  diventerà  $2n \times 2m$ , sarà quindi necessario ricostruire i pixel mancanti.



Esistono vari tipi di interpolazione, un esempio in questo caso può essere il nearest neighbour, ovvero andare a clonare il pixel più vicino.

### Nearest neighbour

Quelli più vicini sono rappresentati da quelli nei 4 punti cardinali compresi anche gli spazi in diagonali per un totale di 8, componendo il vicinato del pixel considerato.



(a)



(b)

Il risultato però lascia molto a desiderare andando a formare una scalettatura molto evidente, il suo vantaggio più grande però è il non aver mai creato un valore non presente nell'immagine di partenza.

## Interpolazione bilineare

Nell'interpolazione lineare si prendono i 4 pixel più vicini a quello da generare facendo una stima del valore che deve avere il nuovo pixel creato.

Otteniamo in particolare il valore assegnato con la seguente equazione:

$$v(x, y) = ax + by + cxy + d$$

Dove i quattro coefficienti sono derivati dal sistema in quattro equazioni nelle quattro incognite derivate dai 4 pixel più vicini al punto di coordinate  $v(x, y)$ , attenzione, questa operazione va effettuata per ogni pixel.



(a)



(b)

Notiamo come l'output sia più rifinito, sebbene comunque il contrasto sia nettamente degradato, dato che le medie implicano sfocatura.

## Interpolazione bicubica

L'interpolazione bicubica consiste nel prendere i 16 pixel più vicini al pixel considerato attraverso la seguente equazione:

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

Dove i sedici coefficienti sono ricavati da un sistema di sedici equazioni in sedici incognite partendo dai sedici punti più vicini al pixel interessato. L'interpolazione bicubica permette una migliore preservazione della qualità di immagine.



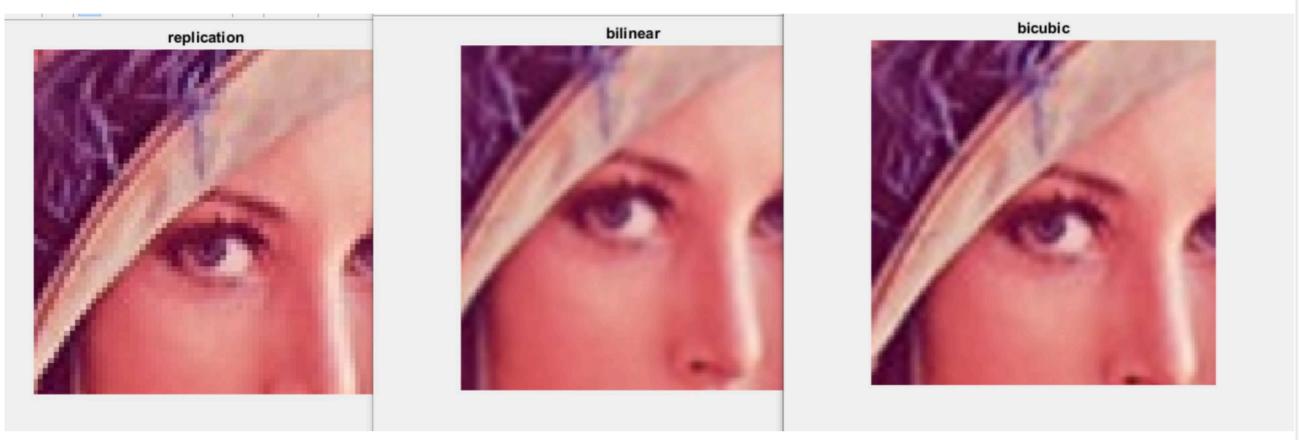
(a)



(b)

Naturalmente come per la bilineare questa operazione va fatta per ogni pixel (evvai).

Infine vediamo una comparazione fra le tre interpolazioni:



input

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

output

1	3
9	11

Cosa fare però quando la maschera di interpolazione non entra all'interno dell'immagine? Ci sono diversi metodi:

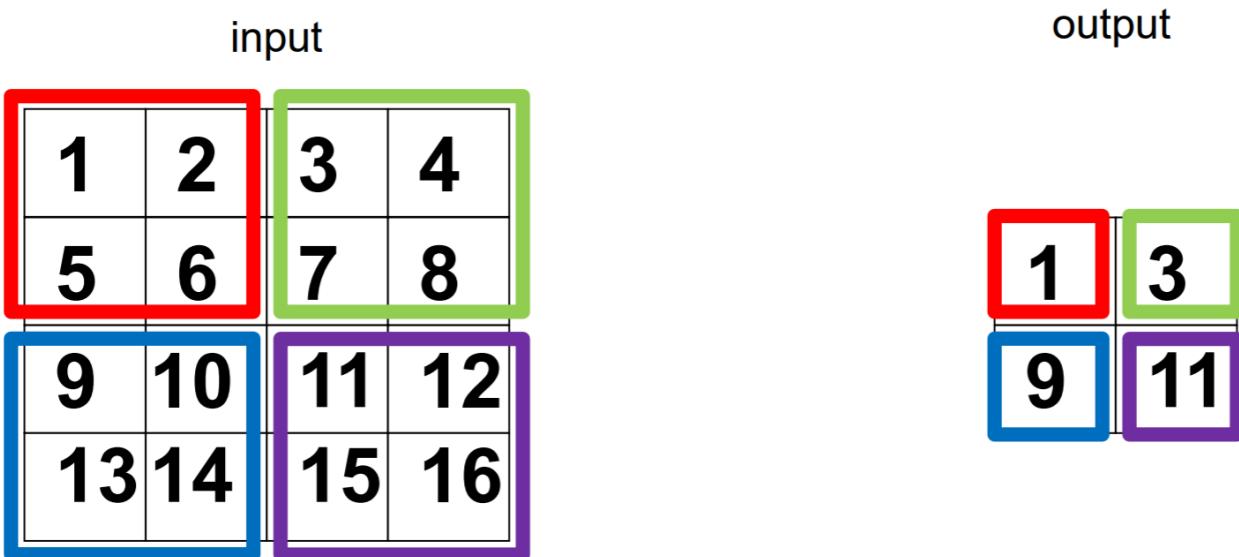
- **Non fare nulla:** La maschera di interpolazione viene applicata solo quando è possibile collocarla per intero all'interno della matrice di pixel.
- **Creazione di nuovi bordi virtuali:** Facciamo finta di possedere valori immaginari fuori, identici a quelli dentro ed applicarne l'algoritmo su questa maschera.

L'ideale, se non è necessario è non fare nulla, ed infatti è la tecnica più utilizzata, se è necessario che i numeri vadano riempiti il metodo migliore è quello di copiare sui bordi i valori adiacenti.

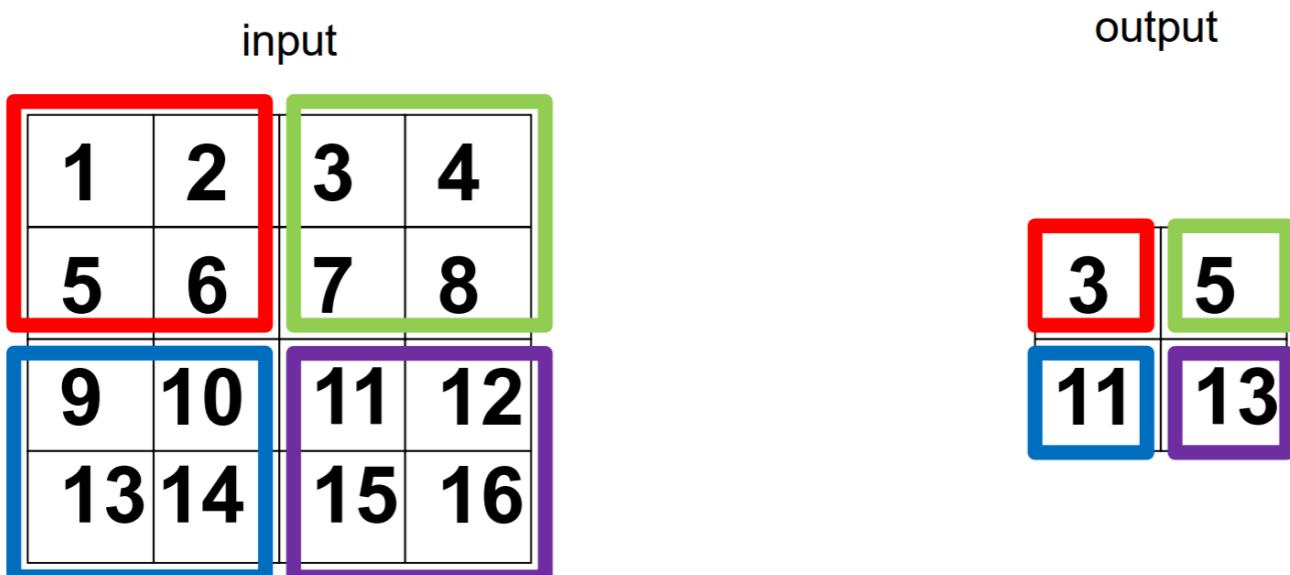
## Zooming out

Se il fattore di zoom ha valori minori di 1 allora stiamo facendo un rimpicciolimento quindi la nostra matrice da valori  $n \times m$  avremo una matrice con valori  $\frac{n}{2} \times \frac{m}{2}$ , ci sono due metodi per fare ciò.

1. Ogni 4 pixel ne scegliamo 1



## 2. Di 4 pixel ne facciamo la media



## Stima della qualità di un algoritmo

Il primo step è fare la differenza punto a punto delle due matrici al quadrato (in modo che valori negativi e valori positivi non si vadano a compensare), sommiamo tutti gli elementi e dividiamo il risultato per il numero di elementi, questo valore viene definito **MSE (Mean Square Error)** -> più è alto il valore più le due immagini sono diverse, più è basso più sono simili. Il massimo valore ottenibile è  $255^2$ .

La sua formula è:

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

Un altro indice di qualità è il **PSNR (Peak Signal to Noise Ratio)** per calcolare il PSNR è necessario possedere l'MSE in primis, a differenza dell'MSE maggiore è il valore del PSNR maggiore sarà la somiglianza con la foto originale.

Ha tre formule, vediamole:

$$PSNR = -10 \log_{10} \frac{MSE}{S^2}$$

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

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