



## ELABORAZIONE DELLE IMMAGINI

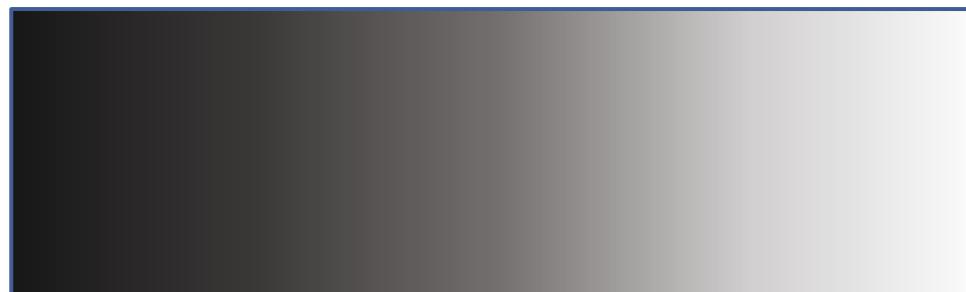
FILTRAGGIO SPAZIALE

## FILTRAGGIO

- Il termine **filtraggio** si riferisce ad una tecnica sviluppata per il dominio delle frequenze che consente di far **passare** o **bloccare** alcuni elementi (frequenze)
- Ci occuperemo di **due tipi di variazioni** d'intensità dei pixel dell'immagine
  - repentini (alte frquenze)
  - graduali (basse frequenze)
- Gli **effetti** del processo di filtraggio che analizzeremo sono
  - l'attenuazione
  - il miglioramento di dettagli



# FILTRAGGIO

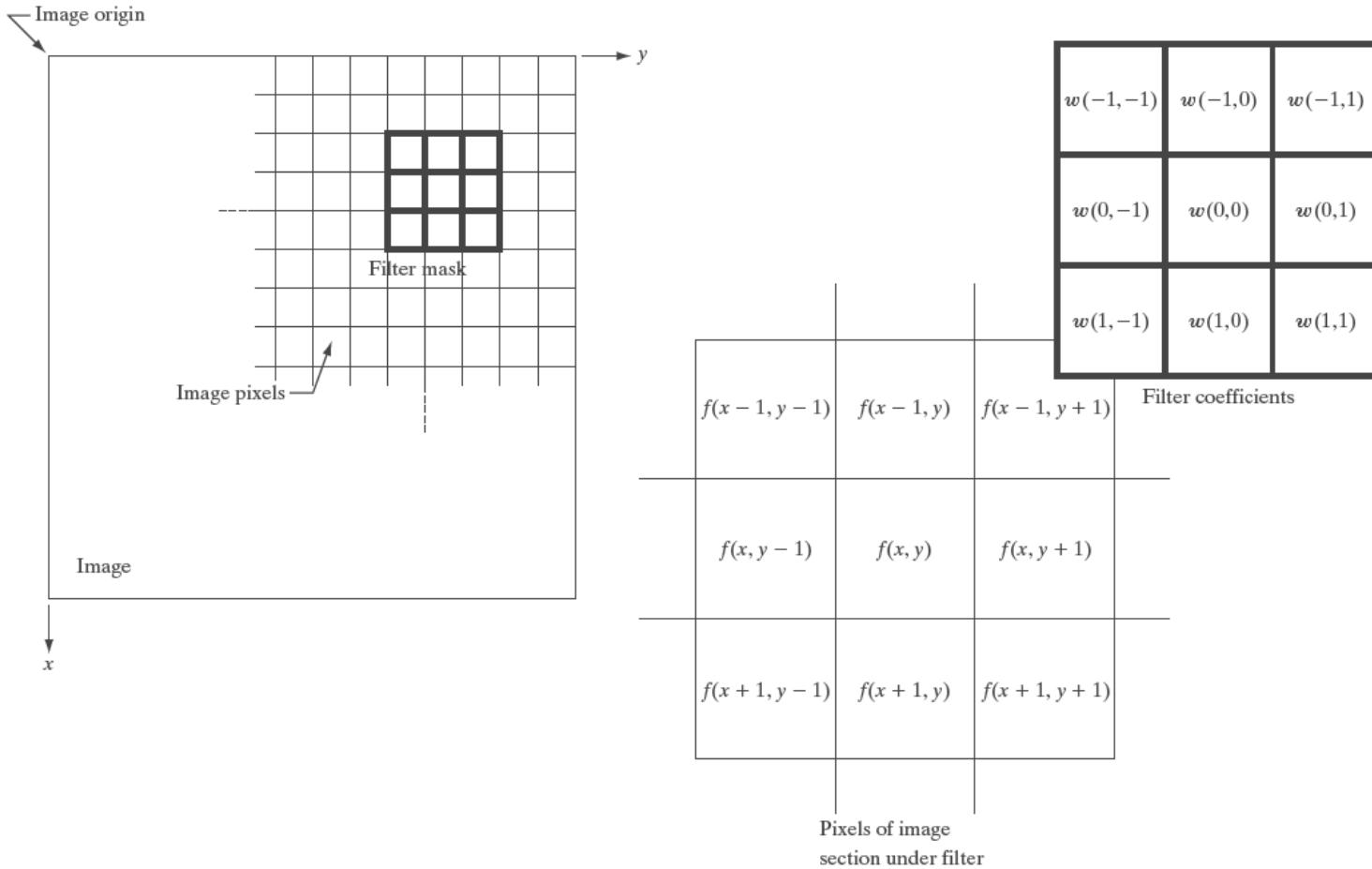


Elim Parte II – Prof. A. Ferone

## FILTRAGGIO SPAZIALE

- Le tecniche di **filtraggio spaziale** operano sui pixel di un'immagine prendendo in considerazione i valori di intensità in un intorno (**neighborhood**)
  - Per **ogni pixel** dell'**immagine originale**, è calcolata l'**intensità** del pixel corrispondente nell'**immagine filtrata**
  - La **regola di trasformazione** spesso è descritta da una **matrice**, chiamata **filtro** (maschera o kernel), della stessa dimensione dell'**intorno**
  - Se la regola di **trasformazione** è una funzione **lineare** delle intensità nell'intorno, la tecnica è chiamata **filtraggio lineare spaziale** (altrimenti, non lineare)

# FILTRAGGIO LINEARE



Elim Parte II – Prof. A. Ferone

## FILTRAGGIO LINEARE

- Il **pixel** nell'**immagine filtrata**,  $g(x,y)$ , è ottenuto come **combinazione lineare** dei pixel nell'immagine originale,  $f()$ , in un intorno di  $(x, y)$ :

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$

- La matrice peso,  $w()$ , è il filtro (maschera o kernel)
- Per comodità, spesso è usata una **matrice** con un numero **dispari** di righe  $2a+1$ , e colonne,  $2b+1$

# CORRELAZIONE E CONVOLUZIONE

## ■ **Correlazione**

- Progressivo scorrimento di una maschera sull'immagine e nel calcolo della somma dei prodotti in ogni posizione

## ■ **Convoluzione**

- Come la correllazione ma il filtro viene ruotato di 180°
- Per poter applicare il filtro a tutti i pixel dell'immagine si inseriscono opportuni valori su tutti i lati dell'immagine (padding)

## CORRELAZIONE 1-D

Impulso unitario discreto

$$\begin{array}{ccccccccccccc} & & & f & & & & & & w & & & & \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & 1 & 2 & 3 & 2 & 8 \end{array}$$

$$\begin{array}{ccccccccccccc} f & & & & \downarrow & & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ w & 1 & 2 & 3 & 2 & 8 & & & & & & & & & \end{array}$$

↑ starting position alignment

$$\begin{array}{cccccccccccccccccccc} & \overline{0 \ 0 \ 0 \ 0} & & \text{zero padding} & & \overline{0 \ 0 \ 0 \ 0} & & \\ f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w & 1 & 2 & 3 & 2 & 8 & & & & & & & & & & & \end{array}$$

## CORRELAZIONE 1-D

$$\begin{array}{llllllllllllllll} f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w & 1 & 2 & 3 & 2 & 8 & & & & & & & & & & \\ w * f & & & & & & 0 & & & & & & & & & \end{array}$$

$$\begin{array}{llllllllllllllll} f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w & & 1 & 2 & 3 & 2 & 8 & & & & & & & & & \\ w * f & & & & & & 0 & 0 & & & & & & & & \end{array}$$

$$\begin{array}{llllllllllllllll} f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w & & & 1 & 2 & 3 & 2 & 8 & & & & & & & & \\ w * f & & & & & & 0 & 0 & 0 & & & & & & & \end{array}$$

$$\begin{array}{llllllllllllllll} f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w & & & 1 & 2 & 3 & 2 & 8 & & & & & & & & \\ w * f & & & & & & 0 & 0 & 0 & 8 & & & & & & \end{array}$$

Elim Parte II – Prof. A. Ferone

## CORRELAZIONE 1-D

$f$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$w$					1	2	3	2	8										
$w \star f$					0	0	0	8											

$f$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$w$					1	2	3	2	8										
$w \star f$					0	0	0	8	2										

$f$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$w$					1	2	3	2	8										
$w \star f$					0	0	0	8	2	3									

$f$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$w$					1	2	3	2	8										
$w \star f$					0	0	0	8	2	3	2								

Elim Parte II – Prof. A. Ferone

## CORRELAZIONE 1-D

$f$	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
$w$							1	2	3	2	8						
$w \star f$					0	0	0	8	2	3	2	1					

• • •

$f$	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
$w$												1	2	3	2	8	
$w \star f$					0	0	0	8	2	3	2	1	0	0	0	0	0

$w \star f$	0	0	0	8	2	3	2	1	0	0	0	0	0	0	0	0	0
Cropping	—	0	8	2	3	2	1	0	0	0	—						
$f$	0	0	0	1	0	0	0	0	0	0	0						

w ruotato di 180°



## CONVOLUZIONE

- La convoluzione è un'operazione fondamentale nella teoria dei sistemi lineari
- Una delle **proprietà fondamentali** è che la convoluzione di una funzione con l'impulso unitario produce una copia della funzione nella posizione dell'impulso
- La **correlazione** produce lo stesso risultato ma l'**immagine** è **ruotata** di  $180^\circ$
- Per applicare la **convoluzione**
  - **pre-rotazione** del filtro di  $180^\circ$
  - applicazione delle operazioni della **correlazione**



## CONVOLUZIONE

$$\begin{array}{ccccccccc} & & f & & & & w \text{ rotated } 180^\circ & & \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 8 & 2 \\ & & & & & & & 3 & 2 \\ & & & & & & & & 1 \end{array}$$

$$\begin{array}{ccccccccc} & & f & & & & w & & \\ & & & & & & 0 & 0 & 0 \\ & & & & & & 1 & 0 & 0 \\ & & w & 8 & 2 & 3 & 2 & 1 & 0 \\ & & & & & & & & 0 \\ & & & & & & & & 0 \\ & & & & & & & & 0 \\ & & & & & & & & 0 \\ & & & & & & & & 0 \end{array}$$

↑ starting position alignment

• • •

$$\begin{array}{ccccccccc} w * f & 0 & 0 & 0 & 1 & 2 & 3 & 2 & 8 & 0 & 0 & 0 & 0 \\ \text{Cropping} & \underline{\quad} & 0 & 1 & 2 & 3 & 2 & 8 & 0 & 0 & \underline{\quad} \\ f & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Elim Parte II – Prof. A. Ferone

## CORRELAZIONE E CONVOLUZIONE 2-D

- Correlazione

$$g(x, y) = w(x, y) \star f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x+s, y+t)$$

- Convoluzione

$$g(x, y) = w(x, y) * f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x-s, y-t)$$

Padded $f$									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
Origin $f(x, y)$					0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	$w(x, y)$	0	0	0	0
0	0	1	0	0	1	2	3	0	0
0	0	0	0	0	4	5	6	0	0
0	0	0	0	0	7	8	9	0	0

Elim Parte II – Prof. A. Ferone

# CORRELAZIONE E CONVOLUZIONE 2-D

Initial position for  $w$

1	2	3	0	0	0	0
4	5	6	0	0	0	0
7	8	9	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

## Full correlation result

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	9	8	7	0	0
0	0	0	6	5	4	0	0
0	0	0	3	2	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

### Cropped correlation result

0	0	0	0	0
0	9	8	7	0
0	6	5	4	0
0	3	2	1	0
0	0	0	0	0

$\nabla$  Rotated  $w$

9	8	7	0	0	0	0
6	5	4	0	0	0	0
3	2	1	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

## Full convolution result

### Cropped convolution result

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

Elim Parte II – Prof. A. Ferone

## RAPPRESENTAZIONE VETTORIALE

- Quando la posizione relativa dei coefficienti non è importante, è possibile rappresentare la risposta del filtro usando una rappresentazione vettoriale
  - Deve essere specificata una indicizzazione convenzionale

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

$$R = w_1 z_1 + \cdots + w_m z_m = \sum_{k=1}^{mn} w_k z_k = \mathbf{w}^T \mathbf{z}$$

Elim Parte II – Prof. A. Ferone

## CORRELAZIONE OPENCV

- In OpenCV è possibile effettuare la correlazione usando la funzione **cv::filter2D()**

```
cv::filter2D(  
    cv::InputArray src,                      // Input image  
    cv::OutputArray dst,                     // Result image  
    int ddepth,                            // Output depth (e.g., CV_8U)  
    cv::InputArray kernel,                  // Your own kernel  
    cv::Point anchor = cv::Point(-1,-1),    // Location of anchor point  
    double delta = 0,                      // Offset before assignment  
    int borderType = cv::BORDER_DEFAULT // Border extrapolation to use  
);
```

## CONVOLUZIONE OPENCV

- Per effettuare la convoluzione è necessario ruotare il filtro di 180°

```
void rotate(InputArray src, OutputArray dst, int rotateCode);
```

- rotateCode:
  - ROTATE\_90\_CLOCKWISE
  - ROTATE\_180
  - ROTATE\_90\_COUNTERCLOCKWISE
- rot.cpp sulla piattaforma elearning

## SPECIFICA DEL FILTRO

- Per creare un **filtro lineare** spaziale è necessario specificarne i coefficienti
- Specifica diretta:

$$R = \frac{1}{9} \sum_{i=1}^9 z_i \quad \Rightarrow \quad w_i = \frac{1}{9}, \quad \forall i$$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- Specifica basata su una funzione:

$$h(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \Rightarrow \quad w(s, t) = h(s, t)$$

- Per i **filtri non lineari** si specifica solo l'intorno su cui si applicheranno determinate operazioni
  - Es.: filtro "max" su un intorno 5x5

## FILTRI DI SMOOTH

- I filtri di **smoothing** vengono utilizzati per sfocare le immagini (**blurring**) e per la riduzione del rumore (denoising)
- Operano **rimuovendo i dettagli** più piccoli della dimensione del filtro
  - Si evidenziano gli oggetti più grandi
  - Si riempiono piccoli gap (es.: interruzioni di linee o curve)
- Per la riduzione del rumore è possibile utilizzare **filtri lineari o non lineari**, a seconda del **tipo di rumore**

## FILTRI DI SMOOTHING LINEARI

- **Filtri di media** (passa basso o low pass)
- Il valore di ogni pixel viene sostituito con la media dei livelli di intensità nell'intorno definito dalla maschera
- L'effetto è un'immagine in cui le brusche transizioni di intesità sono ridotte
  - In questo modo vengono sfocati anche gli edge (lati)

Filtro box

$$\frac{1}{9} \times$$

1	1	1
1	1	1
1	1	1

Elim Parte II – Prof. A. Ferone

$$\frac{1}{16} \times$$

1	2	1
2	4	2
1	2	1

Filtro media ponderata

## EFFETTI DI SMOOTHING



Immagine originale

Elim Parte II – Prof. A. Ferone



Filtro media 3x3

## EFFETTI DI SMOOTHING



Immagine originale



Filtro media 5x5



Filtro media 9x9

Elim Parte II – Prof. A. Ferone

## EFFETTI DI SMOOTHING



Immagine originale



Filtro media 15x15



Filtro media 35x35

Elim Parte II – Prof. A. Ferone

## FILTRI DI SMOOTH OPENCV

- In OpenCV esistono diverse funzioni di smoothing

```
void cv::blur(  
    cv::InputArray src,                                // Input image  
    cv::OutputArray dst,                               // Result image  
    cv::Size ksize,                                 // Kernel size  
    cv::Point anchor = cv::Point(-1,-1), // Location of anchor point  
    int borderType = cv::BORDER_DEFAULT // Border extrapolation to use  
);
```

```
void cv::boxFilter(  
    cv::InputArray src,                                // Input image  
    cv::OutputArray dst,                               // Result image  
    int ddepth,                                    // Output depth (e.g., CV_8U)  
    cv::Size ksize,                                 // Kernel size  
    cv::Point anchor = cv::Point(-1,-1), // Location of anchor point  
    bool normalize = true,                         // If true, divide by box area  
    int borderType = cv::BORDER_DEFAULT // Border extrapolation to use  
);
```

Elim Parte II – Prof. A. Ferone

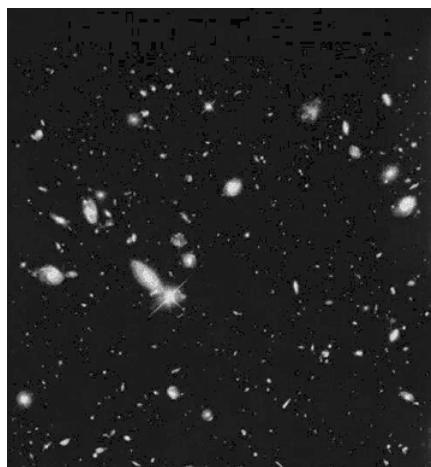
## FILTRI DI SMOOTH OPENCV

- In OpenCV esistono diverse funzioni di smoothing

```
void cv::GaussianBlur(  
    cv::InputArray src,                      // Input image  
    cv::OutputArray dst,                     // Result image  
    cv::Size ksize,                         // Kernel size  
    double sigmaX,                         // Gaussian half-width in x-direction  
    double sigmaY = 0.0,                    // Gaussian half-width in y-direction  
    int borderType = cv::BORDER_DEFAULT // Border extrapolation to use  
) ;
```

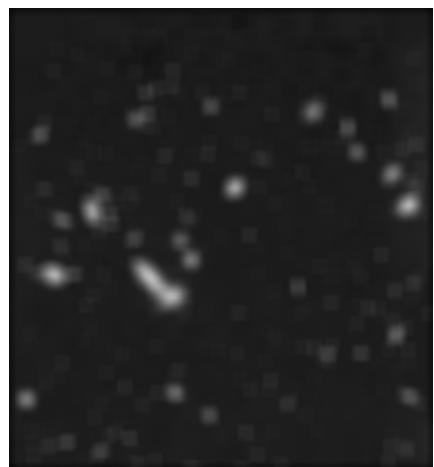
## ESEMPIO: RIMOZIONE DI DETTAGLI

(a)



$f$

(b)



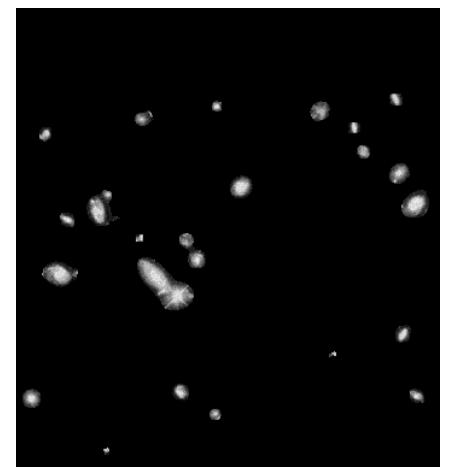
$g = f * h$

(c)



$m = g > \theta$

(d)



$k$

(a)  $f$ , immagine  $353 \times 382$

(b)  $g$ , immagine filtrata con filtro di smooth  $13 \times 13$

(c)  $m$ , immagine con sogliatura ( $\theta=25\%$  dell'intensità massima)

(d)  $k$ , immagine mascherata:  $k(x, y) = f(x, y) \times m(x, y)$

Elim Parte II – Prof. A. Ferone

# THRESHOLDING OPENCV

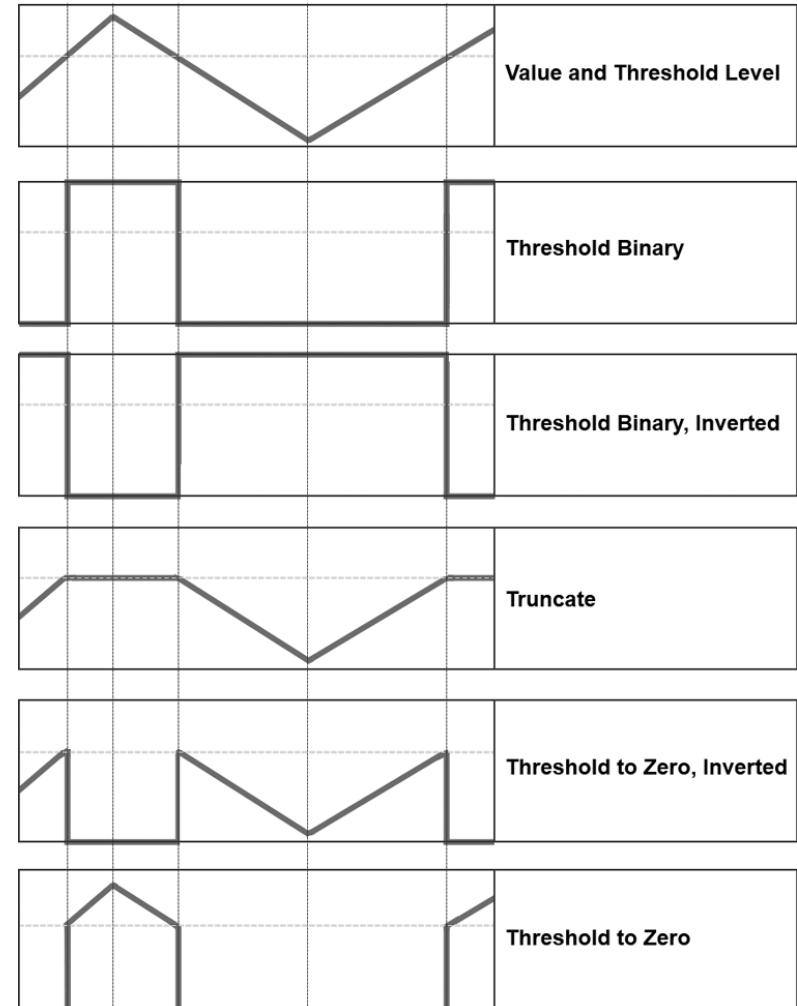
- Per effettuare la sogliatura (thresholding) in OpenCV

```
double cv::threshold(  
    cv::InputArray    src,           // Input image  
    cv::OutputArray   dst,           // Result image  
    double           thresh,         // Threshold value  
    double           maxValue,       // Max value for upward operations  
    int              thresholdType // Threshold type to use  
);
```

```
void cv::adaptiveThreshold(  
    cv::InputArray    src,           // Input image  
    cv::OutputArray   dst,           // Result image  
    double           maxValue,       // Max value for upward operations  
    int              adaptiveMethod, // mean or Gaussian  
    int              thresholdType // Threshold type to use  
    int              blockSize,      // Block size  
    double           C,             // Constant  
);
```

# THRESHOLDING OPENCV

Threshold type	Operation
<code>cv::THRESH_BINARY</code>	$DST_I = (SRC_I > thresh) ? MAXVALUE : 0$
<code>cv::THRESH_BINARY_INV</code>	$DST_I = (SRC_I > thresh) ? 0 : MAXVALUE$
<code>cv::THRESH_TRUNC</code>	$DST_I = (SRC_I > thresh) ? THRESH : SRC_I$
<code>cv::THRESH_TOZERO</code>	$DST_I = (SRC_I > thresh) ? SRC_I : 0$
<code>cv::THRESH_TOZERO_INV</code>	$DST_I = (SRC_I > thresh) ? 0 : SRC_I$



Elim Parte II – Prof. A. Ferone

## FILTRI NON LINEARI BASATI SU STATISTICHE D'ORDINE

- La risposta di questi filtri consiste
  - Ordinare i pixel contenuti nell'intorno definito dal filtro
  - Sostituire il valore del pixel centrale con un valore dell'insieme ordinato
- Esempi
  - Filtro mediano (sostituzione con il valore mediano)
  - Filtri Max e Min
  - Filtri basati su percentile



## FILTRI NON LINEARI BASATI SU STATISTICHE D'ORDINE

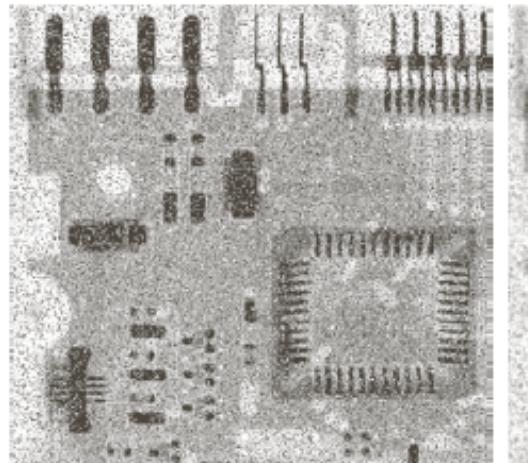
- $f = [100, 120, 98, 99, 110, 255, 100, 200, 10]$
- Ordinato:  $[10, 98, 99, 100, 100, 110, 120, 200, 255]$
- Media: 121
- Mediana: 100
- Min: 10
- Max: 255

10	98	99
100	100	110
120	200	255

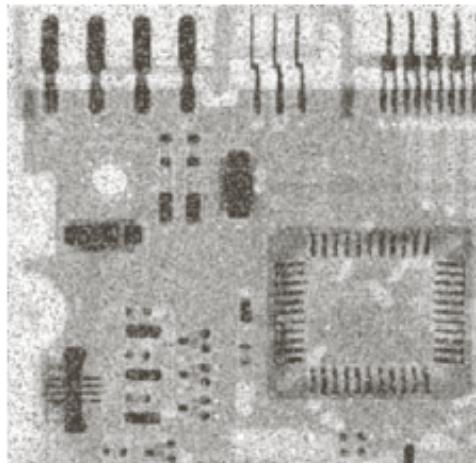
Elim Parte II – Prof. A. Ferone

## ESEMPIO: FILTRO MEDIANO

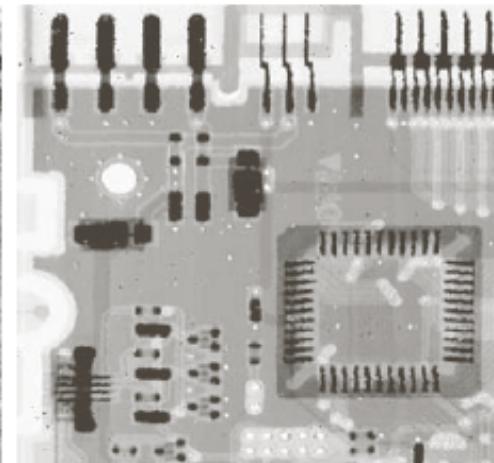
- I filtri mediani sono efficaci in presenza di rumore ad impulso
  - (a) Immagine originale, corrotta da rumore sale e pepe
  - (b) Immagine filtrata con filtro media  $3 \times 3$
  - (c) Immagine filtrata con filtro mediano  $3 \times 3$



(a)



(b)



(c)

Elim Parte II – Prof. A. Ferone

## FILTRO MEDIANO OPENCV

- Per applicare il filtro mediano si utilizza la funzione **cv::medianBlur()**

```
void cv::medianBlur(  
    cv::InputArray src,           // Input image  
    cv::OutputArray dst,          // Result image  
    cv::Size      ksize           // Kernel size  
) ;
```

## ESERCIZI

- Testare il codice filtering.cpp disponibile sulla piattaforma elearning
- Implementare l'algoritmo di correlazione/convoluzione e confrontare l'output con quello prodotto dalla funzione filter2D() usando un filtro media.