

Elaborazione di segnali e immagini

VR443470

febbraio 2024

Indice

1 Fondamenti	6
1.1 Matematica preliminare	6
1.1.1 Numeri complessi	6
1.1.2 Funzioni complesse di variabile reale	7
1.1.3 Funzioni pari e dispari	8
1.1.4 Segnali periodici	9
1.2 Operazioni fondamentali	10
1.2.1 Somma	10
1.2.2 Shift (o traslazione)	11
1.2.3 Funzione box Π e impulso di Dirac	12
1.2.4 Funzione sinc	13
1.2.5 Funzione triangolo Λ	13
1.2.6 Funzione segno (sgn)	13
1.2.7 Funzione gradino	13
1.2.8 Treno di impulsi	14
1.2.9 Energia di un segnale	14
1.2.10 Potenza media di un segnale	15
1.3 Altre operazioni fondamentali	16
1.3.1 Rescaling (o riscalatura)	16
1.3.2 Cross-Correlazione	17
1.3.3 Esercizi d'esame	18
1.3.4 Cross-Correlazione Normalizzata	24
1.3.5 Convoluzione	30
2 Analisi di Fourier	31
2.1 Serie di Fourier	31
2.1.1 Proprietà della serie di Fourier	37
2.2 Trasformata di Fourier continua	38
2.2.1 Trasformata di Fourier	38
2.2.2 Trasformata di Fourier inversa	38
2.2.3 Proprietà della trasformata di Fourier	40
2.2.4 Trasformata di Fourier di una box	41
2.2.5 Trasformata di Fourier di un sinc	42
2.2.6 Trasformata di Fourier di un impulso	43
2.2.7 Trasformata di Fourier di un treno di impulsi	44
2.2.8 Sintesi	45
2.3 Trasformata di Fourier a tempo discreto	46
2.3.1 Campionamento	46
2.3.2 Trasformata di Fourier a tempo discreto	47
2.3.3 Teorema del campionamento	48
2.3.4 Considerazioni	48
2.4 Trasformata di Fourier discreta	49
2.5 Riassunto Trasformate	51
2.6 Domanda da esame	52

3 Elaborazione di immagini - Dominio spaziale	53
3.1 Strumento per l'elaborazione: istogramma	54
3.2 Domini	56
3.3 Operazioni puntuali	57
3.3.1 Identità	58
3.3.2 Negativo	58
3.3.3 Clamping	58
3.3.4 Stretching/Shrinking	58
3.3.5 Trasformazione logaritmica	59
3.3.6 Trasformazione esponenziale	59
3.3.7 Trasformazione di potenza	60
3.3.8 Binarizzazione	60
3.3.9 Binarizzazione attraverso il metodo di Otsu	61
3.3.10 Equalizzazione	62
3.4 Operazioni locali	66
3.4.1 Filtraggi spaziali: lineari e non lineari	67
3.5 Rumore nelle immagini	69
3.5.1 Rumore gaussiano additivo bianco	70
3.5.2 Rumore impulsivo	71
3.6 Altre operazioni locali: tipologie di filtraggio	72
3.6.1 Smoothing - Filtro media	73
3.6.2 Smoothing - Filtro mediano	74
3.6.3 Smoothing - Filtro gaussiano	75
3.6.4 Domanda da esame	76
3.6.5 Filtraggi di sharpening	77
3.6.6 Sharpening - Basic Highpass Spatial Filtering	79
3.6.7 Sharpening - Filtro Laplaciano	80
3.6.8 Sharpening - High Boost Filtering	80
4 Elaborazione di immagini - Rinforzo del dominio delle frequenze	81
4.1 Ripasso formule utili	81
4.2 Rumore nel dominio delle frequenze	83
4.3 Panoramica su filtri passa alto (<i>high-pass</i>) e passa basso (<i>low-pass</i>)	84
4.4 Filtri passa basso (<i>low-pass</i>)	85
4.4.1 Filtri passa basso ideale	85
4.4.2 Filtri passa basso di Butterworth	87
4.4.3 Filtri passa basso Gaussiano	89
4.4.4 Sintesi	89
4.5 Filtri passa alto (<i>high-pass</i>)	90
4.5.1 Filtri passa alto ideale	91
4.5.2 Filtri passa alto di Butterworth	91
4.5.3 Filtri passa alto Gaussiano	92
4.6 Filtri per enfatizzare le alte frequenze	93
4.7 Filtri passa banda ideale 1D e ferma banda ideale	94

5 Elaborazione di immagini - Estrazione di edge robusta	97
5.1 Estrazione di contorni (<i>edge</i>)	97
5.2 Criteri per <i>edge detector</i> ottimi	99
5.3 Il gradiente	100
5.4 Estrazione dell' <i>edge</i>	101
5.4.1 Il gradiente discreto	101
5.4.2 Operatore di Roberts	101
5.4.3 Operatore di Prewitt	101
5.4.4 Operatore di Sobel	102
5.4.5 Operatore di Kirsch	102
5.4.6 Operatore di Robinson	102
5.4.7 Confronto tra operatori	103
5.5 Derivata prima	105
5.5.1 Problemi	105
5.5.2 Effetti del rumore	105
5.5.3 Rilevazione di <i>edge</i> o <i>detection</i> robusta	105
5.5.4 Teorema di convoluzione	106
5.6 Derivata seconda	107
5.6.1 Operatori di derivata seconda	107
5.6.2 Operatore Laplaciano	107
5.7 Laplaciano di Gaussiana	108
5.8 Gradiente e Laplaciano di Gaussiana a confronto	111
5.9 Filtro di Canny	112
5.9.1 Definizione	112
5.9.2 Modello di <i>edge</i> per Canny	112
5.9.3 Algoritmo di Canny	112
5.9.4 Filtro di Rinforzo	113
5.9.5 Non-Maxima Suppression	114
5.9.6 Sogliatura ad isteresi	115
5.9.7 Edge linking	116
5.10 Prestazioni di un <i>edge detector</i>	117
5.11 Figura di merito per gli <i>edge</i>	117
5.12 Trasformata di Hough	118
5.12.1 Definizione	118
5.12.2 Trasformata per le rette	120
5.12.3 Trasformata per i cerchi	122
6 Laboratorio MATLAB	124
6.1 Introduzione a MATLAB	124
6.1.1 Script e funzioni	124
6.1.2 For, while, if	124
6.1.3 Condizioni su vettori e matrici (find)	125
6.1.4 Esercizio 1	126
6.1.5 Esercizio 2	127
6.1.6 Esercizio 3	128
6.2 Concetti avanzati di MATLAB	130
6.2.1 Debug	130
6.2.2 Rappresentazione grafica dei segnali	131
6.2.3 Rappresentazione dei suoni	134
6.2.4 Rappresentazione delle immagini	135

6.2.5	Ulteriori comandi utili per le immagini	138
6.2.6	Esercizio 1	140
6.2.7	Esercizio 2	142
6.3	Cross Correlazione 1D	144
6.3.1	Cross correlazione	144
6.3.2	Applicazione della cross correlazione	145
6.3.3	Cross correlazione nella pratica	147
6.3.4	Esercizio 1	148
6.3.5	Esercizio 2	151
6.4	Cross correlazione 2D	155
6.4.1	Estensione della cross correlazione 1D	155
6.4.2	Applicazione della cross correlazione 2D	155
6.4.3	Ottimizzazione della cross correlazione 2D	157
6.4.4	Esercizio 1	158
6.4.5	Esercizio 2	161
6.5	Trasformata di Fourier Discreta 1D	163
6.5.1	Spiegazione teorica sulla TdF discreta nei calcolatori	163
6.5.2	Aliasing	164
6.5.3	Esempi di TdF discreta 1D	165
6.5.4	Esercizio 1	169
6.5.5	Esercizio 2	174
6.6	Operatori puntuali	176
6.6.1	Operazioni puntuali di rinforzo	176
6.6.2	Negativo	177
6.6.3	Clamping	178
6.6.4	Stretching/Shrinking	180
6.6.5	Trasformazione non lineare	181
6.6.6	Binarizzazione	184
6.6.7	Esercizio 1	185
6.6.8	Esercizio 2	189
6.7	Operatori locali	192
6.7.1	Filtraggio spaziale	192
6.7.2	Tipologie di filtro	193
6.7.3	Filtro media	193
6.7.4	Filtro gaussiano	194
6.7.5	Filtro mediano	195
6.7.6	Filtro laplaciano	196
6.7.7	Signal to noise ratio	197
6.7.8	Esercizio 1	199
6.7.9	Esercizio 2	205
6.7.10	Esercizio 3	209
6.8	TdF 2D e trasformata di Hough per rette	212
6.8.1	Trasformata di Fourier 2D	212
6.8.2	Applicazione della TdF 2D in MATLAB	212
6.8.3	La trasformata di Hough per rette	214
6.8.4	Dettagli sulla trasformata di Hough e algoritmo	215
6.8.5	Esercizio 1	215
6.8.6	Esercizio 2	219

1 Fondamenti

1.1 Matematica preliminare

1.1.1 Numeri complessi

Un numero complesso c appartiene all'insieme dei complessi \mathbb{C} e la sua forma è del tipo:

$$c = \Re + j\Im$$

con \Re, \Im variabili $\in \mathbb{R}$ e j chiamata *unità immaginaria* rappresentata come $j = \sqrt{-1}$. Inoltre, \Re rappresenta la *parte reale* e \Im la *parte immaginaria*. Il coniugato di c è

$$\tilde{c} = \Re - j\Im$$

I numeri complessi, dal punto di vista geometrico, possono essere visti come punti su un piano (chiamato *piano complesso*) e descritti da coordinate (R, I) . Nel piano complesso, le ascisse (x) sono rappresentate dalla parte reale, mentre le ordinate (y) dalla parte immaginaria.

Spesso è utile rappresentare i numeri complessi in coordinate polari formate nel seguente modo (*modulo, angolo*). Questa forma viene denominata *forma polare* di un numero complesso:

$$c = \Re + j\Im = |c|(\cos \theta + j \sin \theta)$$

dove:

$$|c| = \sqrt{\Re^2 + \Im^2} \longrightarrow \text{chiamato } \textit{modulo} \text{ o } \textit{magnitudo}$$

invece, *theta* rappresenta:

$$\theta \cong \arctan \left(\frac{\Im}{\Re} \right) \longrightarrow \text{chiamato } \textit{angolo}, \textit{fase} \text{ o } \textit{argomento in radianti}$$

Grazie alla formula di Eulero:

$$e^{j\theta} = \cos \theta + j \sin \theta$$

è possibile riscrivere la forma polare di un numero complesso in maniera alternativa, ossia:

$$c = \Re + j\Im = |c| (\cos \theta + j \sin \theta) = |c| e^{j\theta}$$

La **somma** e la **moltiplicazione** di due numeri complessi diventa:

$$c_1 = R_1 + jI_1 \quad c_2 = R_2 + jI_2$$

$$\text{Somma: } c_1 + c_2 = (R_1 + R_2) + j(I_1 + I_2)$$

$$\text{Moltiplicazione con Eulero: } c_1 \cdot c_2 = (R_1 R_2 - I_1 I_2) + j(R_1 I_2 + I_1 R_2) \longrightarrow = |c_1||c_2| e^{j(\theta_1 + \theta_2)}$$

1.1.2 Funzioni complesse di variabile reale

Dato $t \in \mathbb{R}$, una funzione f complessa di variabile reale è $f : D_1 \subseteq \mathbb{R} \rightarrow D_2 \subseteq \mathbb{C}$. Viene introdotto questo concetto poiché il **fasore** è un eSEMPIO fondamentALE. Le **caratteristiche** di questa funzione:

- È una funzione complessa che modella la posizione di un punto che ruota attorno all'origine con raggio determinato $|c|$ e velocità angolare costante $\theta(t)$.
- Se la funzione fosse nei numeri reali, sarebbe più dispendioso in termini di numero di funzioni da utilizzare.

L'**obiettivo** dei fasori è quello di *passare dal dominio del tempo* (o spazio) a *quello dell'analisi frequenziale*.

La particolarità è che nel tempo il fasore riesce a variare un numero complesso (in forma polare) mantenendo il modulo $|c|$ fisso:

$$|c|e^{j\theta} \rightarrow |c|e^{j\theta(t)}$$

dove $\theta(t)$ indica la **velocità angolare**. Quest'ultima può essere calcolata tramite:

$$\theta(t) \longrightarrow \frac{2\pi}{T_0}t + \phi$$

dove T_0 indica il *tempo* impiegato per eseguire 2π radianti.

Soltamente si utilizza il fasore con le seguenti supposizioni:

- Coordinate rappresentate con (R, I)
- Impostata una distanza unitaria fissa dall'origine $|c| = 1$
- Velocità angolare costante pari a $2\pi/\text{sec.}$, ossia $\theta(t) = 2\pi t, T_0 = 1\text{sec.}$
- Con $t = 0$ si ha $\theta = 0$
- Viene mantenuto $\phi = 0$

1.1.3 Funzioni pari e dispari

Una funzione $f : \mathbb{R} \rightarrow \mathbb{R}$ è **pari** se e solo se:

$$f(t) = f(-t)$$

Invece, una funzione $f : \mathbb{R} \rightarrow \mathbb{R}$ è **dispari** se e solo se:

$$f(t) = -f(-t)$$

1.1.4 Segnali periodici

Un segnale f è **periodico** di periodo T o T -periodico se:

$$\exists T_0 \in R^+ : f(t + T_0) = f(t), \quad \forall t \in D_1$$

e T_0 è il minor numero per cui la condizione di ripetizione si verifica.

Dato un periodo T_0 con la lettera μ_0 si indica la **frequenza fondamentale**:

$$\mu_0 = \frac{1}{T_0}$$

Fissato $T_0 > 0$ i **segnali trigonometrici** di minimo periodo T_0 sono:

$$f(t) = \cos(2\pi\mu_0 t) \quad f(t) = \sin(2\pi\mu_0 t)$$

dove μ è una frequenza generale, mentre $\mu_0 = \frac{1}{T_0}$ è la **frequenza fondamentale**. Invece, spesso la **velocità angolare** o **pulsazione** viene rappresentata come:

$$2\pi\mu_0 = \frac{2\pi}{T_0} = \omega_0$$

Inoltre, fissato un $\theta \in \mathbb{R}$ chiamato **fase** si osserva che anche le funzioni:

$$f(t) = \cos(2\pi\mu_0 t + \theta) \quad f(t) = \sin(2\pi\mu_0 t + \theta)$$

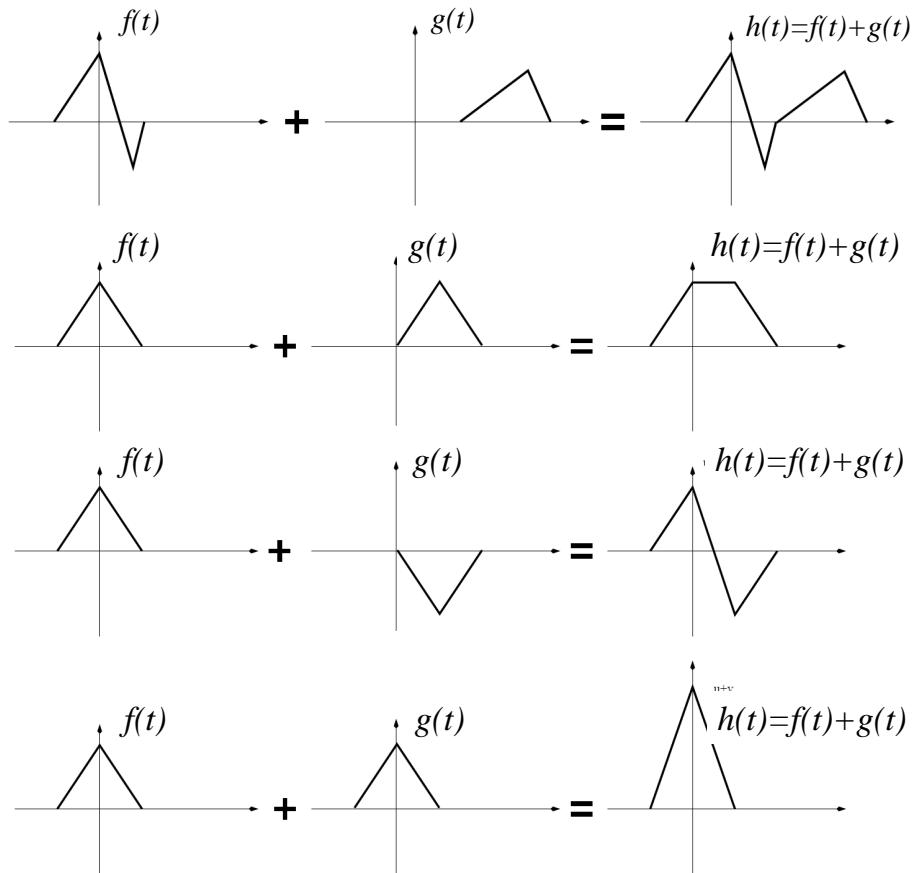
hanno il medesimo periodo T .

Infine, la fase θ permette di eseguire operazione di *shift*.

1.2 Operazioni fondamentali

1.2.1 Somma

La *somma* di due segnali è facile quando essi non interferiscono, ovvero quando **non** sono contemporaneamente $\neq 0$. Alcuni esempi qui di seguito.



1.2.2 Shift (o traslazione)

Lo **shift** (o traslazione) è il cambio di posizione di un segnale. Può essere effettuato:

- **Traslazione a destra** con la funzione $f(t - \tau)$
- **Traslazione a sinistra** con la funzione $f(t + \tau)$

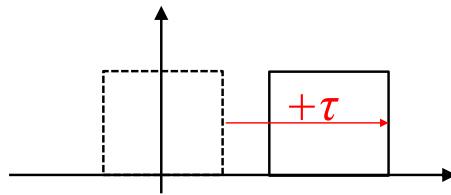


Figura 1: Shift a destra

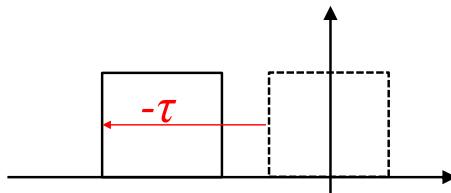


Figura 2: Shift a sinistra

1.2.3 Funzione box II e impulso di Dirac

La funzione ***box*** è definita nel seguente modo:

$$A\Pi\left(\frac{x}{b}\right) \quad x \in \left[-\frac{b}{2}, \frac{b}{2}\right]$$

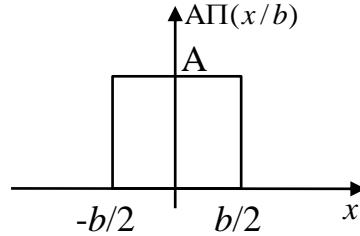


Figura 3: Box generica

La funzione $\delta(x)$ è chiamata ***impulso unitario*** o ***impulso di Dirac*** perché è definita nel seguente modo:

$$\delta(x) = \begin{cases} \infty & \text{se } x = 0 \\ 0 & \text{se } x \neq 0 \end{cases} \quad \int_{-\infty}^{\infty} \delta(x) dx = 1$$

Quindi è un impulso che tende all'infinito solamente quando la x è nell'origine, ma il suo integrale è uguale a 1. Alcune **proprietà** dell'impulso:

1. $\delta(x - x_0) = 0 \quad \forall x \neq x_0$
2. Data una funzione generica f (**setacciamento**): $\int_{-\infty}^{\infty} f(x)\delta(x - x_0) dt = f(x_0)$
3. $\delta(x - x_0) = \delta(x_0 - x)$
4. $\delta(ax) = \frac{1}{|a|}\delta(x) \quad \forall x \in \mathbb{R}, \text{ fissato } a \in \mathbb{R} - \{0\}$

1.2.4 Funzione sinc

La funzione **sinc** è definita nel seguente modo:

$$\text{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$$

Ha due **caratteristiche** importanti: (1) l'intersezione con l'asse delle x avviene sempre nei numeri interi positivi e negativi (quindi 1 e -1, 2 e -2, ecc.); (2) il limite $\lim_{t \rightarrow \pm\infty} \text{sinc}(t) = 0$.

Questa funzione è **importante per l'analisi nel dominio del tempo (o frequenza)**.

1.2.5 Funzione triangolo Λ

La funzione **triangolo** è definita nel seguente modo:

$$\Lambda(x) = \begin{cases} 1 - |x|, & |x| < 1 \\ 0 & \text{altrimenti} \end{cases}$$

Questa funzione è **importante per l'analisi spettrale e per le operazioni di convoluzione**.

1.2.6 Funzione segno (sgn)

La funzione **segno** è definita nel seguente modo:

$$\text{sgn}(x) = \begin{cases} -1, & x < 0 \\ +1, & x > 0 \\ 0 & x = 0 \end{cases}$$

Questa funzione ribalta segnali sopra o sotto l'asse delle x .

1.2.7 Funzione gradino

La funzione **gradino** è definita nel seguente modo:

$$u(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

Questa funzione rappresenta un **segnale** che si attiva a partire dal tempo specificato e rimane attivo indefinitamente. Attenzione! Non si confonda questo segnale con il segno.

1.2.8 Treno di impulsi

Il **treno di impulsi** $S_{\Delta T}(x)$ è la somma di un numero infinito di impulsi periodici discreti distanziati di una quantità ΔT :

$$S_{\Delta T}(x) = \sum_{n=-\infty}^{\infty} \delta(x - n\Delta T) \quad n \in \mathbb{Z}$$

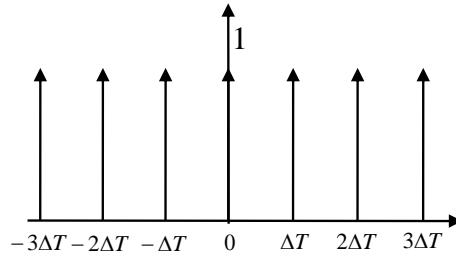


Figura 4: Treno di impulsi

1.2.9 Energia di un segnale

L'**energia di un segnale** è definita nel seguente modo:

$$E_f = \begin{cases} \int_{-\infty}^{+\infty} f^2(t) dt & \text{se } f \in \mathbb{R} \\ \int_{-\infty}^{+\infty} |f(t)|^2 dt & \text{con } |f(t)|^2 = \tilde{f}(t)f(t), \quad f \in \mathbb{C} \end{cases}$$

Un segnale si dice **ad energia finita** (o **di energia**) se l'integrale che rappresenta l'energia converge ed è diverso da 0. Quindi:

- ☞ **Condizione sufficiente** all'esistenza della sua trasformata di Fourier. Le funzioni trigonometriche non sono di energia ma hanno comunque la Trasformata di Fourier.
- ☞ **Condizione necessaria** per essere un segnale ad energia finita, all'infinito ($+\infty$ e $-\infty$) l'**ampiezza** va a zero.

Alcuni esempi:

- ★ **Segnali di energia.** Impulsi rettangolari, oscillazioni smorzate (sinc);
- ★ **Segnali non di energia.** Funzioni trigonometriche sin e cos.

L'**unità di misura** è il *joule*.

1.2.10 Potenza media di un segnale

La *potenza media di un segnale* è definita nel seguente modo:

$$P_f = \begin{cases} \lim_{T \rightarrow +\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} f^2(t) dt & \text{se } f \in \mathbb{R} \\ \lim_{T \rightarrow +\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} |f(t)|^2 dt \quad \text{con } |f(t)|^2 = \tilde{f}(t)f(t), & f \in \mathbb{C} \end{cases}$$

Un segnale si dice **a potenza finita** (o **di potenza**) se l'integrale che rappresenta la potenza converge ed è diverso da 0. L'**unità di misura** è il *watt*.

Infine, un segnale ad energia finita ha la potenza che tende a zero (per cui un segnale non può appartenere ad entrambe le categorie). Invece, esistono segnali che non sono né di energia, né di potenza finita.

1.3 Altre operazioni fondamentali

1.3.1 Rescaling (o riscalatura)

La funzione di *rescaling* è definita nel seguente modo:

$$\forall f(t) : D_1 \in \mathbb{R}, \quad \omega \neq 0$$

Simile allo *shift*, il *rescaling* ha una definizione generica e due varianti:

- **Definizione generica** con la funzione semplice $f(t)$ (immagine 5).
- **Ritardo lineare del segnale di un fattore ω** con la funzione $f(\omega t)$, $0 < \omega < 1$ (immagine 6).
- **Accelero lineare del segnale di un fattore ω** con la funzione $f(\omega t)$, $\omega > 1$ (immagine 7).

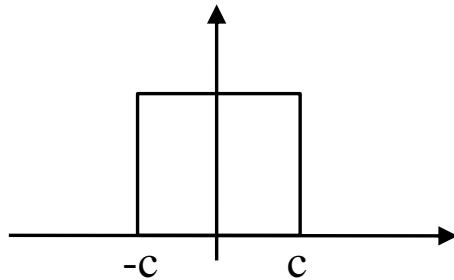


Figura 5: Definizione generica

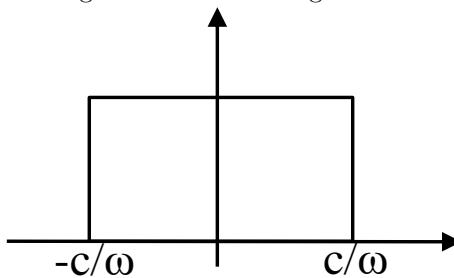


Figura 6: Ritardo lineare del segnale di un fattore ω

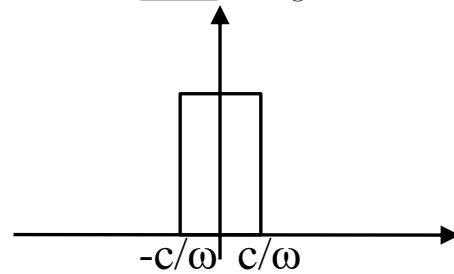


Figura 7: Accelero lineare del segnale di un fattore ω

1.3.2 Cross-Correlazione

Dati $f_1(\tau), f_2(\tau)$ segnali continui, $\tau \in \mathbb{R}$ il segnale di ***cross-correlazione*** viene definito come:

$$f_1 \otimes f_2(t) = \int_{-\infty}^{+\infty} \tilde{f}_1(\tau) f_2(\tau - t) d\tau$$

In cui $\tilde{f}_1(\tau)$ rappresenta un *complesso coniugato*. Nel caso in cui f_1 è reale, allora $\tilde{f}_1(\tau) \rightarrow f_1(\tau)$.

Infine, con $t = 0$ si ha l'***integrale di cross-correlazione***, il quale è definito se l'integrale converge (ovviamente se il segnale non è né di energia, né di potenza, la convergenza non esiste!).

1.3.3 Esercizi d'esame

Esercizio.

Il primo esercizio fornisce una funzione $f(t)$:

$$f(t) = \Pi\left(\frac{t-2}{4}\right) e^{-2t}$$

Le **richieste** dell'esercizio sono le seguenti:

- I Rappresentare graficamente il segnale;
- II Calcolare sia l'energia che la potenza media. Inoltre, dire se $f(t)$ è una funzione di energia o di potenza fornendo una motivazione valida. Infine, calcolare l'energia o la potenza nel caso in cui $f(t)$ sia solo composta da e^{-2t} ;
- III Scrivere l'espressione analitica rispetto $z(t) = -f(-t)$ e $v(t) = f(t+4)$

Risoluzione I.

Il **primo passo** è quello di scomporre la funzione così da avere una visione più chiara sulle operazioni da effettuare:

$$f(t) = \Pi\left(\frac{t-2}{4}\right) e^{-2t} \longrightarrow f(t) = \Pi\left(\frac{1}{4} \cdot (t-2)\right)$$

Come si può osservare, ci sono due operazioni da eseguire. Quindi, dopo l'esplicitazione si esegue la rappresentazione del segnale base $\Pi(t)$:

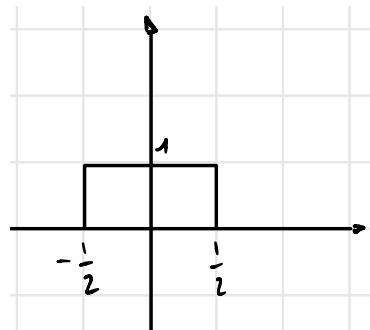


Figura 8: Rappresentazione della funzione $f(t)$, ovvero un box.

Adesso si esegue l'operazione di moltiplicazione per un fattore che in questo caso è $\frac{1}{4}$. Quindi si rappresenta la box $\Pi\left(\frac{1}{4} \cdot t\right)$:

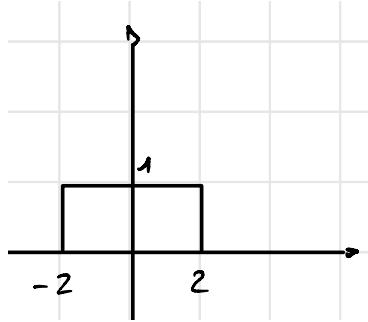


Figura 9: Box $\Pi\left(\frac{1}{4} \cdot t\right)$ allargata.

L'operazione che è stata effettuata è stata semplicemente considerare la box del tipo $\Pi\left(\frac{t}{4}\right)$. Ricordandosi le nozioni del corso di Sistemi, per definizione quindi la box è definita nell'intervallo $-2, +2$.

Infine, viene applicata l'ultima operazione, ovvero il -2 all'incognita t . Quindi, la funzione box diventerà $\Pi\left(\frac{1}{4}(t - 2)\right)$ e la sua rappresentazione grafica sarà uno shift a destra (ritardo):

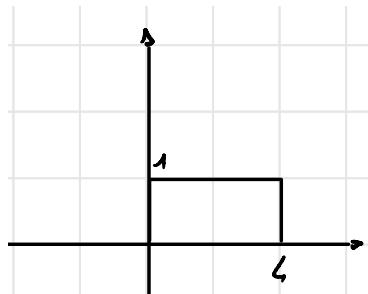


Figura 10: Box $\Pi\left(\frac{1}{4}(t - 2)\right)$ dopo lo shift a destra.

Il **primo punto si conclude** con la rappresentazione del segnale e^{-2t} e la sua combinazione con la box. Quindi:

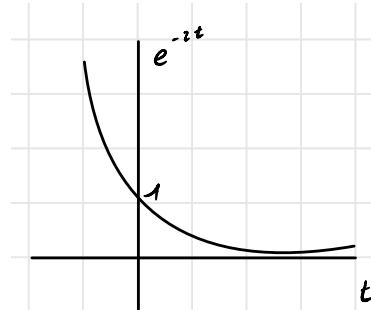


Figura 11: Rappresentazione della funzione e^{-2t}

E infine la sua concatenazione con la box, quindi una sorta di applicazione di un filtro:

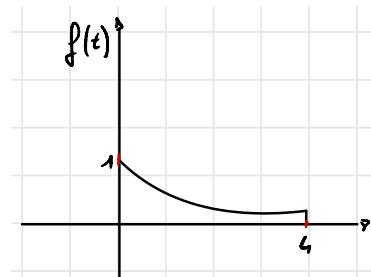


Figura 12: Rappresentazione finale della funzione $f(t) = \Pi\left(\frac{t-2}{4}\right) e^{-2t}$

Risoluzione II.

Guardando la figura 12 si può già intuire che tipo di segnale sia. Infatti, dato che è limitato e non si estende all'infinito, per definizione è un **segnale finito**, quindi **di energia e non di potenza**. Per dimostrare questa affermazione, si eseguono i calcoli:

$$\text{Definizione di energia: } E_f = \int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} f(t)^2 dt$$

$$\text{Definizione di potenza: } P_f = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f^2(t) dt$$

Dopo le definizioni, si esegue l'effettivo calcolo con i valori numerici:

Energia finita

$$E_f = \int_0^4 e^{-4t} dt = \frac{e^{-4t}}{-4} \Big|_0^4 = \frac{-e^{-16} + 1}{4} = \frac{1}{4} \neq 0$$

Potenza finita

$$P_f = \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^4 e^{-4t} dt = \lim_{T \rightarrow \infty} \frac{1}{T} \cdot \frac{1}{4} = 0$$

Come si osserva dai risultati, è un segnale di energia finita poiché è un valore noto, invece non è un segnale di potenza poiché il risultato è zero e non rispetta la definizione.

Al contrario, se la funzione fosse composta solamente dall'esponenziale, il calcolo dell'energia e della potenza sarebbe:

$$\text{Energia: } E_f = \int_{-\infty}^{\infty} e^{-4t} dt = \frac{e^{-4t}}{-4} \Big|_{-\infty}^{\infty} = \lim_{T \rightarrow \infty} \frac{e^{-4t} - e^{4t}}{-4} = \infty$$

$$\text{Potenza: } P_f = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-4t} dt = \lim_{T \rightarrow \infty} \frac{e^{-4t}}{-4} \cdot \frac{1}{T} \Big|_{-\frac{T}{2}}^{\frac{T}{2}} = \lim_{T \rightarrow \infty} \frac{e^{-2T} - e^{2T}}{-4T} = \infty$$

Come si evince dai calcoli, il segnale non è né di energia né di potenza perché entrambi i risultati sono uguali a infinito.

Risoluzione III.

Considerando la funzione $z(t)$, si osserva che è la copia simmetrica rispetto all'origine di $f(t)$. Invece, la funzione $v(t)$ è identica alla funzione $f(t)$ ma "shiftata" a sinistra di 4:

$$f(t) = -f(-t) \quad v(t) = f(t+4)$$

Esercizio 2.

Il secondo esercizio fornisce una funzione $f(t)$:

$$f(t) = \operatorname{sgn} \left(a \cdot \cos \left(\frac{2\pi}{T_0} t \right) \right)$$

Con $T_0 = 2$. Le **richieste** dell'esercizio sono le seguenti:

- I Rappresentare graficamente il segnale;
- II Calcolare sia l'energia che la potenza media. Inoltre, dire se $f(t)$ è una funzione di energia o di potenza fornendo una motivazione valida.

Risoluzione I.

Viene rappresentato il segnale della funzione segno sng:

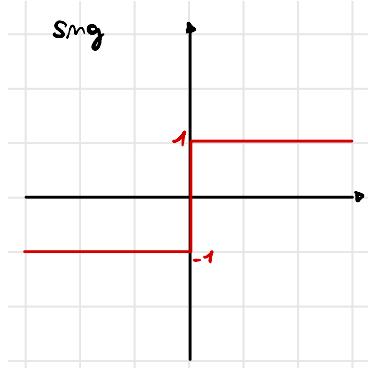


Figura 13: Funzione segno sng.

Si esplicitando le operazioni della funzione:

$$f(t) = \operatorname{sgn} \left(a \cdot \cos \left(\frac{2\pi}{T_0} t \right) \right) = \cos \left(\frac{1}{T_0} \cdot 2\pi t \right)$$

E si rappresenta inizialmente la funzione $\cos(2\pi)$ con $T_0 = 1$:

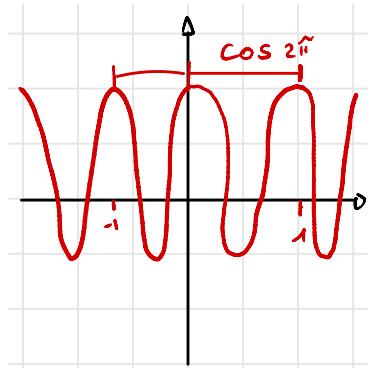


Figura 14: Funzione coseno $\cos(2\pi)$.

Si conclude la rappresentazione grafica aumentando T_0 in maniera molto semplice:

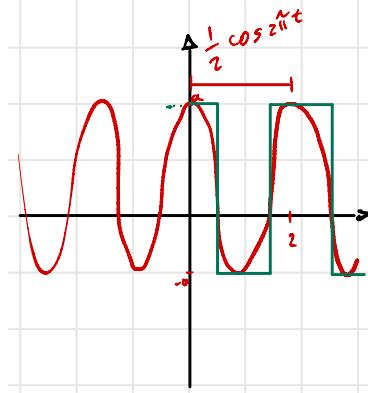


Figura 15: Funzione coseno $\cos(2\pi)$ moltiplicata per $\frac{1}{T_0} = \frac{1}{2}$.

Risoluzione II.

Si conclude l'esercizio calcolando l'energia o la potenza del segnale. Per farlo, dato che non è definito in un intervallo ma continua all'infinito, si calcolano i rispettivi integrali in un intervallo arbitrario n e poi lo si estende all'infinito:

$$\begin{aligned} E_f &= \int_{-\infty}^{\infty} f^2(t) dt = \lim_{n \rightarrow \infty} \int_{-n \cdot \frac{T_0}{2}}^{n \cdot \frac{T_0}{2}} f^2(t) dt = \lim_{n \rightarrow \infty} n \cdot \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} f^2(t) dt = \infty \\ P_f &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f^2(t) dt = \lim_{n \rightarrow \infty} \frac{1}{n T_0} \int_{-n \frac{T_0}{2}}^{n \frac{T_0}{2}} f^2(t) dt = \lim_{n \rightarrow \infty} \frac{1}{n T_0} \cdot n \cdot \int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} f^2(t) dt = \\ &= \frac{1}{T_0} \cdot T_0 = \frac{1}{2} \cdot 2 = 1 \longrightarrow \neq 0 \end{aligned}$$

È evidente che il segnale è di potenza. Come si evince dalla figura 15, i tratti di colore verde indicano il rettangolo formato dal segnale. Calcolando l'area del rettangolo, si ottiene esattamente il valore di T_0 . Infatti, la base del rettangolo (verticale) è 2, mentre l'altezza (orizzontale) è 1.

1.3.4 Cross-Correlazione Normalizzata

Ha l'**obiettivo** di trattare segnali con range di valori diversi e consente di eseguire **confronti uno-a-molti** (*one-to-many*):

$$f_1 \bar{\otimes} f_2 (t) = \frac{\int_{-\infty}^{+\infty} \tilde{f}_1 (\tau) f_2 (\tau - t) d\tau}{\sqrt{E_{f_1} E_{f_2}}}$$

In cui E_f indica l'**energia** del segnale f . Ci sono due caratteristiche importanti:

- $f_1 \bar{\otimes} f_2 (t) \in [-1, 1]$
- $|f_1 \bar{\otimes} f_2 (t)| = 1 \iff f_1 (\tau) = \alpha f_2 (\tau - t)$

Inoltre, si parla di **autocorrelazione** (normalizzata e non) quando $f_1 = f_2$. Utile per i segnali stocastici.

Nel **caso di segnali discreti**, dati $x_1(k), x_2(k)$:

$$x_1 \otimes x_2 (n) = \sum_{k=-\infty}^{+\infty} \tilde{x}_1 (k) x_2 (k - n) \quad k \in \mathbb{Z}$$

Sotto l'ipotesi di convergenza della serie, cioè la serie deve convergere.

Nel caso in cui $x_1(k)$ e $x_2(k)$ sono limitati di lunghezza M ed N rispettivamente, allora la **cross correlazione è di lunghezza $M + N - 1$** .

Cross-Correlazione 1D

Data la definizione:

$$x_1 \otimes x_2(n) = \sum_{k=-\infty}^{+\infty} x_1(k) x_2(k-n)$$

Esistono diverse casistiche:

- $n = 0$ si confronta tra x_1 e x_2 nei loro domini temporali originali.
- $n > 0$ sposta x_2 a destra poiché c'è l'anticipo di x_2
- $n < 0$ sposta x_2 a sinistra poiché c'è ritardo di x_2

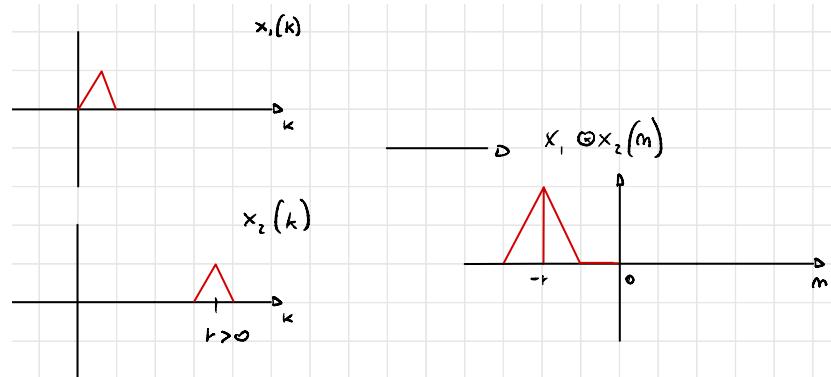


Figura 16: Esempio di cross-correlazione normalizzata 1D.

Il triangolo x_2 va verso sinistra e il lasso di tempo che x_2 non combacia con x_1 , viene rappresentato come una linea orizzontale sull'asse delle n nel piano cartesiano di destra.

Cross-Correlazione 2D

Data la definizione:

$$x_1 \otimes x_2 (m, n) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} x_1(u, v) x_2(u - m, v - n) \quad u, v, m, n \in \mathbb{Z}$$

Nel 2D x_1 e x_2 possono essere pensate come **immagini infinite**.

Di solito x_1 e x_2 sono **immagini finite** (segnali digitali ad intervallo limitato), e gli estremi di sommatoria sono quindi finiti.

Il primo segnale x_1 viene chiamato **template**, o **matrice kernel**, mentre x_2 genericamente **immagine** (di solito, la matrice kernel x_1 ha una dimensionalità minore di quella dell'immagine).

Nel caso $x_1 = x_2$ si ha **autocorrelazione 2D**.

Cross-Correlazione normalizzata 2D

Si definisce come:

$$x_1 \otimes x_2 (m, n) = \frac{\sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} [x_1(u, v)] [x_2(u - m, v - n)]}{\sqrt{\sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} [x_1(u, v)]^2 \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} [x_2(u, v)]^2}}$$

In altre parole, fissato il punto di applicazione n, m , si sottrae la media ad ogni punto nell'interno di applicazione dalla matrice kernel. Successivamente, si divide per il prodotto della varianza dei due segnali, estraendo a radice alla fine.

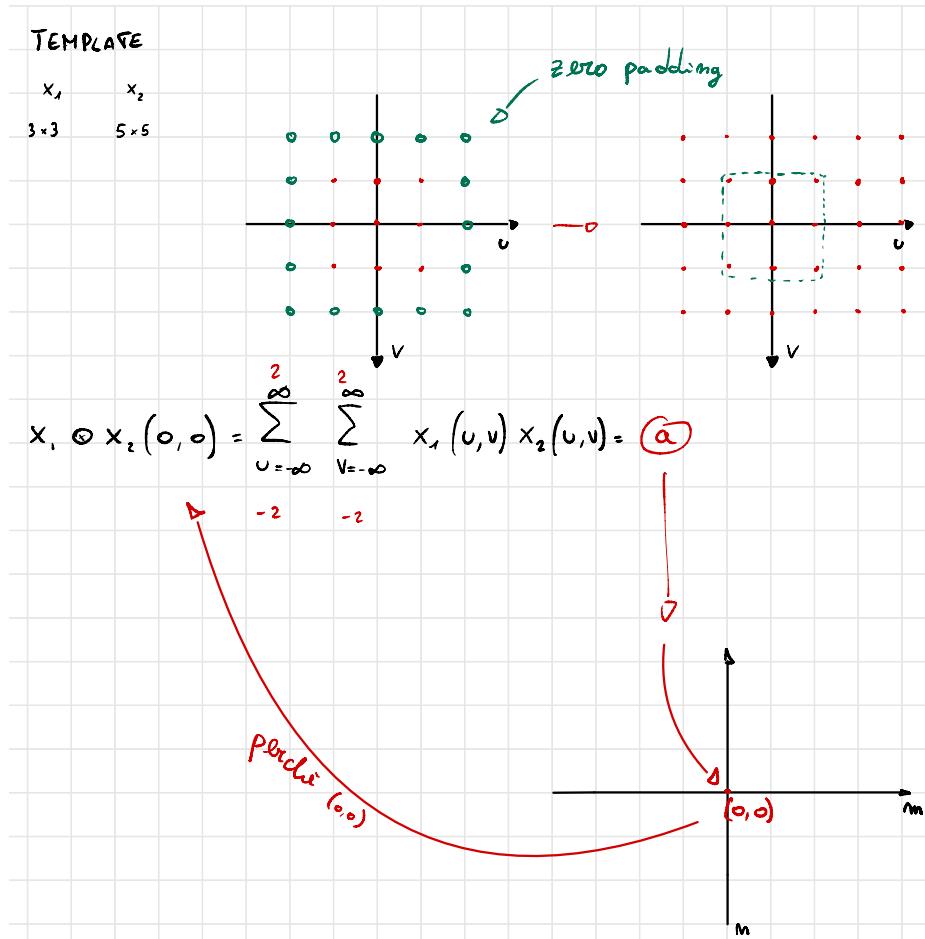


Figura 17: Esempio di Cross-Correlazione normalizzata 2D.

Esercizio Cross-Correlazione 2D

Dati le due immagini x_1 di dimensione 5×5 e x_2 di dimensione 3×3 , si calcola la cross-correlazione 2D. Quindi, si effettua la rappresentazione grafica.

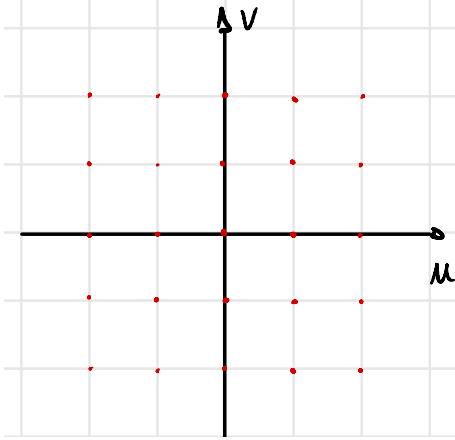


Figura 18: Piano cartesiano di x_2 di dimensione 5×5 .

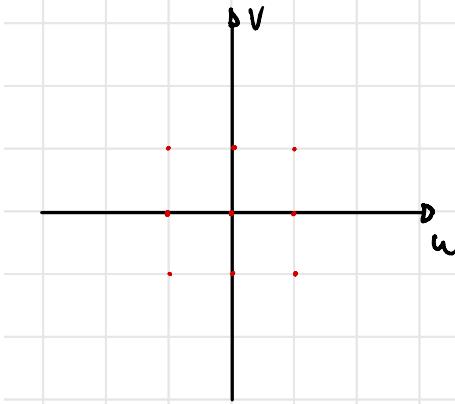


Figura 19: Piano cartesiano di x_1 di dimensione 3×3 .

E vengono fornite dall'esercizio le due matrici:

$$x_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad x_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Esse indicano i valori nei punti corrispondenti. L'**obiettivo dell'esercizio** è trovare:

- L'argomento massimo della cross-correlazione ($\arg \max x_1 \otimes x_2 (m, n)$);
- Il massimo della cross-correlazione ($\max x_1 \otimes x_2 (m, n)$).

L'argomento massimo è con i valori $m = 1$ e $n = -1$ poiché così facendo la diagonale incontra tutti i valori positivi e che formano il massimo. Infatti, prendendo in considerazione la matrice x_2 5×5 e osservando l'operazione di cross-correlazione 2D:

$$\sum_u \sum_v x_1(u, v) \cdot x_2(u - m, v - n)$$

$$\xrightarrow{\text{sostituzione termini noti } (m,n)} \sum_u \sum_v x_1(u, v) \cdot x_2(u - 1, v - (-1))$$

Risulta evidente come si debba spostare a destra, rispetto l'origine, la matrice x_2 di un solo valore¹ e sotto, rispetto sempre l'origine, di un valore negativo². Così facendo, la diagonale della matrice x_2 corrisponderà esattamente a tutti i valori 1 della matrice x_1 .

¹Shift a destra poiché $u - 1$ nell'equazione rappresenta un ritardo.

²Spostamento sotto l'asse delle ascisse poiché è un valore positivo $v + 1$.

1.3.5 Convoluzione

La **convoluzione** è un parente stretto della cross-correlazione, ma è leggermente diverso. È definito nel seguente modo:

$$f_1 * f_2(t) = \int_{-\infty}^{+\infty} f_1(\tau) f_2(t - \tau) d\tau$$

Con $t \in \mathbb{R}$. Si ricordi che se i **segnali** non sono né di energia né di potenza, l'integrale converge.

Nel caso in cui i **segnali** siano **discreti**, dati $x_1(n)$, $x_2(n)$:

$$x_1 * x_2(n) = \sum_{k=-\infty}^{+\infty} x_1(k) x_2(n - k)$$

Con $k \in \mathbb{Z}$.

Nel caso in cui $x_1(n)$ e $x_2(n)$ sono limitati di lunghezza M ed N rispettivamente, allora la **convoluzione** è di lunghezza $M + N - 1$.

Convoluzione 2D

Nel caso delle immagini, quindi del 2D, x_1 ed x_2 sono solitamente **segnali digitali ad intervallo limitato**, e la convoluzione diventa dunque:

$$x_1 * x_2(m, n) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} x_1(u, v) x_2(m - u, n - v) \quad u, v, m, n \in \mathbb{Z}$$

Solitamente il primo segnale x_1 viene chiamato **filtro**, o **matrice kernel**, mentre x_2 genericamente **immagine** (solitamente la matrice kernel ha una dimensione inferiore di quella dell'immagine).

2 Analisi di Fourier

2.1 Serie di Fourier

Una funzione, chiamata **funzione di sintesi**, $f : \mathbb{R} \rightarrow \mathbb{R}$ di variabile continua t , periodica di periodo T , si esprime come:

$$f(t) = \sum_{n=-\infty}^{+\infty} c_n \underbrace{e^{j \frac{2\pi n}{T} t}}_{\text{fasore}} \quad n \in \mathbb{Z}$$

Dove c_n è un numero complesso. Invece, una **funzione di analisi** è espressa come:

$$c_n \in \mathbb{C} = \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} f(t) \underbrace{e^{-j \frac{2\pi n}{T} t}}_{\text{fasore}} dt \quad n \in \mathbb{Z}$$

N.B. si ricorda che $e^{j \frac{2\pi n}{T} t}$ è un **fasore rotante** di velocità angolare $\frac{2\pi n}{T} t$.

La **funzione di sintesi** quindi non è altro che una somma di infiniti termini. Ciascuno è composto dalla moltiplicazione tra un numero complesso ed un fasore, il quale *produce un altro fasore*. Esprimendo c_n come numero complesso in forma polare:

$$c_n e^{j \frac{2\pi n}{T} t} = |c_n| e^{j \theta_n} e^{j \frac{2\pi n}{T} t} = |c_n| e^{j(\frac{2\pi n}{T} t + \theta_n)}$$

Si può notare come questa conversione corrisponda ad **estendere** il fasore $e^{j \frac{2\pi n}{T} t}$ ad una lunghezza $|c_n|$ facendolo partire con un **angolo di partenza** uguale a θ_n (chiamato **angolo di fase**).

Altra osservazione: se c_n appartiene all'insieme \mathbb{R} , significa che θ_n non compare. Questo comporta un cambiamento nella lunghezza dell' n -esimo fasore pari a $|c_n|$:

$$c_n = |c_n| e^{j \theta_n}$$

Esempio 1

Il primo esempio di serie di Fourier si applica per il segnale trigonometrico:

$$f(t) = \cos(2\pi t) \quad \text{con } T = 1$$

Applicando la **funzione di analisi** e saltando i passaggi perché complessi, si ottengono i seguenti valori:

$$c_{-1} = \frac{1}{2} \quad c_0 = 0 \quad c_1 = \frac{1}{2} \quad c_{i \leq -2, i \geq 2} = 0$$

E sostituendo nella **funzione di sintesi**:

$$\cos(2\pi t) = \frac{1}{2}e^{-j2\pi t} + \frac{1}{2}e^{j2\pi t} = \frac{e^{j2\pi t} + e^{-j2\pi t}}{2}$$

Ci sono **tre osservazioni** da fare:

I. $\frac{2\pi}{T} = f_0;$

II. $c_n = |c_n|e^{j\theta_n};$

III. In questo caso, $c_n \in \mathbb{R}$ quindi l'angolo di fase non è presente.

Le parti dell'equazione sono le seguenti:

$$\cos(2\pi t) = \frac{1}{2}e^{-j2\pi t} + \frac{1}{2}e^{j2\pi t}$$

☞ $\cos(2\pi t) \rightarrow$ La funzione trigonometrica da studiare

☞ $\frac{1}{2}e^{-j2\pi t} \rightarrow$ Fasore di modulo 0.5 e velocità angolare $-2\pi t$

☞ $\frac{1}{2}e^{j2\pi t} \rightarrow$ Fasore di modulo 0.5 e velocità angolare $2\pi t$

I coefficienti $c_{n=-1}$ e $c_{n=1}$ sono relativi ai **moduli o ampiezze dei fasori** complessi di frequenza $f_0 \cdot n$ con $n = -1, 1$ e ricordando che:

$$\exp\left(j\left(\frac{2\pi n}{T}t\right)\right) = \exp(j(f_0 nt))$$

Che si possono annotare con le variabili f_{-1} e f_1 per $f_0 \cdot n$ con $n = -1, 1$ e analogamente per gli altri $n \in \mathbb{Z}$.

Inoltre, è possibile disegnare lo **spettro di ampiezza** che **mostra i moduli dei fasori costruiti con la trasformata di Fourier**, in particolare la funzione di sintesi.

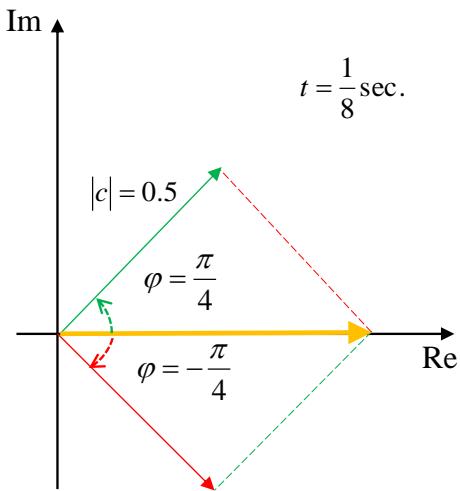


Figura 20: Grafico rappresentante i due fasori. La freccia verde rappresenta il valore assunto da $\cos(2\pi t)$ per $t = \frac{1}{8}$.

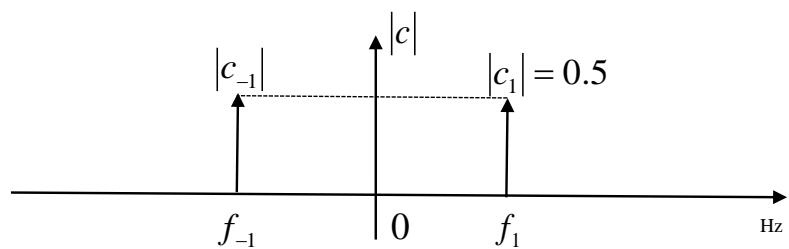


Figura 21: Grafico che rappresenta lo spettro di ampiezza.

Esempio 2

Il secondo esempio di serie di Fourier è il segnale trigonometrico:

$$f(t) = \sin(2\pi t) \quad \text{con } T = 1$$

Applicando la **funzione di analisi** e saltando i passaggi perché complessi, si ottengono i seguenti valori:

$$c_{-1} = -\frac{1}{2j} \quad c_0 = 0 \quad c_1 = \frac{1}{2j} \quad c_{i \leq -2, i \geq 2} = 0$$

Dove questa volta $c_n \in \mathbb{C}$ ed in particolare:

$$\pm \frac{1}{2j} = \pm \frac{1}{2j} \cdot \frac{j}{j} = \pm \frac{1}{2} \cdot \frac{j}{j^2} = j \cdot \mp \frac{1}{2}$$

Si passa alla forma di esponenziale complesso:

$$\begin{aligned} j \cdot \frac{1}{2} &= 0 + j \cdot \frac{1}{2} \\ |c| &= \sqrt{0^2 + \left(\frac{1}{2}\right)^2} = \frac{1}{2} \\ \theta &= \arctan\left(\frac{0.5}{0}\right) \rightarrow \frac{\pi}{2} \\ \frac{1}{2}e^{j \cdot \frac{\pi}{2}} &= c_{-1} \end{aligned}$$

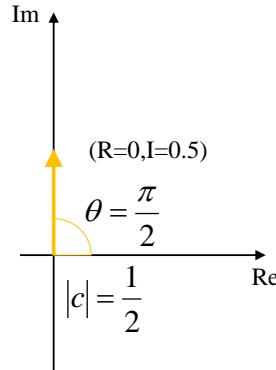


Figura 22: Grafico di c_{-1} .

Analogamente:

$$\begin{aligned}j \cdot -\frac{1}{2} &= 0 + j \cdot \left(-\frac{1}{2}\right) \\|c| &= \sqrt{0^2 + \left(-\frac{1}{2}\right)^2} = \frac{1}{2} \\\theta &= \arctan\left(-\frac{0.5}{0}\right) \rightarrow -\frac{\pi}{2} \\\frac{1}{2}e^{j \cdot \left(-\frac{\pi}{2}\right)} &= c_1\end{aligned}$$

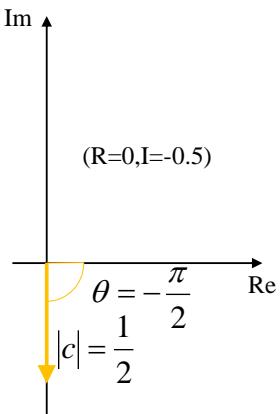


Figura 23: Grafico di c_1 .

Applicando l'**equazione di sintesi** e sostituendo i termini noti:

$$\begin{aligned}\sin(2\pi t) &= \sum_{n=-\infty}^{+\infty} c_n e^{j \frac{2\pi n}{T} t} = c_{-1} e^{j -2\pi t} + c_1 e^{j 2\pi t} \\ \text{sostituzione dei termini noti } c_{-1}, c_1 &\Rightarrow \frac{1}{2} e^{j \frac{\pi}{2}} e^{j \cdot (-2\pi t)} + \frac{1}{2} e^{j -\frac{\pi}{2}} e^{j 2\pi t} \\ \text{forma finale} &\Rightarrow \frac{1}{2} \exp\left(j\left(-2\pi t + \frac{\pi}{2}\right)\right) + \exp\left(j\left(2\pi t - \frac{\pi}{2}\right)\right)\end{aligned}$$

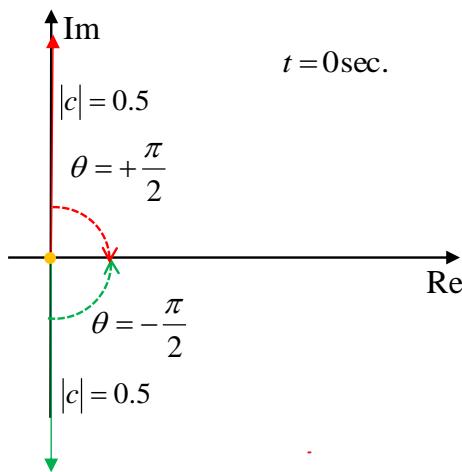


Figura 24: Grafico finale.

Infine, si disegna lo **spettro di ampiezza** e lo **spettro di fase**, quest'ultimo è un **grafico in cui si riportano gli angoli di fase della funzione**.

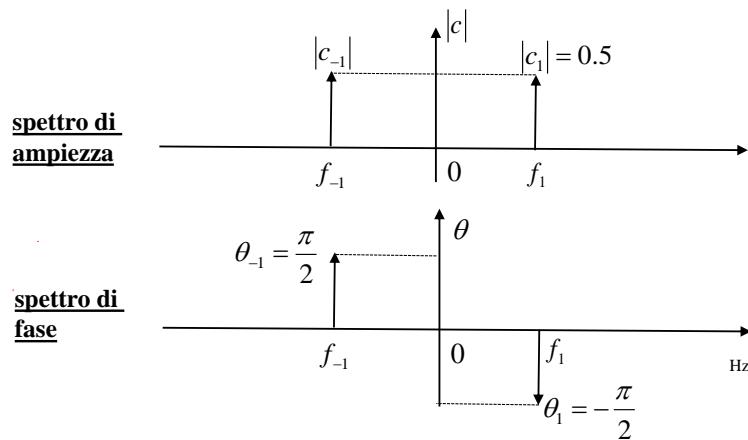


Figura 25: Spettro di ampiezza e di fase della funzione $\sin(2\pi t)$.

2.1.1 Proprietà della serie di Fourier

Lo **spettro di ampiezza e di fase** sono funzioni nel dominio delle frequenze che formano lo **spettro di Fourier**. Lo spettro di Fourier per i segnali periodici gode delle **seguenti proprietà**:

- Lo **spettro di ampiezza** è *simmetrico* rispetto all'asse y ;
- Lo **spettro di fase** è *antisimmetrico* rispetto all'asse y ;
- Se i coefficienti c_n sono reali, **non esiste lo spettro di fase**;
- Entrambe gli spettri sono funzioni a pettine, definite su frequenze multiple rispetto a quella fondamentale:

$$\left\{ \frac{2\pi n}{T} \right\}_{n \in \mathbb{Z}} = \{f_0 \cdot n\}_{n \in \mathbb{Z}} \equiv \{f_n\}_{n \in \mathbb{Z}}$$

2.2 Trasformata di Fourier continua

2.2.1 Trasformata di Fourier

Sia $f(t)$ un **segnale reale continuo** $f : \mathbb{R} \rightarrow \mathbb{R}$ periodico o non, si definisce la **Trasformata di Fourier** (TdF) $\mathcal{F}(f(t)) = F(\mu)$ il segnale $\mathcal{F} : \mathbb{R} \rightarrow \mathbb{C}$:

$$\mathcal{F}(f(t)) = F(\mu) = \int_{-\infty}^{+\infty} f(t) e^{-j2\pi\mu t} dt$$

L'**unità frequenziale** μ è l'angolo di $\frac{n}{T}$ della serie di Fourier (per esempio, con $n = 1$, $T = 1$ sec. $\rightarrow \mu = 1$ sec. $^{-1}$ = 1 Hz).

La Trasformata di Fourier esiste se $f(t)$ è un **segnale di energia**. Condizione sufficiente e non necessaria perché altri segnali ammettono la TdF.

2.2.2 Trasformata di Fourier inversa

Sia $F(\mu)$ la trasformata di Fourier di un segnale $f : \mathbb{R} \rightarrow \mathbb{R}$. Si definisce la **trasformata di Fourier inversa** il segnale $\mathcal{F}^{-1}(F(\mu)) = f(t)$:

$$\mathcal{F}^{-1} = (F(\mu)) = f(t) = \int_{-\infty}^{+\infty} F(\mu) e^{j2\pi\mu t} d\mu$$

La trasformata di Fourier restituisce, per una data frequenza μ , un coefficiente di “presenza” $F(\mu)$. Infatti, la sua inversa permette di ricostruire f a partire da F .

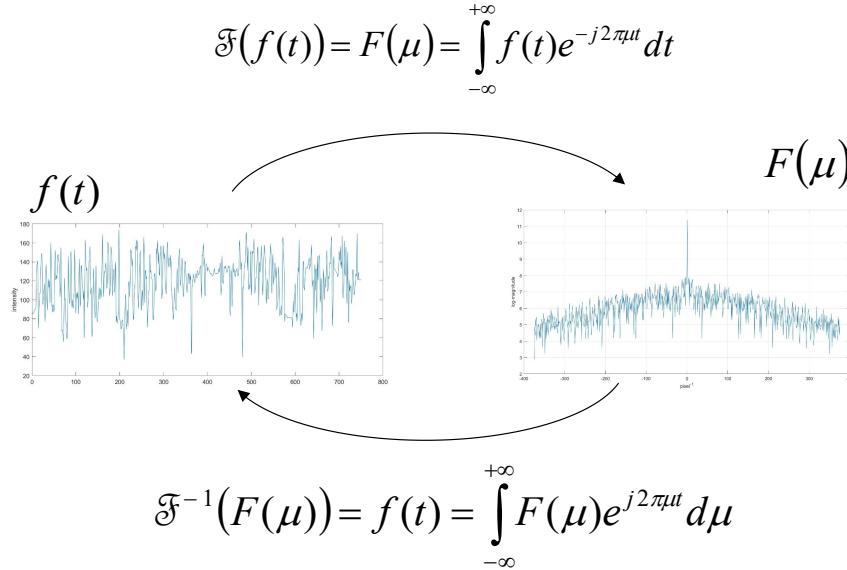


Figura 26: (Anti)Trasformata di Fourier su un segnale $f(t)$.

Nel caso in cui il segnale $f(t)$ non è reale, la trasformata è complessa:

- t rappresenta il **tempo** (in secondi), allora μ rappresenta gli **Hertz**, cioè $\frac{\text{numero cicli}}{\text{secondi}}$;
- t rappresenta lo **spazio** (in metri), allora μ rappresenta la **frequenza spaziale**, cioè $\frac{\text{numero cicli}}{\text{metri}}$

Mentre nella serie di Fourier le funzioni rappresentate negli spettri di ampiezza e di fase erano a “pettine” (paragrafo 2.1.1), in questo caso le funzioni sono solitamente continue, nello spettro di ampiezza, o continue a tratti:

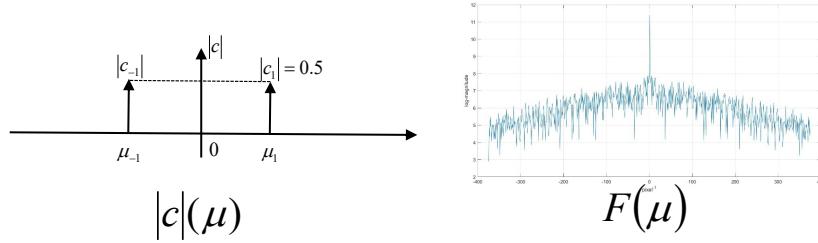


Figura 27: Esempio di spettro di ampiezza.

2.2.3 Proprietà della trasformata di Fourier

☞ Linearità

$$a_1 f_1(t) + a_2 f_2(t) \xrightarrow{\mathcal{F}} a_1 F_1(\mu) + a_2 F_2(\mu)$$

☞ Scalatura temporale

$$z(t) = f(at) \xrightarrow{\mathcal{F}} Z(\mu) = \frac{1}{a} F\left(\frac{\mu}{a}\right)$$

☞ Dualità

$$\begin{aligned} f(t) &\xrightarrow{\mathcal{F}} F(\mu) \\ F(t) &\xrightarrow{\mathcal{F}^-} f(-\mu) \end{aligned}$$

N.B. derivando la forma analitica per una trasformata, la sua antitrasformata ne produce un'altra con segno opposto.

☞ Time shift

$$\begin{aligned} \mathcal{F}(f(t - t_0)) &= \int_{-\infty}^{+\infty} f(t - t_0) e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{+\infty} f(u) e^{-j2\pi\mu(u+t_0)} du \\ &= \int_{-\infty}^{+\infty} f(u) e^{-j2\pi\mu u} e^{-j2\pi\mu t_0} du \\ &= e^{-j2\pi\mu t_0} \int_{-\infty}^{+\infty} f(u) e^{-j2\pi\mu u} du \\ &= F(\mu) \underbrace{e^{-j2\pi\mu t_0}}_{\text{phase}} \end{aligned}$$

2.2.4 Trasformata di Fourier di una box

La trasformata di Fourier di una box (paragrafo 1.2.3) è la seguente:

$$\mathcal{F}(f(t)) = \int_{-\infty}^{+\infty} A\Pi\left(\frac{t}{w}\right) e^{-j2\pi\mu t} dt = F(\mu)$$

Il risultato corrisponde alla funzione sinc:

$$f(\mu) = Aw \cdot \text{sinc}(\mu w)$$

Dove la funzione sinc è uguale a:

$$\text{sinc} = \frac{\sin(\pi\mu w)}{\pi\mu w}$$

Per ripassare la funzione sinc, si rimanda al paragrafo 1.2.4. Tuttavia, si ricorda che la sua forma generale è del tipo:

$$\text{sinc}(m) = \frac{\sin(\pi m)}{\pi m}$$

E risultata uguale a:

- $\text{sinc}(0) = 1$
- $\text{sinc}(m) = 0 \quad \forall m \in \mathbb{Z}$

Prima di concludere, si ricorda che:

- ☞ All'aumentare della larghezza della box, la funzione sinc tenderà a stringersi;
- ☞ La box è **limita**, invece la sinc è **infinita** a destra e sinistra, anche se il termine al denominatore attenua il valore della funzione comportando un limite a 0.
- ☞ In sintesi, la TdF di una box è:

$$\Pi\left(\frac{t}{w}\right) \xrightarrow{\mathcal{F}} w \cdot \text{sinc}(\mu w)$$

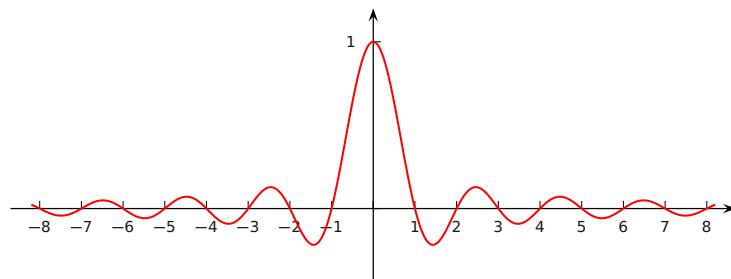


Figura 28: Grafico della funzione sinc.

2.2.5 Trasformata di Fourier di un sinc

La trasformata di Fourier di un segnale sinc (segnale rappresentato in figura 28) è la seguente:

$$\mathcal{F}(f(t)) = \int_{-\infty}^{+\infty} \text{sinc}(tw) e^{-j2\pi\mu t} dt = F(\mu)$$

Dato che la TdF di una box è:

$$\Pi\left(\frac{t}{w}\right) \xrightarrow{\mathcal{F}} w \cdot \text{sinc}(\mu w)$$

Al contrario, si ottiene la trasformata di Fourier di un sinc:

$$\text{sinc}(tw) \xrightarrow{\mathcal{F}} \frac{1}{w} \Pi\left(-\frac{\mu}{w}\right) = \frac{1}{w} \cdot \Pi\left(\frac{\mu}{w}\right)$$

2.2.6 Trasformata di Fourier di un impulso

La trasformata di Fourier di un impulso³ è la seguente:

$$\mathcal{F}(f(t)) = F(\mu) = \int_{-\infty}^{+\infty} \delta(t) e^{-j2\pi\mu t} dt$$

Il risultato della trasformata di Fourier di un impulso è molto semplice grazie alle sue proprietà. Infatti, il risultato è uguale a:

$$\int_{-\infty}^{+\infty} \delta(t) e^{-j2\pi\mu t} dt = \int_{-\infty}^{+\infty} \delta(0) e^{-j2\pi\mu 0} dt = 1$$

La proprietà che consente di ottenere il risultato uguale a 1 è la seguente:

$$\delta(t) = \begin{cases} \infty & \text{se } t = 0 \\ 0 & \text{se } t \neq 0 \end{cases} \quad \rightarrow \quad \int_{-\infty}^{+\infty} \delta(t) dt = 1$$

N.B. In questo caso è rappresentabile solo lo spettro di ampiezza!

Analogamente, con un impulso centrato in t_0 , quindi non nell'origine:

$$\mathcal{F}(f(t)) = F(\mu) = \int_{-\infty}^{+\infty} \delta(t - t_0) e^{-j2\pi\mu t} dt = e^{-j2\pi\mu t_0}$$

Il risultato è stato ottenuto grazie alla proprietà di setacciamento (definita a pagina 12). Tuttavia, in questo caso i valori non sono più reali ma complessi.

³Definizione di impulso al paragrafo 1.2.3.

2.2.7 Trasformata di Fourier di un treno di impulsi

Data la definizione di treno di impulsi (funzione definita nel paragrafo 1.2.8):

$$S_{\Delta T}(t) = \sum_{n=-\infty}^{+\infty} \delta(t - n\Delta T) \quad n \in \mathbb{Z}$$

Si ottiene la sua relativa trasformata di Fourier:

$$\mathcal{F}(S_{\Delta T}(t)) = \int_{-\infty}^{+\infty} S_{\Delta T}(t) e^{-j2\pi\mu t} dt = F(\mu)$$

Tralasciando i vari calcoli numerici per arrivare al risultato, si può scrivere la trasformata di Fourier in maniera più semplice:

$$S_{\Delta T}(t) \xrightarrow{\mathcal{F}} \sum_{n=-\infty}^{+\infty} \frac{1}{\Delta T} \delta\left(\mu - \frac{n}{\Delta T}\right)$$

2.2.8 Sintesi

Qui di seguito si lascia un riassunto rapido delle trasformate di Fourier continue dei segnali più importanti.

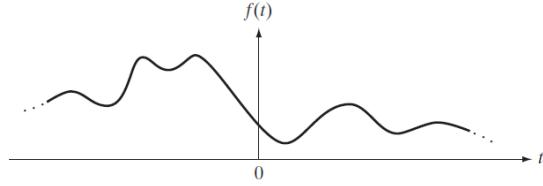
Segnale	Trasformata di Fourier	
Box:	$A\Pi\left(\frac{t}{w}\right)$	$\xrightarrow{\mathcal{F}}$ $Aw \cdot \text{sinc}(\mu w)$
Sinc:	$\text{sinc}(tw)$	$\xrightarrow{\mathcal{F}}$ $\frac{1}{w} \cdot \Pi\left(-\frac{\mu}{w}\right) = \frac{1}{w} \cdot \Pi\left(\frac{\mu}{w}\right)$
Impulso:	$\delta(t)$	$\xrightarrow{\mathcal{F}}$ $\begin{cases} 1 & \text{se valori reali} \\ e^{-j2\pi\mu t_0} & \text{se valori complessi} \end{cases}$
Treno di impulsi:	$S_{\Delta T}(t)$	$\xrightarrow{\mathcal{F}}$ $\sum_{n=-\infty}^{+\infty} \frac{1}{\Delta T} \cdot \delta\left(\mu - \frac{n}{\Delta T}\right)$

Tabella 1: Trasformate di Fourier continue.

2.3 Trasformata di Fourier a tempo discreto

2.3.1 Campionamento

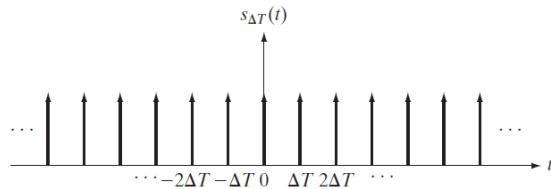
Sia $f(t)$ un segnale reale continuo definito $f :]-\infty, +\infty[\in \mathbb{R} \rightarrow \mathbb{R}$ (attenzione al dominio non limitato), anche non periodico:



Questo tipo di segnale, per essere elaborato al computer deve essere **campionato** ad intervalli discreti. Per farlo, si prenda in considerazione il treno di impulsi:

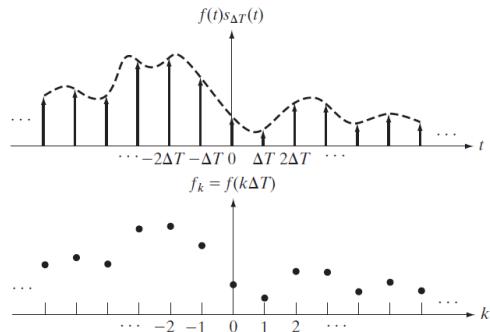
$$S_{\Delta T}(t) = \sum_{n=-\infty}^{+\infty} \delta(t - n\Delta T)$$

Con periodo ΔT , ossia con **frequenza di campionamento** pari a: $\mu_S = \frac{1}{\Delta T}$



Si assume che il treno di impulsi sia un **segnale discreto**. Matematicamente parlando, **campionare un segnale** significa moltiplicarlo per un treno di impulsi:

$$\tilde{f}(t) = f(t) \cdot S_{\Delta T}(t) = f(t) \cdot \sum_{n=-\infty}^{+\infty} \delta(t - n\Delta T)$$



2.3.2 Trasformata di Fourier a tempo discreto

Sia $F(\mu)$ la trasformata di Fourier di un segnale $f(t) : \mathbb{R} \rightarrow \mathbb{R}$. Si considera $\tilde{f}(t)$ e si calcola la trasformata di Fourier $\tilde{F}(\mu)$ (entrambi sono a tempo discreto). Grazie alla convoluzione si ottiene:

$$\tilde{F}(\mu) = \mathcal{F}\left\{\tilde{f}(t)\right\} = F(\mu) * S_{\Delta T}(\mu)$$

Si ricorda che:

$$S_{\Delta T}(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{+\infty} \delta\left(\mu - \frac{n}{\Delta T}\right)$$

E risolvendo la convoluzione, si ottiene che la **trasformata di Fourier a tempo discreto** corrisponde a:

$$F(\mu) * S_{\Delta T}(\mu) = \int_{-\infty}^{+\infty} F(\tau) \cdot S_{\Delta T}(\mu - \tau) d\tau = \frac{1}{\Delta T} \sum_{n=-\infty}^{+\infty} F\left(\mu - \frac{n}{\Delta T}\right)$$

Analizzando la formula si evidenziano alcuni termini:

- $F(\mu)$ è la trasformata di Fourier della funzione originale $f(t)$;
- $F\left(\mu - \frac{n}{\Delta T}\right)$ è la trasformata di Fourier della funzione originale $f(t)$ shiftato a destra di una quantità pari a $\frac{n}{\Delta T}$;
- $\frac{1}{\Delta T} \sum_{n=-\infty}^{+\infty} F\left(\mu - \frac{n}{\Delta T}\right)$ sono infinite copie dello spettro $F(\mu)$, ripetute ovviamente ogni $\frac{1}{\Delta T}$.

Inoltre, è un **segnale periodico** (nelle frequenze) di periodo $\frac{1}{\Delta T}$, ovvero si ripete ogni $\frac{1}{\Delta T}$ Hz.

La sua scalatura nell'ampiezza è pari a $\frac{1}{\Delta T}$ e rappresenta la T.d.F. a tempo discreto.

2.3.3 Teorema del campionamento

Un segna~~le reale continuo~~ f(t), limitato in banda, può essere ricostruito senza errori completamente dai suoi campioni se essi sono acquisiti con un tempo di campionamento ΔT tale per cui:

$$\frac{1}{\Delta T} = \mu_S \geq 2\mu_{\max}$$

Ovvero se nel tempo si adotta una frequenza di campionamento $\frac{1}{\Delta T}$ almeno doppia rispetto alla frequenza massima del segnale μ_{\max} .

In altre parole, il teorema del campionamento afferma che tutte le proprietà di un segnale possono essere espresse usando dei campioni.

Attenzione! L'espressione $\frac{1}{\Delta T}$ viene chiamata *frequenza di Nyquist* e per frequenze minori si crea aliasing, fenomeno che impedisce la ricostruzione senza errori.

2.3.4 Considerazioni

Dal punto di vista teorico la trasformata di Fourier a tempo discreto consente di ricostruire il segnale. Tuttavia, è impossibile implementarla in un computer poiché tende, come limiti, all'infinito e ci vorrebbe un numero infinito di campioni e di segnali di tipo sinc.

2.4 Trasformata di Fourier discreta

La trasformata di Fourier di un segnale reale continuo $f(t)$ di dominio illimitato e non periodico, campionato nel tempo con periodo di campionamento ΔT , è una funzione continua, periodica (di periodo $\frac{1}{\Delta T}$) anch'essa di dominio illimitato:

$$\tilde{f}(t) = f(t) \cdot \sum_{n=-\infty}^{+\infty} \delta(t - n\Delta T) \xrightarrow{\text{F}} \tilde{F}(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{+\infty} F\left(\mu - \frac{n}{\Delta T}\right)$$

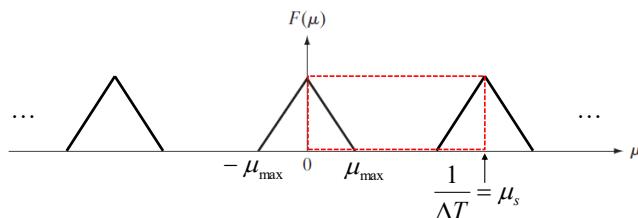
Il **problema** di questa formulazione è data l'espressione analitica dello spettro, la quale suppone che si è a **conoscenza della T.d.F. teorica F del segnale di partenza**. Questo, spesso, è molto difficile.

Si ricava dunque una *forma più semplice* da manipolare. Essa consente di **costruire una rappresentazione spettrale a partire dai campioni della funzione originale $f(t)$** :

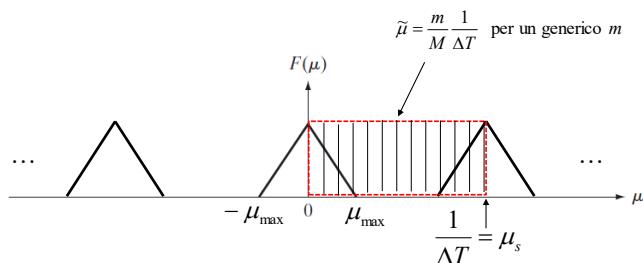
$$\tilde{F}(\mu) = \sum_{n=-\infty}^{+\infty} f_n e^{-j2\pi\mu n \Delta T}$$

Al contrario, la prima formulazione era più chiara per comprendere il fatto che la T.d.F. di una funzione campionata genera delle repliche dello spettro originale $F(\mu)$.

L'equazione alternativa deve essere modificata, eseguendo un campionamento per il dominio spettrale, per poterla implementare su un computer. Per farlo, si prende in considerazione solo l'intervallo frequenziale da 0 a $\frac{1}{\Delta T} = \mu_s$.



Inoltre, si prendono in considerazione M campioni tramite l'operazione di campionamento.



In cui:

$$\tilde{\mu} = \frac{m}{M} \cdot \frac{1}{\Delta T} \quad \text{con } m = 0, \dots, M - 1 \text{ e dove } \frac{m}{M} \in \left[0, 1 - \frac{1}{M}\right]$$

La m indica il **range di variazione** degli indici dei campioni frequenziali. Calcolando la trasformata di Fourier a tempo discreto sui campioni M , si giunge alla **forma finale della trasformata di Fourier discreta**:

$$\tilde{F}(\tilde{\mu}) = \tilde{F}\left(\frac{m}{M} \cdot \frac{1}{\Delta T}\right) = \sum_{n=0}^{M-1} f_n e^{-j2\pi \frac{m}{M} n} \quad \text{con } m = 0, \dots, M - 1$$

La **trasformata di Fourier discreta inversa**, ovvero l'antitrasformata:

$$\tilde{f}(n\Delta T) = f(n\Delta T) = f_n = \frac{1}{M} \sum_{m=0}^{M-1} F_m e^{j2\pi \frac{m}{M} n}$$

2.5 Riassunto Trasformate

Qui di seguito vengono rappresentate le trasformate più importanti.

Funzione	Serie di Fourier
Funzione di sintesi:	$f(t) \quad \sum_{n=-\infty}^{+\infty} c_n \underbrace{e^{j \frac{2\pi n}{T} t}}_{\text{fasore}} \quad n \in \mathbb{Z}$
Funzione di analisi:	$c_n \in \mathbb{C} \quad \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} f(t) \underbrace{e^{-j \frac{2\pi n}{T} t}}_{\text{fasore}} dt \quad n \in \mathbb{Z}$
Segnale	Trasformata di Fourier continua
Box:	$A\Pi\left(\frac{t}{w}\right) \xrightarrow{\mathcal{F}} Aw \cdot \text{sinc}(\mu w)$
Sinc:	$\text{sinc}(tw) \xrightarrow{\mathcal{F}} \frac{1}{w} \cdot \Pi\left(-\frac{\mu}{w}\right) = \frac{1}{w} \cdot \Pi\left(\frac{\mu}{w}\right)$
Impulso:	$\delta(t) \xrightarrow{\mathcal{F}} \begin{cases} 1 & \text{se valori reali} \\ e^{-j2\pi\mu t_0} & \text{se valori complessi} \end{cases}$
Treno di impulsi:	$S_{\Delta T}(t) \xrightarrow{\mathcal{F}} \sum_{n=-\infty}^{+\infty} \frac{1}{\Delta T} \cdot \delta\left(\mu - \frac{n}{\Delta T}\right)$
Funzione	Trasformata di Fourier a tempo discreto
$F(\mu) * S_{\Delta T}(\mu)$	$\frac{1}{\Delta T} \sum_{n=-\infty}^{+\infty} F\left(\mu - \frac{n}{\Delta T}\right)$
Funzione	Trasformata di Fourier discreta
$\tilde{F}(\tilde{\mu})$	$\tilde{F}\left(\frac{m}{M} \cdot \frac{1}{\Delta T}\right) = \sum_{n=0}^{M-1} f_n e^{-j2\pi \frac{m}{M} n}$
	$m = 0, \dots, M-1$
Funzione	Trasformata di Fourier discreta inversa
$\tilde{f}(n\Delta T) = f(n\Delta T) = f_n$	$\frac{1}{M} \sum_{m=0}^{M-1} F_m e^{j2\pi \frac{m}{M} n}$

Tabella 2: Trasformate di Fourier.

2.6 Domanda da esame

All'esame è possibile che sia richiesto come domanda: "quali sono le trasformate di Fourier studiate durante il corso?"

La risposta, anche se banale, è la seguente: le trasformate di Fourier studiate durante il corso sono 4:

- I. Serie di Fourier (paragrafo 2.1)
- II. Trasformata di Fourier continua (paragrafo 2.2)
- III. Trasformata di Fourier a tempo discreto (paragrafo 2.3)
- IV. Trasformata di Fourier discreta (paragrafo 2.4)

3 Elaborazione di immagini - Dominio spaziale

L'**elaborazione delle immagini** consiste nel prendere come input un'immagine (segnalet) e restituirne un'altra (sempre segnale) come output.

Il **rinforzo (enhancement) di immagini** è un tipo di elaborazione delle immagini. Il suo **obbiettivo** è elaborare un'immagine in modo che il risultato sia più adatto alle esigenze soggettive dell'utente. La definizione è *problem-oriented* poiché, per esempio, per visualizzare lo spettro di ampiezza di un'immagine, è necessario eseguire un'operazione di rinforzo (*log-transformation*); oppure, per migliorare la visibilità dei dettagli di un'immagine, si effettua un'altra operazione di rinforzo (*sharpening*).

La **qualità di un'immagine** è la combinazione pesata di tutti gli attributi significativi di un'immagine. Infatti, *non* esiste una ricetta univoca per determinare quando un'immagine sia di qualità poiché è un'opinione soggettiva. Tuttavia, è più facile dire quando un'immagine *non* è di qualità. In genere, un'**immagine non è di qualità** quando non viene interpretata facilmente da un operatore umano.

A differenza del rinforzo, il **restauro (restoration)** è un processo di ricostruzione dell'immagine a partire da un modello di degradazione noto.

3.1 Strumento per l'elaborazione: istogramma

I pixel di un'immagine sono una “popolazione” sulla quale è possibile calcolare tutte le quantità statistiche descrittive che vengono usate normalmente come media, mediana, varianza, deviazione standard, quartili, percentili, etc. Uno **strumento fondamentale** per l'elaborazione delle immagini è l'**istogramma**, il quale può essere visto come una funzione continua o discreta.

Infatti, per ogni livello di grigio (in un'immagine solo a livelli di grigi) vengono riportati il numero di pixel. Per un'immagine $I [M, N]$ si identifica con M, N il **numero di pixel righe per colonne** e con la funzione $H(r)$ il **numero di pixel di valore r** , quest'ultimo è definito nell'intervallo $0 \leq r \leq L - 1$ con $r, L \in \mathbb{N}$ dove L indica i livelli di grigio. Inoltre:

$$\sum_{r=0}^{L-1} H(r) = M \cdot N$$

Grazie all'istogramma, è possibile comprendere immediatamente le caratteristiche dell'immagine (come in figura).



Figura 29: Esempio di istogramma.

Un istogramma può essere anche visto come una distribuzione di probabilità:

$$p_h(r) = \frac{H(r)}{M \cdot N} \quad \sum_r p_h(r) = 1$$

Uno **svantaggio** di questo strumento è che immagini diverse potrebbero avere istogrammi simili, questo perché l'istogramma non tiene conto della distribuzione spaziale dei pixel. Dunque, **utilizzando solo questo metodo è impossibile ricostruire un'immagine**.

Al contrario, un **vantaggio** dell'istogramma è la possibilità di identificare facilmente il **contrasto**: rapporto o differenza tra il valore più alto (punto più luminoso) e il valore più basso (punto più scuro) della luminosità (che corrisponde al livello di grigio per le immagini a livello di grigio).

Un'immagine viene definita:

- Valori più alti sulla destra:
 - **chiara**, caratteristica dell'immagine;

- **sovraesposta**, caratteristica di come è stata acquisita l'immagine.
- Valori più alti sulla sinistra:
 - **scura**, caratteristica dell'immagine;
 - **sottoesposta**, caratteristica di come è stata acquisita l'immagine.

3.2 Domini

L'elaborazione delle immagini può avvenire nel **dominio spaziale** o nel **dominio frequenziale** (dopo aver applicato la T.d.F. discreta 2D). Nel **dominio spaziale**, l'elaborazione delle immagine può essere espressa come:

$$g(x, y) = T[f(x, y)]$$

In cui:

- f è l'immagine di ingresso (input) da elaborare;
- g è l'immagine d'uscita (output) elaborata;
- T è un operatore su f definito in un intorno di (x, y) .

L'**operatore** definito in un intorno di (x, y) può essere di tre tipi:

- **Puntuale:** $[f(x, y)] = f(x, y)$, l'intorno coincide con il pixel stesso;
- **Locale:** $[f(x, y)]$ rappresenta una regione, per esempio quadrata, attorno al pixel di locazione (x, y) ;
- **Globale:** $[f(x, y)]$ rappresenta l'intera immagine f .

3.3 Operazioni puntuale

Si dice **operatore puntuale**, un operatore che ha preso in input il valore di un pixel e ne restituisce uno cambiato, il quale dipende esclusivamente dal valore del pixel in ingresso.

L'**obbiettivo** è quello di variare il contrasto. Infatti, eseguendo questa operazione, si evidenziano le differenze strutturali dell'oggetto rappresentato. Per farlo, basta cambiare il valore di un pixel per renderlo più scuro o più chiaro.

Un operatore puntuale può essere rappresentato tramite una **funzione** che prende in input un valore r e lo modifica in un valore $s = T(r)$ con s, r appartenenti allo stesso campo di definizione (esempio tra 0 e 255). Più in generale viene definita come:

$$T : [0, L - 1] \subset \mathbb{R} \longrightarrow [0, L - 1] \subset \mathbb{R}$$

Dato che un operatore puntuale dipende solo dal singolo valore del pixel, esso è dunque descritto da una tabella di questo tipo:

r	0	1	2	3	4	5	6	...
s	$T(0)$	$T(1)$	$T(2)$	$T(3)$	$T(4)$	$T(5)$	$T(6)$...

3.3.1 Identità

È l'operazione più semplice e non fa nulla:

$$s = r$$

3.3.2 Negativo

Rende l'immagine più scura:

$$s = L - 1 - r$$

Nel caso dei livelli di grigio:

$$s = 255 - r$$

Viene **utilizzata** quando si hanno dettagli grigi immersi in zone nere che si vogliono evidenziare.

3.3.3 Clamping

Limita l'intensità ad un range definito $[a, b]$:

$$T(r) = \begin{cases} a & \text{se } r < a \\ r & \text{se } a \leq r \leq b \\ b & \text{se } r > b \end{cases}$$

Viene **utilizzata** nel caso in cui ci siano dei pixel di rumore molto chiari o molto scuri che *mascherano* l'immagine. Quindi, si pensi per esempio ad un'immagine con dei puntini bianchi al quale si applica il *clamping* per rimuoverli.

3.3.4 Stretching/Shrinking

Stira/comprime le intensità di un range $[r_{\min}, r_{\max}]$ ad un range definito $[a, b]$:

$$s = \left[\frac{r - r_{\min}}{r_{\max} - r_{\min}} \right] [b - a] + a$$

In cui:

- r_{\min} / r_{\max} sono il più piccolo/grande livello di grigio del range che si vuole trattare;
- a, b sono il minimo e il massimo “stretchati”.

Nota bene: l'operazione è seguita da *rounging* (arrotondamento) nel caso di dominio di valori nei naturali (come in 0-255). Inoltre, lo **stretching** non risolve il problema del rumore impulsivo (puntini neri o bianchi), neanche se mascherato con il clamping.

3.3.5 Trasformazione logaritmica

La forma generale è:

$$s = c \cdot \log(1 + r) \quad r \in [0, L - 1]$$

Con c che rappresenta la **costante di normalizzazione**:

$$c = \frac{L - 1}{\log(L)}$$

La quale assicura la mappatura in $[0, L - 1]$. Inoltre, l'aggiunta di 1 permette di evitare il calcolo di quantità $\in [0, 1]$ che generano valori minori di zero ed in particolare il calcolo di $\log(0) = -\infty$.

Viene **utilizzata** quando si vuole mappare fasce strette di valori dell'immagine originale in fasce più ampie, aumentandone così il range del contrasto, rendendo inoltre l'interpretazione umana più informativa.

3.3.6 Trasformazione esponenziale

Al contrario della trasformazione logaritmica, la trasformazione esponenziale consente di aumentare il range di una fascia determinata di livelli di grigi chiari:

$$s = (e^r)^{\frac{1}{c}} - 1 \quad r \in [0, L - 1]$$

Con la **costante di normalizzazione**:

$$c = \frac{L - 1}{\log(L)}$$

3.3.7 Trasformazione di potenza

La trasformazione di potenza può essere espressa come:

$$s = cr^\gamma \quad c, \gamma > 0 \in \mathbb{R}$$

La costante c è scelta **in dipendenza da** γ in modo da normalizzare i valori di s nell'intervallo $[0, 255]$.

- $\gamma < 1$, la trasformazione ha effetti analoghi alla trasformazione logaritmica (3.3.5), cioè espansione della dinamica per bassi valori di r , mentre compressione della dinamica per alti valori di r ;
- $\gamma > 1$, la trasformazione ha effetti opposti ai valori negativi di gamma.

Nella pratica, il termine di normalizzazione c è complicato da definire analiticamente, quindi si preferiscono due versioni di s :

- **Non normalizzata:** $\tilde{s} = r^\gamma$;
- **Normalizzata** $s = cr^\gamma$

Per passare dalla versione **non normalizzata** alla **versione normalizzata** si esegue lo *stretching*:

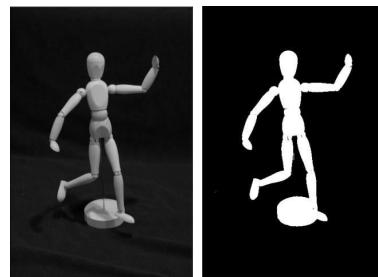
$$s = \left[\frac{\tilde{s} - \tilde{s}_{\min}}{\tilde{s}_{\max} - \tilde{s}_{\min}} - \tilde{s}_{\min} \right] [\max - \min]$$

Dove $\tilde{s}_{\min/\max}$ sono il più piccolo/grande livello di grigio e max e min sono il massimo e il minimo livello di grigio possibile (255, 0).

3.3.8 Binarizzazione

Produce un'immagine che ha solo due livelli: nero e bianco. Si **ottiene** scegliendo una soglia T , si imposta a colore nero tutti i pixel il cui valore è minore a T e si imposta a colore bianco tutti gli altri.

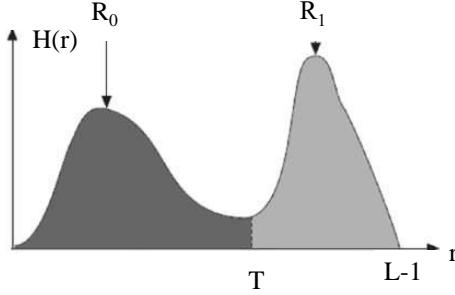
Si **utilizza** la binarizzazione per discriminare un oggetto dalla scena.



Soltanamente il suo utilizzo è prevalente nell'ambito delle immagini biomedicali e di videosorveglianza. La difficoltà maggiore di questa tecnica è il saper scegliere la soglia T più ragionevole.

3.3.9 Binarizzazione attraverso il metodo di Otsu

Questo metodo assume che ci siano due regioni da scegliere, come nella seguente figura:



Se l'immagine ha un **istogramma bimodale** la binarizzazione è efficace, altrimenti no. Le due regioni sono definite come:

$$\begin{aligned} p_{0 \rightarrow T}(r) &= \frac{H(r)}{\sum_{r=0}^T H(r)} \\ \sigma_{0 \rightarrow T}^2 &= \sum_{r=0}^T p_{0 \rightarrow T}(r) (r - \mu_{0 \rightarrow T})^2 \\ \mu_{0 \rightarrow T} &= \sum_{r=0}^T r \cdot p_{0 \rightarrow T}(r) \end{aligned}$$

La formula da minimizzare su T è la seguente:

$$\sigma_w^2(T) = W_0(T) \sigma_0^2(T) + W_1(T) \sigma_1^2(T)$$

Dove si considera la versione probabilistica dell'istogramma, ovvero la sua versione normalizzata, e si ha:

$$\begin{aligned} W_0(T) &= \sum_{r=0}^{T-1} p(r) \\ W_1(T) &= \sum_{r=T}^{L-1} p(r) \\ p(r) &= \frac{1}{M \cdot N} \sum_{r=0}^{L-1} H(r) \end{aligned}$$

Dove W_0, W_1 sono le probabilità che le due classi siano separate da T e σ_0^2 e σ_1^2 sono le varianze sui valori di istogramma assunti dalle due classi. **L'approccio "cicla" su tutti i possibili valori di T e restituisce:**

$$T_{best} = \arg_T \min (\sigma_w^2(T))$$

3.3.10 Equalizzazione

Un’immagine si dice **equalizzata** quando il contributo di ogni differente tonalità di grigio è simile. L’istogramma tende ad essere uniforme o appiattito.

L’**obbiettivo** è vedere l’istogramma come una distribuzione e di renderla il più simile a quella uniforme. Una **distribuzione uniforme** ha un’**entropia massima**⁴. Nelle immagini, ogni valore della distribuzione è un valore di grigio, per cui ogni valore di grigio appartiene all’entropia massima.

Si **utilizza** questo operatore poiché un istogramma piatto assicura a livello percettivo una risposta del cervello migliore (in termini di numero di dettagli che si riesce a riconoscere), per cui l’immagine diventa più “informativa” da osservare.

Se r_k è il k -esimo livello di grigio $k = 0, \dots, L - 1$ e $H(r_k)$ è il conteggio dato dall’istogramma dell’immagine di dimensione $M \times N$, allora si può definire:

$$p_r(r_k) = \frac{H(r_k)}{M \cdot N}$$

L’**equalizzazione dell’istogramma** si basa sulla seguente funzione T , con s_k che rappresenta il k -esimo valore di grigio in cui viene “mappato” r_k :

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) = \frac{\sum_{j=0}^k H(r_j)}{\frac{M \cdot N}{(L - 1)}}$$

Algoritmo

- I. Calcolare le L somme cumulative $\sum_{j=0}^k p_r(r_j)$ dei valori dell’istogramma visto come distribuzione con $k = 0, \dots, L - 1$;
- II. Moltiplicare i valori del passo precedente per il massimo di livelli di grigio $L - 1$;
- III. Normalizzazione dei valori calcolati al primo passo, dividendo per il numero totale di pixel $M \cdot N$ e arrotondamento;
- IV. Applicare il mapping T ottenuto.

⁴Secondo l’entropia, un sistema isolato si trasforma ed evolve nel tempo fino a raggiungere uno stato di equilibrio finale macroscopico in cui le differenze locali sono minime.

Esempio

Sia data un'immagine con $L = 8, 64 \times 64$ pixel ($M \cdot N = 4096$), con la seguente distribuzione d'intensità:

r_k	$H(r_k)$	$p_r(r_k) = \frac{H(r_k)}{M \cdot N}$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

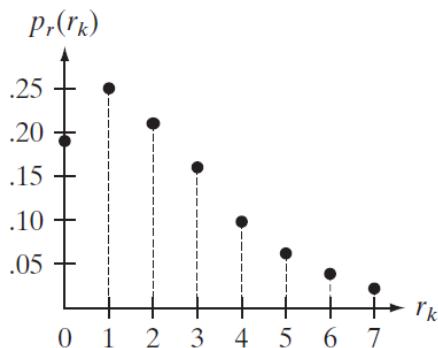


Figura 30: Rappresentazione della distribuzione di intensità.

Si applica la formula di equalizzazione:

$$\begin{aligned}
 s_0 &= T(r_0) = 7 \sum_{j=0}^0 p_r(r_j) = 7p_r(r_0) = 1.33 \\
 s_1 &= T(r_1) = 7 \sum_{j=0}^1 p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08
 \end{aligned}$$

Analogamente anche per gli altri valori si applica la formula e si trovano i seguenti valori:

$$\begin{aligned}s_2 &= 4.55 \\ s_3 &= 5.67 \\ s_4 &= 6.23 \\ s_5 &= 6.65 \\ s_6 &= 6.86 \\ s_7 &= 7.00\end{aligned}$$

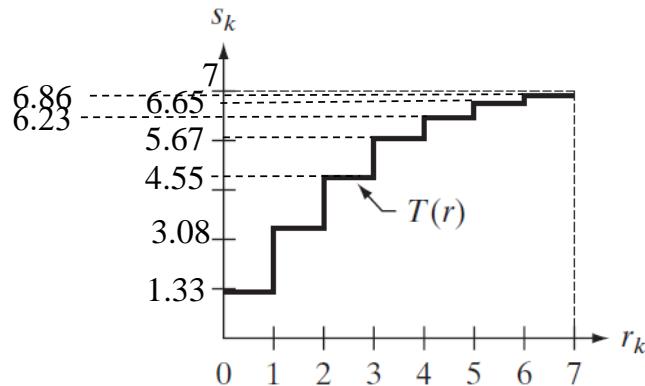


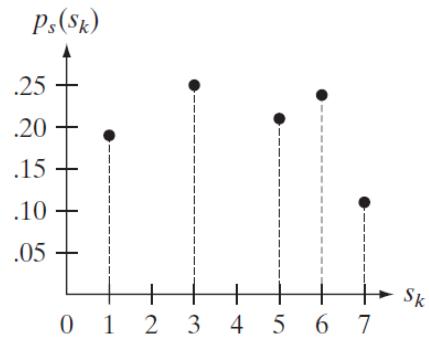
Figura 31: LUT (*Lookup Table*)

LUT (*Lookup Table*) è un termine utilizzato per descrivere una predeterminata lista di numeri che offre una “scorciatoia” per una specifica computazione. Nel contesto dei colori, una LUT trasforma i colori, ricevuti come input (camera), in un output desiderato (final footage).

L’immagine è quantizzata, quindi si effettua l’arrotondamento dei valori ottenendo l’intero più vicino:

$$\begin{aligned}s_0 &= 1.33 \longrightarrow 1 \\ s_1 &= 3.08 \longrightarrow 3 \\ s_2 &= 4.55 \longrightarrow 5 \\ s_3 &= 5.67 \longrightarrow 6 \\ s_4 &= 6.23 \longrightarrow 6 \\ s_5 &= 6.65 \longrightarrow 7 \\ s_6 &= 6.86 \longrightarrow 7 \\ s_7 &= 7.00 \longrightarrow 7\end{aligned}$$

Dopo l'arrotondamento, si ottiene una nuova immagine e il suo relativo istogramma.



3.4 Operazioni locali

Un'**operazione locale** restituisce un pixel che dipende da un limitato intorno del corrispondente punto in input. Tali operazioni vengono **utilizzati** per migliorare la qualità delle immagini o per estrarre delle informazioni dall'immagine.

Le operazioni locali sono come dei filtri spaziali dell'immagine. Il **filtraggio spaziale** è un'elaborazione T dell'immagine f dove un pixel di locazione (n, m) , di intensità $f(n, m)$, viene cambiato in $g(n, m)$ da una funzione dei pixel in un intorno di (n, m) , ossia:

$$g(n, m) = T([f(n, m)])$$

Dove le parentesi quadrate indicano che viene preso in considerazione un intorno di n, m . Ovviamente, il risultato dell'operazione, se applicato a tutti i pixel dell'immagine f , è una nuova immagine g .

Gli intorni presi maggiormente in considerazione sono di grandezza $K \times K$, con K dispari (per fare in modo di considerare uniformemente i pixel attorno al punto (n, m) di applicazione), di solito $3 \times 3, 5 \times 5, 7 \times 7$. I pixel al di fuori dell'intorno non prendono parte alla funzione.

Pseudocodice

- **Input:**

- Immagine f definita con un suo valore di pixel generico $f(n, m) \in [0 \dots L - 1] \subset \mathbb{N}$ con $(n, m) \in [1 \dots N] \times [1 \dots M] \subset \mathbb{N} \times \mathbb{N}$;
- Intorno di valori di pixel $[f(n, m)]$ ossia $f(n - u, m - v)$ definita come:

$$(u, v) \in \left[-\frac{K-1}{2} \dots \frac{K-1}{2} \right] \times \left[-\frac{K-1}{2} \dots \frac{K-1}{2} \right] \subset \mathbb{N} \times \mathbb{N} \quad K \in \{3, 5, 7, \dots\}$$

- **Output:**

- Nuova immagine $g(n, m) \in \mathbb{R}$ che attraverso operazioni puntuali può essere riportata in $g(n, m) \in [0 \dots L - 1] \subset \mathbb{N}$

- **Procedimento:**

```
for n = 1 ... N
    for m = 1 ... M
        g(n, m) = T([f(n, m)])
```

3.4.1 Filtraggi spaziali: lineari e non lineari

Le due principali categorie di operazioni locali sono lineari e non lineari:

- **Filtraggio lineare** se T è una combinazione lineare dei valori di pixel nel vicinato. Quindi, la convoluzione di un'immagine con un kernel è una somma di fattori ognuno dei quali è una moltiplicazione di un valore dell'immagine per un coefficiente del filtro:

$$g(n, m) = T([f(n, m)]) = h * f(n, m) = \sum_{u=-k}^{+k} \sum_{v=-k}^{+k} h(u, v) f(n - u, m - v)$$

$$k = \frac{K - 1}{2}$$

Un esempio di operazione lineare è la convoluzione;

- **Filtraggio non lineare** se T contiene operazioni non lineari sulle variabili indipendenti. Un esempio di operazioni non lineari sono la mediana dei pixel nell'intorno e il valore massimo dei pixel nel vicinato.

I filtraggi lineari **non** presentano un **problema ai bordi** nel momento in cui l'intorno è definito all'interno dell'immagine (cioè non cade fuori dall'immagine). Alcuni filtri lineari:

- **Cropping** è un filtro dove solo l'intorno cade all'interno dell'immagine, quindi il filtro viene applicato ad un'area ristretta e non a tutta l'immagine. Un esempio:

$$\text{Input} = \begin{bmatrix} 1 & 2 & 2 & 3 & 1 \\ 3 & 2 & 2 & 1 & 4 \\ 2 & 5 & 2 & 7 & 1 \\ 9 & 0 & 1 & 1 & 2 \\ 3 & 1 & 2 & 4 & 1 \end{bmatrix} \longrightarrow \text{Output} = \begin{bmatrix} * & * & * & * & * \\ * & 30 & 45 & 30 & * \\ * & 46 & 27 & 37 & * \\ * & 34 & 41 & 28 & * \\ * & * & * & * & * \end{bmatrix}$$

Le aree con un * non verranno calcolate;

- **Zero Padding** utilizzato per inserire degli zero che creano degli artefatti così da consentire il filtraggio. Un esempio:

$$\text{Input} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 3 & 1 & 0 \\ 0 & 3 & 2 & 2 & 1 & 4 & 0 \\ 0 & 2 & 5 & 2 & 7 & 1 & 0 \\ 0 & 9 & 0 & 1 & 1 & 2 & 0 \\ 0 & 3 & 1 & 2 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \longrightarrow \text{Output} = \begin{bmatrix} 11 & 19 & 17 & 22 & 11 \\ 25 & 30 & 45 & 30 & 31 \\ 25 & 46 & 27 & 37 & 19 \\ 35 & 34 & 41 & 28 & 29 \\ 16 & 27 & 12 & 18 & 10 \end{bmatrix}$$

Le aree in grigio non vengono calcolate.

- **Replicazione** utilizzata per creare artefatti, infatti l'immagine risultante non è realistica. Un esempio:

$$\text{Input} = \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 1 & 1 \\ 1 & 1 & 2 & 2 & 3 & 1 & 1 \\ 3 & 3 & 2 & 2 & 1 & 4 & 4 \\ 2 & 2 & 5 & 2 & 7 & 1 & 1 \\ 9 & 9 & 0 & 1 & 1 & 2 & 2 \\ 3 & 3 & 1 & 2 & 4 & 1 & 1 \\ 3 & 3 & 1 & 2 & 4 & 1 & 1 \end{bmatrix} \longrightarrow \text{Output} = \begin{bmatrix} 25 & 27 & 29 & 31 & 33 \\ 34 & 30 & 45 & 30 & 39 \\ 51 & 46 & 27 & 37 & 32 \\ 54 & 34 & 41 & 28 & 35 \\ 48 & 32 & 24 & 34 & 26 \end{bmatrix}$$

Le aree in grigio non vengono calcolate.

3.5 Rumore nelle immagini

Il **rumore nelle immagini** è un disturbo dell'immagine introdotto dal sistema di acquisizione (e.g. fotocamera) o dal mezzo di propagazione che ne degrada la qualità (e.g. Whatsapp). Il rumore è tipicamente *modellato* come **additivo e casuale**:

$$\tilde{f}(n, m) = f(n, m) + \varepsilon(n, m)$$

Dove f è l'**immagine** priva di rumore e ε è un processo aleatorio che genera delle quantità che seguono una distribuzione particolare, indipendentemente da dove il processo è collocato nell'immagine, ovvero indipendentemente da n, m .

Esistono due **tipi di rumore**: gaussiano additivo bianco (rumore generato da una distribuzione gaussiana) e impulsivo (rumore generato da una distribuzione bernoulliana).

La **quantità di rumore** può essere stimata attraverso la misura di *SNR* (*signal to noise ratio*), di cui esistono varie versioni. La più utilizzata è la **mean square, SNR_{ms}**:

$$SNR_{ms} = \frac{\sum_{n=1}^N \sum_{m=1}^M \tilde{f}(n, m)^2}{\sum_{n=1}^N \sum_{m=1}^M [\tilde{f}(n, m) - f(n, m)]^2}$$

Una forma alternativa della *SNR* può essere stimata grazie alla varianza σ_n^2 , o alla deviazione standard σ_n :

$$SNR = \frac{\sigma_s}{\sigma_n}$$

Dove σ_s è la deviazione standard del segnale e σ_n è la deviazione standard dell'immagine affetta da rumore. Per questo motivo si utilizzano immagini ad alto contrasto, poiché σ_s risulta maggiore!

3.5.1 Rumore gaussiano additivo bianco

Il **rumore gaussiano additivo bianco** è un processo stocastico, ovvero una variabile aleatoria che emette valori casuali nel tempo $\varepsilon(t)$ o nello spazio $\varepsilon(n, m)$ con le seguenti caratteristiche:

- Si somma al segnale pulito;
- Non è periodico nel tempo o nello spazio;
- I valori vengono prodotti con la seguente probabilità:

$$P(\varepsilon(n, m) = l) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(l - 0)^2}{2\sigma^2}\right)$$

In cui 0 è uguale a μ , ovvero indica la media del rumore. Inoltre, data una distribuzione gaussiana μ, σ^2 il 98% di valori $l \in [\mu - 2.5\sigma, \mu + 2.5\sigma]$.

- I valori seguono una distribuzione gaussiana di media pari a zero, ed una particolare varianza σ^2 (o deviazione standard σ) dove più è alta la varianza, più distanti da zero saranno i numeri prodotti e sommati all'immagine pulita, più rumorosa l'immagine finale.

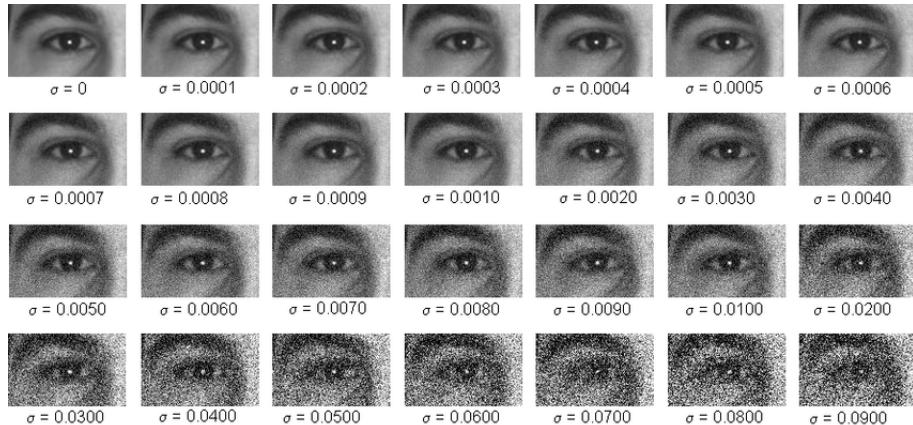


Figura 32: Esempio di rumore gaussiano a diversi valori di σ .

3.5.2 Rumore impulsivo

Il **rumore impulsivo** è causato da alterazione brusche nel segnale, viene parametrizzato da un fattore D (una percentuale) che è la densità con cui esso si localizza su pixel dell'immagine: maggiore il valore di intensità D , maggiore sarà il numero di pixel affetti.

Per esempio, il disturbo sale e pepe (*salt-and-pepper noise*) può essere utilizzato selezionando una percentuale D di pixel, ovvero $D\%$, in maniera uniforme nell'immagine e per ogni pixel si assegna un valore minimo o massimo con probabilità uniforme pari a $p = 0.5$.

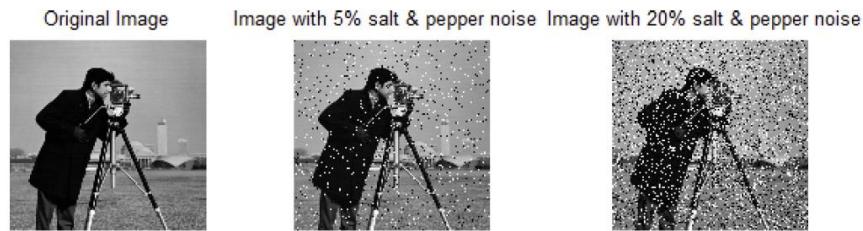


Figura 33: Esempio di applicazione dell'effetto *salt-and-pepper noise*.

3.6 Altre operazioni locali: tipologie di filtraggio

Esistono altre 3 tipologie principali di filtraggio:

- **Smoothing**, utilizzato per aumentare il *SNR* (pagina 69), ovvero per rimuovere il rumore.
Per esempio, il filtro di media, mediano, gaussiano;
- **Sharpening**, utilizzato per aumentare il grado di dettaglio delle immagini.
Per esempio, il filtro laplaciano;
- **Estrazione di caratteristiche**, utilizzato per estrarre rappresentazioni alternative alle immagini di partenza, che ne evidenzino aspetti particolari (edge, microstrutture, oggetti).
Per esempio, il filtro prewitt, sobel, canny.

3.6.1 Smoothing - Filtro media

Il **filtro media** è utilizzato per **rimuovere il rumore gaussiano**. È un filtraggio T lineare, si attua attraverso la convoluzione dell'immagine con la maschera media la quale ha le seguenti caratteristiche:

- Dimensioni $K \times K$ con K dispari;
- I suoi coefficienti sono tutti uguali e pari a $\frac{1}{K^2}$;
- Il suo funzionamento è il seguente: dato un intorno di applicazione $[(n, m)]$, esso calcola la media dei valori vii compresi $[\tilde{f}(n, m)]$, e la sostituisce al posto del valore $\tilde{f}(n, m)$:

$$g(n, m) = T([\tilde{f}(n, m)]) = E([f(n, m)])$$

Dove E è l'operatore di media, perché T essenzialmente è l'operatore di media

Si osservi che la somma dei valori del kernel è 1:

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Questo significa che il filtraggio in una locazione (x, y) è una **combinazione lineare convessa**. In altre parole, la somma dei livelli di grigio dell'immagine originale f e di quella processata g sono uguali (a meno di padding!).

Maggiore è l'ampiezza K della maschera, **più severo è l'effetto della media** sulla struttura dell'immagine.

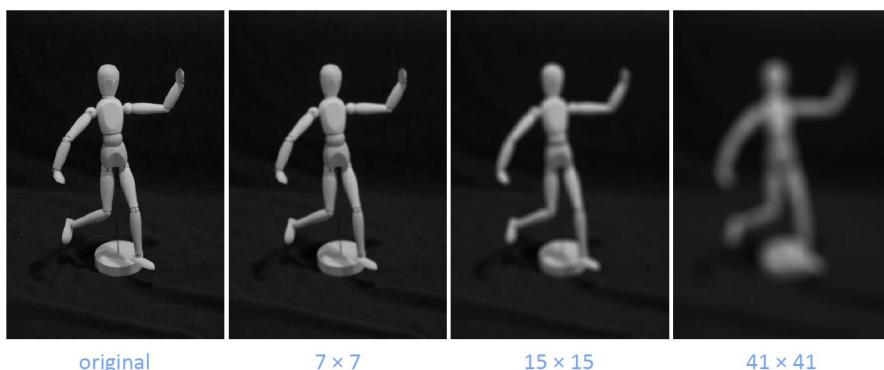


Figura 34: Esempio di filtro media all'aumentare di K , ovvero della grandezza.

3.6.2 Smoothing - Filtro mediano

Il **filtro mediano** è utilizzato per **rimuovere il rumore impulsivo**. È un filtraggio T non lineare, che si realizza attraverso un algoritmo. Data la matrice dell'immagine:

$$\begin{bmatrix} 240 & 245 & 0 \\ 247 & 0 & 244 \\ 251 & 246 & 250 \end{bmatrix}$$

1. Si calcola la media di tutti i valori della matrice:

$$\frac{240 + 245 + 0 + 247 + 0 + 244 + 251 + 246 + 250}{9} \approx 191$$

2. Si inseriscono in un vettore riga i valori in ordine crescente:

$$[0 \ 0 \ 240 \ 244 \ 245 \ 246 \ 247 \ 250 \ 251]$$

3. Si calcola la mediana del vettore:

$$9 \div 2 = 4.5 \longrightarrow [0 \ 0 \ 240 \ 244 \ \underline{245} \ 246 \ 247 \ 250 \ 251]$$



Figura 35: Esempio di applicazione di filtro mediano.

3.6.3 Smoothing - Filtro gaussiano

Il **filtro gaussiano** è quello di rendere più “lisica (*smooth*)” l’immagine, in modo simile al filtraggio di media, e come parametro l’operatore prende il valore σ che rappresenta la **forza** (maggiore è il valore, più è forte lo *smoothing*). La **differenza** sostanziale tra il filtraggio di media e il filtraggio gaussiano è che quest’ultimo è una media pesata, dove i pesi più vicini al centro della maschera hanno valori più alti. Così facendo si ha:

- **Vantaggio**

- Il filtraggio gaussiano effettua uno *smoothing* più lieve, **preservando i contorni meglio** di quanto faccia il filtraggio media. Quindi, la struttura viene preservata meglio.

- **Svantaggio**

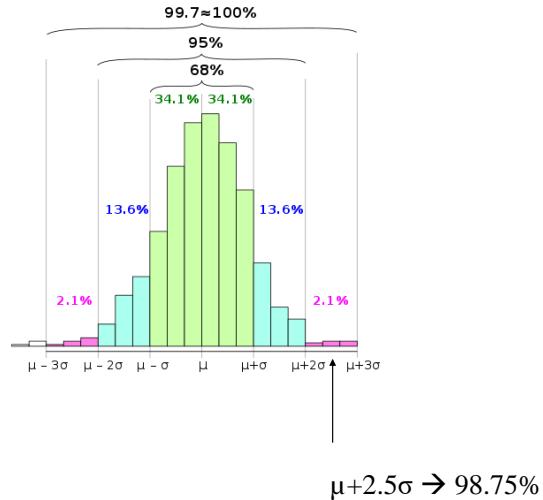
- Il rumore viene rimosso in maniera inferiore e questo provoca l'**impossibilità di applicare la formula di annullamento del rumore** visto per il filtro media.

Questo filtro può essere **implementato in maniera efficiente** in quanto la maschera è separabile, ovvero è possibile eseguirlo facendo un filtraggio prima su tutte le N righe dell’immagine come se fossero funzioni $1D$, e poi su tutte le M colonne:

$$\begin{aligned} I_G(i, j) &= \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} G(h, k) I(i + h, j + k) \\ &= \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} \exp\left(-\frac{h^2 + k^2}{2\sigma^2}\right) I(i + h, j + k) \\ &= \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \exp\left(-\frac{h^2}{2\sigma^2}\right) \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} \exp\left(-\frac{k^2}{2\sigma^2}\right) I(i + h, j + k) \end{aligned}$$

Dove le variabili i, j, h, k, m sono indici per comprendere la **separabilità**. Quest’ultima consente di progettare manualmente un filtro gaussiano come segue nella prossima pagina.

Si definiscono i parametri σ e W . Quindi, si fissi σ e si trovi la dimensione della maschera W sapendo che W deve essere tale da contenere un'elevata percentuale di probabilità (uguale all'area della densità gaussiana, come in figura). In particolare, la statistica dice che con $W = 5\sigma$ si copre il 98.75% dell'area della densità gaussiana. In altre parole, se si vuole $\sigma = 1$ allora $W = 5 \cdot 1 = 5$; se si vuole $\sigma = 0.6$ allora $W = 5 \cdot 0.6 = 3$.



3.6.4 Domanda da esame

Domanda

Il livello di noise gaussiano presente in un'immagine, se esso noto (e.g. $\sigma_1 = 0.01$) può guidare la scelta del parametro σ_2 della maschera di filtro gaussiano?

Risposta

No, perché il rumore gaussiano lavora su tutti i valori di grigio, mentre il filtro gaussiano sulle coordinate e non c'è correlazione. Invece, il filtro media è quello ideale.

3.6.5 Filtraggi di sharpening

I **filtraggi di sharpening** servono per evidenziare i dettagli o come post processing dopo filtraggi di *smoothing* (questo perché i filtraggi di *smoothing* eliminano i dettagli). Per lo stesso motivo, i filtraggi di sharpening possono incrementare il rumore (un'immagine di rumore è un'immagine ad alta frequenza). Esistono due categorie: ***basic highpass spatial filtering*** e ***high boost filtering***.

I filtri di sharpening sono detti anche **filtri di derivata**, poiché calcolano numericamente nell'intorno in cui sono definiti la derivata locale (prima o seconda) dell'immagine.

Rispetto a x

Derivata asimmetrica

$$\begin{aligned} I_x(x, y) &= \frac{\partial I}{\partial x}(x, y) = \lim_{h \rightarrow 0} \frac{I(x+h, y) - I(x, y)}{h} \\ I_x[x, y] &= \frac{\partial I}{\partial x}[x, y] = I[x+1, y] - I[x, y] \end{aligned}$$

Derivata simmetrica

$$\begin{aligned} I_x(x, y) &= \frac{\partial I}{\partial x}(x, y) = \lim_{h \rightarrow 0} \frac{I(x+h, y) - I(x-h, y)}{2h} \\ I_x[x, y] &= \frac{\partial I}{\partial x}[x, y] = \frac{1}{2}(I[x+1, y] - I[x-1, y]) \end{aligned}$$

Filtro differenziale asimmetrico

$$\partial_x = [-1 \quad 1]$$

Filtro differenziale simmetrico

$$\partial_x = \frac{1}{2} [-1 \quad 0 \quad 1]$$

Rispetto a y

Derivata asimmetrica

$$I_y(x, y) = \frac{\partial I}{\partial y}(x, y) = \lim_{h \rightarrow 0} \frac{I(x, y + h) - I(x, y)}{h}$$

$$I_y[x, y] = \frac{\partial I}{\partial y}[x, y] = I[x, y + 1] - I[x, y]$$

Derivata simmetrica

$$I_y(x, y) = \frac{\partial I}{\partial y}(x, y) = \lim_{h \rightarrow 0} \frac{I(x, y + h) - I(x, y - h)}{2h}$$

$$I_y[x, y] = \frac{\partial I}{\partial y}[x, y] = \frac{1}{2} (I[x, y + 1] - I[x, y - 1])$$

Filtro differenziale asimmetrico

$$\partial_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Filtro differenziale simmetrico

$$\partial_y = \frac{1}{2} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Proprietà della derivata prima:

- Nulla in regioni di intensità costante;
- Non nulla in presenza di variazioni di intensità.

Proprietà della derivata seconda:

- Nulla in regioni di intensità costante;
- Nulla in presenza di variazioni costanti di intensità (rampe);
- Non nulla in presenza di variazioni non costanti (all'inizio e alla fine di rampe).

3.6.6 Sharpening - Basic Highpass Spatial Filtering

Il filtraggio di sharpening chiamato **basic highpass spatial filtering** è un **filtraggio lineare** con il laplaciano, con la maschera H caratterizzata da coefficienti di un segno (e.g. positivo) vicino al centro e di segno opposto (e.g. negativo) nella periferia esterna.

Una tipica maschera di filtraggio, chiamata laplaciana:

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Altre maschere laplaciane:

$$\begin{array}{ll} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \\ \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} & \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \end{array}$$

Alcune caratteristiche:

- La **somma dei coefficienti è zero**. Questo indica che quando il filtro passa su regioni con livelli di grigio quasi stabili, l'output della maschera è zero o molto piccolo;
- L'**uscita è alta** quando il valore centrale differisce dai valori periferici;
- L'**immagine di output non assomiglierà** a quella originale;
- L'**immagine di output mostra tutti i dettagli**;
- Sono inclusi alcuni **ridimensionamenti e/o clipping**, necessari per compensare eventuali livelli di grigio negativi dopo il filtraggio.

Il filtraggio lineare utilizza il *basic highpass spatial filtering* per **creare un'immagine realistica**, simile a quella di partenza, con gli edge amplificati:

$$g(n, m) = T([f(n, m)]) = f(n, m) + c \cdot h * f(n, m)$$

In cui h indica la **maschera laplaciana**, c è una **costante** pari a uno nel caso in cui il pixel centrale della maschera laplaciana sia positivo, altrimenti -1 .

3.6.7 Sharpening - Filtro Laplaciano

La funzione **laplaciana** prende in ingresso un parametro α il cui significato è legato all'importanza che si vuole dare agli edge verticali e orizzontali ($\alpha = 0$), diagonali ($\alpha = 1$), tutti gli edge ($\alpha = 0.5$), attraverso questa formula:

$$h = \frac{1}{\alpha + 1} \begin{bmatrix} -\alpha & \alpha - 1 & -\alpha \\ \alpha - 1 & \alpha + 5 & \alpha - 1 \\ -\alpha & \alpha - 1 & -\alpha \end{bmatrix}$$

3.6.8 Sharpening - High Boost Filtering

L'immagine filtrata dallo *sharpening* si ottiene sottraendo l'immagine filtrata con *smoothing* dall'immagine originale:

Immagine filtrata dallo sharpening = Originale – Im. filtrata con smoothing

Se la costante A rappresenta un **fattore di amplificazione degli edge**, allora il filtro **high boost filtering** è definito come:

$$\text{High-boost} = A \cdot \text{Originale} + \text{Im. filt. dallo sharpening}$$

A differenza degli altri filtri, questo dà maggiore libertà al progettista. Infatti, il blur può avvenire attraverso una maschera di supporto arbitrariamente grande.

4 Elaborazione di immagini - Rinforzo del dominio delle frequenze

4.1 Ripasso formule utili

Si ricorda la trasformata di Fourier discreta a una dimensione (1D) che si rappresenta con la seguente equazione:

$$\tilde{F}\left(\frac{m}{M} \frac{1}{\Delta T}\right) = F\left(\frac{m}{M} \frac{1}{\Delta T}\right) = F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi \frac{m}{M} n}$$

Mentre la trasformata di Fourier discreta inversa (o antitrasformata), corrisponde a:

$$\tilde{f}(n\Delta T) = f(n\Delta T) = f_n = \frac{1}{M} \sum_{m=0}^{M-1} F_m e^{j2\pi \frac{m}{M} n}$$

È possibile fare alcune osservazioni riguardo le trasformate di fourier in una e due dimensioni. Partendo dalla definizione:

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi \frac{m}{M} n}$$

Si intende la trasformata di Fourier discreta come la moltiplicazione del segnale per delle funzioni sinusoidali 1D. Più il segnale “assomiglia” alla sinusoide di frequenza specifica con cui si moltiplica, più il valore di ampiezza della trasformata di Fourier discreta è alto per quella frequenza.

Si esegue un cambio di variabile, quindi μ diventa la **variabile delle frequenze** e x quella dello **spazio**. Entrambe rappresentano una quantità campionata e 1D è:

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi \frac{m}{M} n} \longrightarrow F_u = F(u) = \sum_{x=0}^{M-1} f(x) e^{-j2\pi \frac{u}{M} x}$$

Si passa alla seconda dimensione aggiungendo y e v per l'**analisi verticale**:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi \left(\frac{u}{M} x + \frac{v}{N} y \right)}$$

In cui $F(u, v)$ rappresenta la **risposta** di I all'immagine (complessa) di base $(u, v) e^{\dots}$, invece l'esponenziale rappresenta l'**immagine** (complessa) base che è funzione di (u, v) .

Si elencano un paio di **proprietà della trasformata di Fourier discreta 2D**, utili per questo capitolo:

- **Traslazione:**

$$\text{Segnale: } g(x, y) = f(x - x_0, y - y_0)$$

$$\text{T.d.F.: } G(u, v) = F(u, v) e^{-j2\pi(\frac{u}{M}x_0 + \frac{v}{N}y_0)}$$

$$\text{Segnale: } g(x, y) = f(x, y) e^{j2\pi(\frac{u}{M}x_0 + \frac{v}{N}y_0)}$$

$$\text{T.d.F.: } G(u, v) = f(u - u_0, v - v_0)$$

- **Rotazione:** la trasformata di Fourier di un'immagine a cui è stata applicata una rotazione θ , porterà ad un'immagine di trasformata ruotata di angolo θ :

$$\mathcal{F}(f_\theta)(u, v) = \mathcal{F}(f)_\theta(u, v)$$

Si richiama anche il **teorema di convoluzione**, ovvero la convoluzione nel tempo tra due segnali corrisponde alla moltiplicazione nel dominio delle frequenze. Ovviamente per cambiare dominio è necessario applicare Fourier:

$$\mathcal{F}(f * h) = \mathcal{F}(f) \cdot \mathcal{F}(h)$$

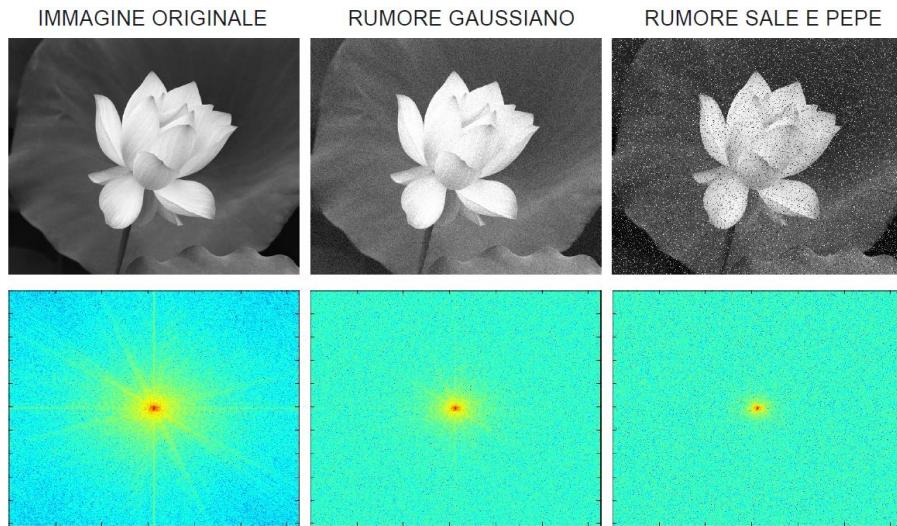
$$\mathcal{F}(f \cdot h) = \mathcal{F}(f) * \mathcal{F}(h)$$

Questo teorema è la base del filtraggio in frequenza.

4.2 Rumore nel dominio delle frequenze

Il **rumore** porta ad un aumento delle alte frequenze. Infatti, eseguendo una riduzione delle alte frequenze, si riduce sia il rumore (aspetto *positivo*) sia il livello di dettaglio (aspetto *negativo*).

Qui di seguito, vengono lasciate tre immagini per eseguire un confronto. L'immagine originale presenta certe frequenze, ma applicando il rumore gaussiano, o sale e pepe (*salt-and-pepper noise*), viene ridotto sensibilmente sia il rumore che il livello di dettaglio.



4.3 Panoramica su filtri passa alto (*high-pass*) e passa basso (*low-pass*)

I filtri passo alto (*high-pass*) e passa basso (*low-pass*) sono identici a quelli visti durante il corso di Sistemi. L'obbiettivo di questo paragrafo è introdurre qualche definizione e dare una panoramica generale dei due filtri approfonditi nei prossimi paragrafi.

In un immagine, le **frequenze basse** rappresentano informazioni con variazioni di intensità “lente”. Per esempio, le gradazioni di colore su un muro illuminato, o la nuvolosità nel cielo.

Al contrario, le informazioni ad **alte frequenze** indicano informazioni con variazioni di intensità “repentine”. Per esempio, spigoli, angoli e rumore.

Per lavorare sulle frequenze esistono due filtri importanti:

- **Filtro passa basso** (*low-pass filter*) rimuove dall'immagine le informazioni ad alte frequenze e mantiene quelle a basse frequenze.
- **Filtro passa alto** (*high-pass filter*) rimuove dall'immagine le informazioni a basse frequenze e mantiene quelle ad alte frequenze.

Dato che ogni filtro è il contrario dell'altro, da ciascuno è possibile derivare il suo opposto:

$$H_{PA} = 1 - H_{PB}$$

In cui PA indica passa alto e PB passa basso.

4.4 Filtri passa basso (*low-pass*)

4.4.1 Filtri passa basso ideale

Il **filtro passa basso ideale** viene usato per ottenere: lo sfocamento e lo *smoothing*. Solitamente, per ottenere questi risultati si attenuano le alte frequenze, ma facendo così si ottiene anche una riduzione inevitabile del rumore.

Matematicamente parlando, un filtro passa basso ideale è una funzione di trasferimento (uguale alla sua trasformata nel dominio delle frequenze) di una box.

Viene detta **ideale** perché come si vede in figura, una transizione così netta in corrispondenza alla *frequenza di taglio* non è analogicamente realizzabile. In parole poche, è ideale poiché nell'elettronica non può fisicamente avvenire un calo di energia così repentino.

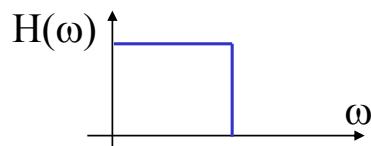


Figura 36: Funzione di trasferimento di un filtro passa basso ideale.

Dato che i filtri studiati sono solamente digitali, teoricamente potremmo tralasciare questo taglio repentino. Purtroppo, esso provoca un effetto visivo indesiderato chiamato **ringing**.

Il **ringing** (o effetto di Gibbs) è dovuto al fatto di eseguire un filtraggio passa basso ideale (in frequenza) equivalente ad eseguire una convoluzione con l'operatore sinc (nello spazio). Ne consegue che la risposta all'impulso del passa basso ideale è ancora un sinc e l'**immagine visivamente risulta increspata vicino ai bordi taglienti**⁵.

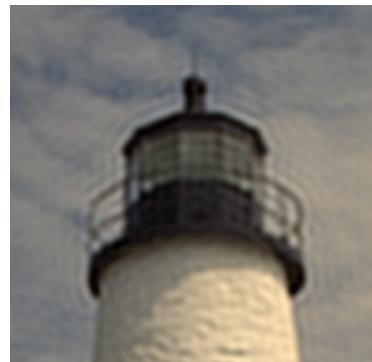
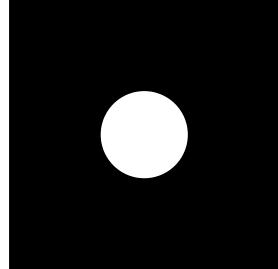


Figura 37: Effetto ringing su un'immagine.

⁵Fonte: *Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University*

Segnale H :



Le formule da applicare per il filtro basso ideale sono:

$$H(u, v) = \begin{cases} 1 & \text{se } D(u, v) \leq D_0 \\ 0 & \text{se } D(u, v) > D_0 \end{cases}$$

In cui D_0 è uguale a μ_{thresh} che indica la soglia e $D(u, v)$ è il raggio del cerchio:

$$D(u, v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$$

Da notare che solo le frequenze nel cerchio di raggio D_0 vengono mantenute.

4.4.2 Filtri passa basso di Butterworth

Il **filtro passa basso di Butterworth** è un filtro con attenuazione dolce in prossimità della frequenza di taglio. La **proprietà caratterizzante** di questo

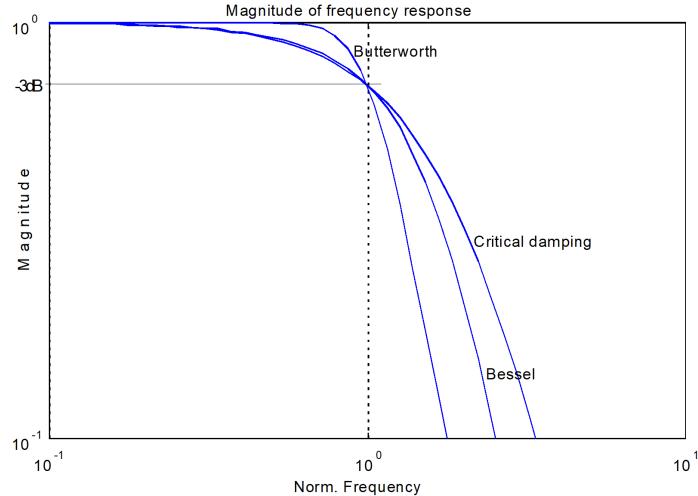


Figura 38: Filtro passa basso di Butterworth.

filtro è la **risposta molto ripida nella banda passante**.

L'ordine è n e la frequenza di taglio $D_0 = \mu_{\text{thresh}}$:

$$H(u, v) = \frac{1}{1 + \left(\frac{D(u, v)}{D_0}\right)^{2n}}$$

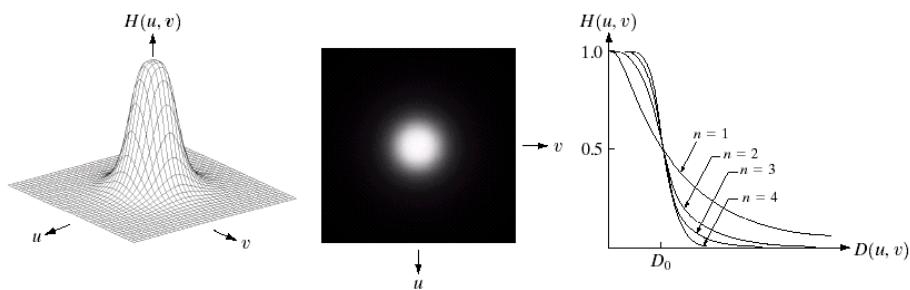


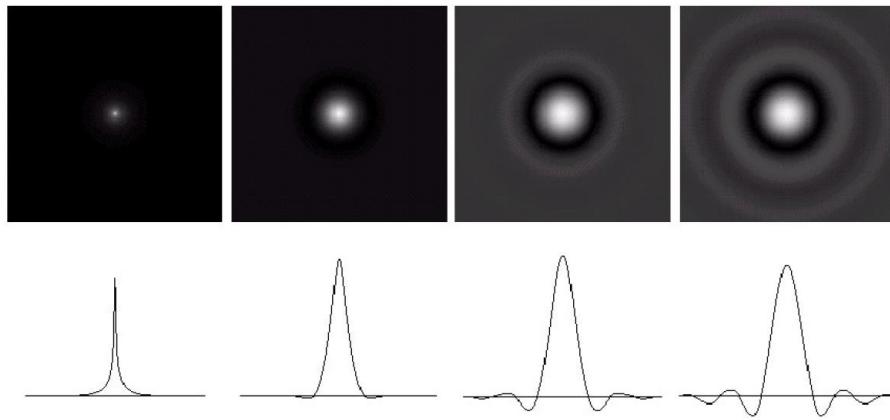
Figura 39: Applicazione dell'ordine e della frequenza di taglio.

Le formule da applicare:

$$H(u, v) = \frac{1}{1 + \left[\frac{D(u, v)}{D_0} \right]^{2n}}$$

$$D(u, v) = \sqrt{\left(u - \frac{M}{2} \right)^2 + \left(v - \frac{N}{2} \right)^2}$$

È interessante notare che facendo tendere n all'infinito, si ottiene il filtro passa basso ideale:



4.4.3 Filtri passa basso Gaussiano

La trasformata di Fourier di una funzione Gaussiana è anch'essa Gaussiana:

$$F(u) = Ae^{-\frac{u}{2\sigma^2}}$$

$$f(t) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2t^2}$$

Oppure allo stesso identico modo:

$$f(t) = Ae^{-\frac{t}{2\sigma^2}}$$

$$F(u) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2u^2}$$

Le equazioni di questa operazione sono:

$$\begin{aligned} H(u, v) &= e^{-\frac{D^2(u, v)}{2D_0^2}} \\ D(u, v) &= \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2} \end{aligned}$$

In cui la costante D_0 può essere sostituita con σ , ovvero l'effettiva deviazione standard della distribuzione Gaussiana. In questo caso, la frequenza di taglio D_0 corrisponde alla deviazione standard σ . In altre parole, quando $D(u, v) = \sigma$, allora l'intensità di taglio è 0.607 e il filtraggio crea un attenuamento di quella frequenza pari al 60.7%.

Un'**osservazione** interessante è la seguente. Un filtro gaussiano con una certa scala nel dominio delle frequenze, corrisponde ad un filtro gaussiano con scala inversa nel dominio dello spazio.

4.4.4 Sintesi

Qui di seguito si elencano i filtri passa basso più importanti:

- **Ideale:** è una brusca transizione in corrispondenza della frequenza di *cut-off*. Questo causa un fenomeno di Gibbs o di *ringing*;
- **Gaussiano:** transizione di *cut-off* dolce. Il parametro σ determina la frequenza di *cut-off*;
- **Butterworth:** ha una rapidità variabile e transizione *smooth*. La ripidità viene modellata dall'ordine del filtro. La frequenza di *cut-off* viene selezionata indipendentemente dall'ordine del filtro. Si può avere *ringing* per ordini elevati.

4.5 Filtri passa alto (*high-pass*)

Un **filtro passa alto** sopprime (blocca) le basse frequenze e lascia passare le alte frequenze. La costruzione di un filtro passa alto può essere eseguita come:

$$H_{PA} = 1 - H_{PB}$$

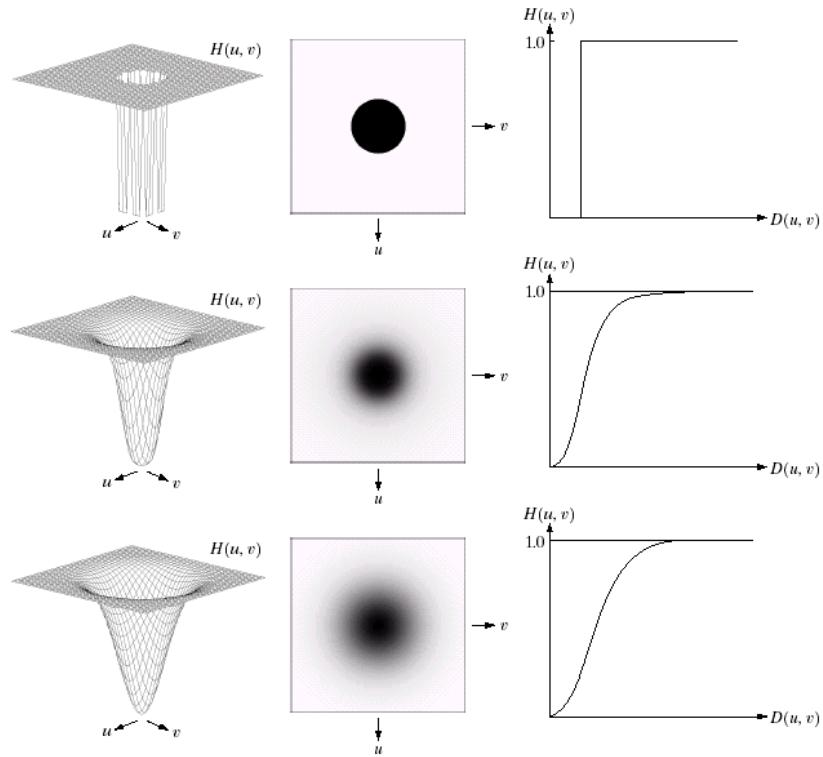
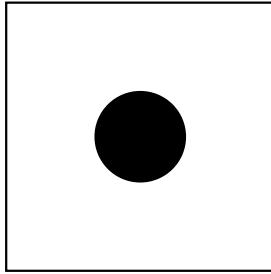


Figura 40: Filtri passa alto, dall'alto: ideale, di Butterworth, Gaussiano.

4.5.1 Filtri passa alto ideale

Dato il segnale H :



Le formule da applicare per il **filtro passa alto ideale** sono:

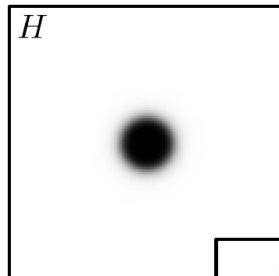
$$H(u, v) = \begin{cases} 0 & \text{se } D(u, v) \leq D_0 \\ 1 & \text{se } D(u, v) > D_0 \end{cases}$$

$$D(u, v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$$

Solamente le frequenze fuori dal cerchio di raggio D_0 vengono mantenute.

4.5.2 Filtri passa alto di Butterworth

Dato il segnale H :



Le formule da applicare per il **filtro passa alto di Butterworth** sono:

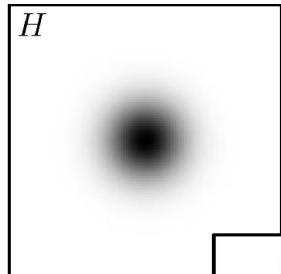
$$H(u, v) = \frac{1}{1 + \left[\frac{D(u, v)}{D_0}\right]^{-2n}}$$

$$D(u, v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$$

Per n tendente ad infinito si ha il filtro passa alto ideale.

4.5.3 Filtri passa alto Gaussiano

Dato il segnale H :



Le formule da applicare per il **filtro passa alto Gaussiano** sono:

$$H(u, v) = 1 - e^{-\frac{D^2(u, v)}{2D_0^2}}$$
$$D(u, v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$$

4.6 Filtri per enfatizzare le alte frequenze

Esiste anche il **filtro omeomorfo** che attenua le basse frequenze e aumenta quelle delle alte:

$$H(u, v) = (\gamma_H - \gamma_L) \left[1 - e^{-\frac{cD^2[u, v]}{D_0^2}} \right] + \gamma_L$$

Invece, usando il **filtro passa alto**, è possibile enfatizza aumentando la variabile k :

$$H(u, v) = (1 + k \cdot H_{PA}(u, v)) \cdot H$$

Dove k è il contributo delle alte frequenze, H lo spettro dell'immagine e H_{PA} il filtro passa alto

4.7 Filtri passa banda ideale 1D e ferma banda ideale

I seguenti filtri operano su una banda di frequenze, al contrario i precedenti filtri lavoravano su alte o basse frequenze.

Analogamente ai filtri passa basso e alto, anche qui è possibile derivare uno dei due filtri avendo solamente l'altro:

$$H_{PBn} = 1 - H_{FBn}$$

Dove PB indica passa banda e FB ferma banda.

Un **filtro passa banda ideale 1D** sopprime tutte le frequenze al di fuori di un intervallo di frequenze specificato. Più formalmente, esso è rappresentato con la seguente funzione:

$$G(\mu) = \begin{cases} 1 & \text{se } u_1 < |\mu| < u_2 \\ 0 & \text{altrimenti} \end{cases}$$

Le variabili u rappresentano l'intervallo (come si vede in figura) e al di fuori di esso il valore è zero.

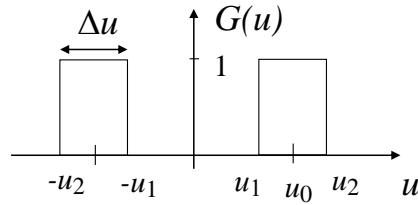


Figura 41: Rappresentazione grafica del filtro passa banda ideale 1D nel **dominio delle frequenze**.

Dalla rappresentazione grafica si ricava anche la relativa **funzione di trasferimento nel dominio delle frequenze**, ovvero la somma di due box traslate nel tempo ottenibili eseguendo la convoluzione di una box per due impulsi unitari (Dirac):

$$G(u) = \Pi\left(\frac{u}{\Delta u}\right) * [\delta(u - u_0) + \delta(u + u_0)]$$

Dove i termini u_0 e Δu indicano:

- u_0 indica dov'è centrata la box ed è ottenibile in questo modo:

$$u_0 = \frac{(u_1 + u_2)}{2}$$

- Δu indica la larghezza della box ed è ottenibile in questo modo:

$$\Delta u = u_2 - u_1$$

Invece, la **funzione di trasferimento nel dominio del tempo** è la seguente:

$$g(t) = 2\Delta u \cdot \frac{\sin(\pi t \Delta u)}{\pi t \Delta u} \cdot \cos(2\pi u_0 t)$$

Dato che $\Delta u < u_0$, allora la funzione $g(t)$ è un coseno modulato dal sinc. Pensandoci bene, la box viene rappresentata dalla funzione sinc⁶ e i due impulsi possono essere rappresentati, ricordando il corso di sistemi, dalla funzione coseno. Se u_0 è costante e Δu_0 tende a zero, allora si ha un coseno.

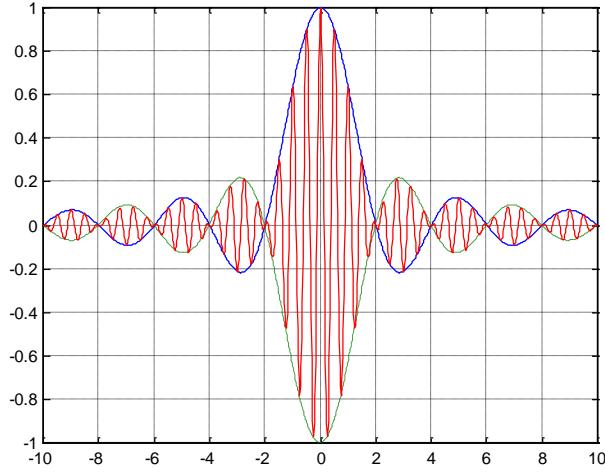


Figura 42: Rappresentazione grafica del filtro passa banda ideale 1D nel **dominio del tempo**.

⁶Si ricorda che la funzione sinc è così formata: $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$

Un **filtro ferma banda ideale** sopprime tutte le frequenze nell'intervallo specificato. Più formalmente, esso è rappresentato con la seguente funzione:

$$G(\mu) = \begin{cases} 0 & \text{se } u_1 < |\mu| < u_2 \\ 1 & \text{altrimenti} \end{cases}$$

Le variabili μ rappresentano l'intervallo (come si vede in figura) e al di fuori di esso il valore è uno. Quindi, al contrario del filtro passa banda ideale, il ferma banda non consente al segnale di esistere nelle frequenze specificate.

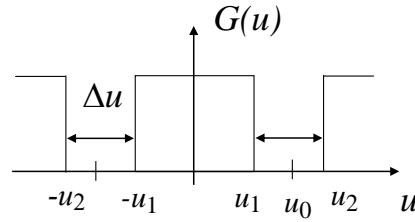


Figura 43: Rappresentazione grafica del filtro ferma banda ideale nel **dominio delle frequenze**.

Dalla rappresentazione grafica si ricava anche la relativa **funzione di trasferimento nel dominio delle frequenze**, ovvero la rimozione (segno meno) di due box traslate nel tempo ottenibili eseguendo la convoluzione di una box per due impulsi unitari (Dirac):

$$G(u) = 1 - \Pi\left(\frac{u}{\Delta u}\right) * [\delta(u - u_0) + \delta(u + u_0)]$$

I valori $u_0, \Delta u$ hanno lo stesso significato del filtro passa banda ideale. Invece, la **funzione di trasferimento nel dominio del tempo** è la seguente:

$$g(t) = \delta(t) - 2\Delta u \cdot \frac{\sin(\pi t \Delta u)}{\pi t \Delta u} \cdot \cos(2\pi u_0 t)$$

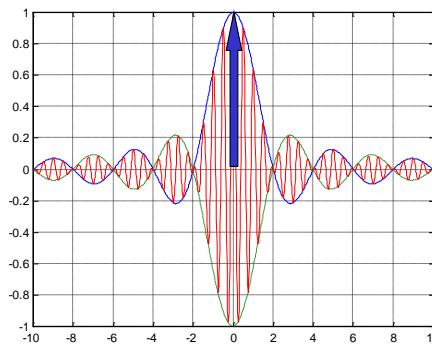


Figura 44: Rappresentazione grafica del filtro ferma banda ideale nel **dominio del tempo**.

5 Elaborazione di immagini - Estrazione di edge robusta

5.1 Estrazione di contorni (*edge*)

Si definisce come **feature** una proprietà dell'immagine. Quest'ultima può essere:

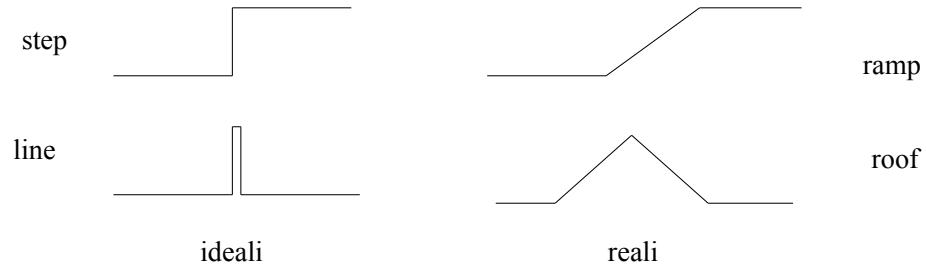
- **Globale**, ovvero dell'intera immagine;
- **Locale**, ovvero di una parte dell'immagine con speciali proprietà.

Con il termine **edge** si indica una forte variazione locale dei livelli di grigio, tipicamente associata ad un contorno o ad una separazione tra regioni.

Il fenomeno dell'*edge* provoca una serie di complicazioni. Si definisce il **problema dell'estrazione**: data un'immagine, la **rilevazione** (*detection*) e la **localizzazione** degli edge derivati dalla scena e non dal rumore.

La soluzione al problema non è affatto banale perché: il rumore può mascherare gli *edge* e possono esistere falsi *edge* positivi dovuti a ombre e variazioni di luminosità.

Esistono varie **tipologie di livello di grigio**:



Si definisce un **punto di edge**, un punto dell'immagine con coordinate (i, j) in cui è presente una variazione significativa locale dell'intensità.

Si definisce un **contorno**, una lista di punti di *edge* o un'altra struttura dati che modella i punti di *edge*, ordinandoli dal primo all'ultimo. In altre parole, è l'*edge* vero e proprio.

Ancora una volta, nasce una complicazione: il **problema della localizzazione**. Infatti, nonostante la posizione di un punto di *edge* si identifica tramite le coordinate (i, j) , le costanti i, j potrebbero non essere indici di pixel.

Inoltre, la localizzazione può richiedere anche l'**orientazione** di un punto d'*edge*. Questo implica l'introduzione del **gradiente**.

Per concludere, si **classificano gli edge**:

- **Corretti:** *edge* della scena;
- **Falsi:** non esistono nella scena (*false positives*);
- **Mancanti:** presenti nella scena, ma non rilevanti (*false negatives*).

5.2 Criteri per *edge detector* ottimi

Esistono 3 criteri per ottenere un *edge detector* ottimo:

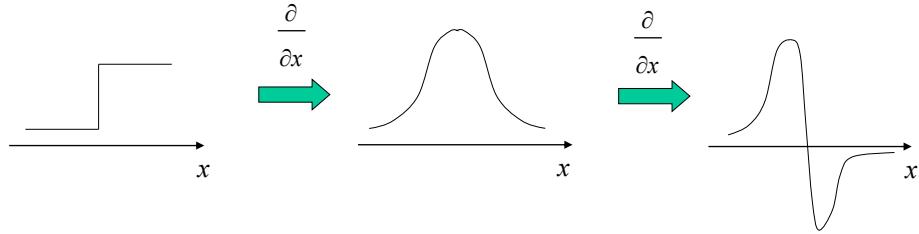
- **Buon rilevamento (*detection*)**. Deve minimizzare la probabilità di presenza di falsi positivi e la probabilità di presenza di falsi negativi. La massimizzazione del SNR (*Signal to Noise Ratio*, rapporto segnale-rumore; è una grandezza numerica che mette in relazione la potenza del segnale utile rispetto a quella del rumore⁷⁾) definito come il rapporto tra la risposta quadratica media (RMS) del filtro agli *step edge* ideali e il rumore;
- **Buona localizzazione**. Gli *edge* rilevati devono essere più vicino possibile agli *edge* reali;
- **Vincolo di singola risposta**. Per ogni punto di *edge* reale, il filtro deve rilevare un solo punto di *edge*. Ossia, minimizzare il numero di massimi locali creati dal rumore intorno all'*edge*.

I criteri di buon rilevamento e di buona localizzazione, identificano due modi di valutazione per l'estrazione di *edge* indipendentemente dalla lunghezza dell'*edge* e dalla potenza del rumore.

⁷Fonte: [WikipediaIT](#)

5.3 Il gradiente

Il **gradiente** è una misura della variazione in una funzione. Nel caso monodimensionale, corrisponde al picco della derivata prima:



Assumendo una funzione $f(x, y)$, allora:

$$\mathbf{G}[f(x, y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Il modulo è pari alla lunghezza del vettore G localizzato su x e y :

$$\|\mathbf{G}\| = G = \sqrt{G_x^2 + G_y^2}$$

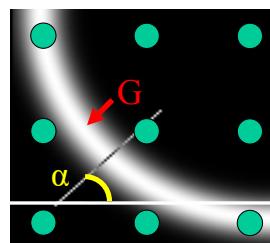
Approssimazioni del modulo:

$$\begin{aligned} G &\cong |G_x| + |G_y| \\ G &\cong \max(|G_x|, |G_y|) \end{aligned}$$

L'orientazione, misurata rispetto all'asse x , è data da:

$$\alpha(x, y) = \arctan \frac{G_y}{G_x}$$

In pratica, il vettore \mathbf{G} punta nella direzione di massimo scarto della funzione f :



5.4 Estrazione dell'edge

5.4.1 Il gradiente discreto

Per differenziare un'immagine digitale $F[x, y]$ è possibile seguire due strade:

1. Ricostruire un'immagine continua, quindi prendere il gradiente analitico;
2. Prendere la derivata discreta, tramite “differenze finite”, come visto nei capitoli precedenti.

Il **gradiente discreto** è approssimativamente la differenza tra pixel adiacenti:

$$\frac{\partial f}{\partial x}[x, y] \approx F[x+1, y] - F[x, y]$$

E le approssimazioni numeriche, invece, sono:

$$\begin{aligned} f_x &\approx f[i, j+1] - f[i, j] \longrightarrow [-1 \quad 1] \longrightarrow S_x = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \\ f_y &\approx f[i, j] - f[i+1, j] \longrightarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix} \longrightarrow S_y = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \end{aligned}$$

5.4.2 Operatore di Roberts

L'**operatore di Roberts** è un'approssimazione con differenze incrociate:

$$|d[f(i, j)]| = |f(i, j) - f(i+1, j+1)| + |f(i, j+1) - f(i+1, j)|$$

La convoluzione dell'immagine (guardare le due matrici S_x e S_y nel paragrafo 5.4.1) con 2 maschere del tipo:

$$S_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad S_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

5.4.3 Operatore di Prewitt

Data l'approssimazione con vicinato 3×3 :

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ a_7 & (i, j) & a_3 \\ a_6 & a_5 & a_4 \end{bmatrix}$$

L'**operatore di Prewitt** è un'approssimazione del tipo:

$$|r[f(i, j)]| = \underbrace{|(a_2 + a_3 + a_4) - (a_0 + a_7 + a_6)|}_{S_x} + \underbrace{|(a_0 + a_1 + a_2) - (a_6 + a_5 + a_4)|}_{S_y}$$

Il risultato:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

5.4.4 Operatore di Sobel

L'**operatore di Sobel** ha molte matrici predefinite, tra le più famose:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

5.4.5 Operatore di Kirsch

L'**operatore di Roberts** ha 8 maschere direzionali, ognuna delle quali risponde massimamente per ciascun *edge* in una determinata direzione. Il **valore massimo** tra gli 8 valori stimati è il valore di output del gradiente in quel punto, mentre l'indice della maschera codifica le direzioni dell'*edge*.

Un esempio di maschere direzionali:

Horizontal line: $D_0 = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$

45 line: $D_{45} = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$

Vertical line: $D_{90} = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$

135 line: $D_{135} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$

5.4.6 Operatore di Robinson

L'**operatore di Roberts** ha 8 maschere direzionali, simili a quelle di Kirsch.

5.4.7 Confronto tra operatori

Dati gli operatori per estrarre l'*edge*:

1. Gradiente
2. Roberts
3. Prewitt
4. Sobel

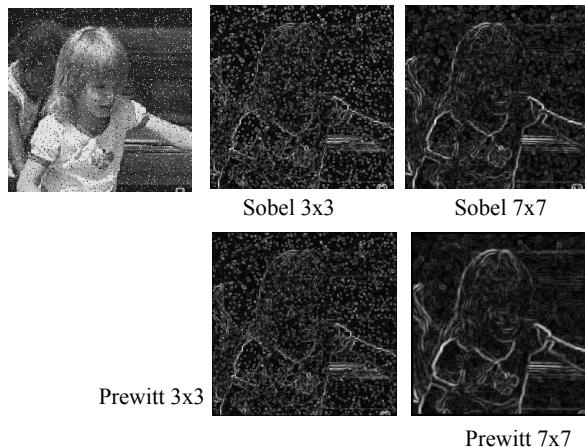
È possibile osservare come più si è **vicini al gradiente**, più:

- La localizzazione è buona;
- Aumenta la sensibilità al rumore;
- Diminuisce la rilevazione, diventa scarsa con il gradiente.

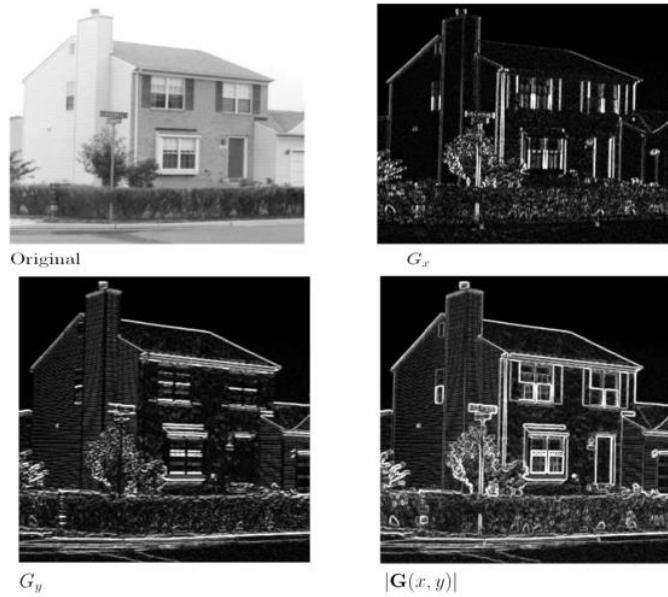
Al contrario, **vicino all'operazione di Sobel**:

- Diminuisce la localizzazione;
- Diminuisce la sensibilità al rumore;
- Aumenta la rilevazione.

Un esempio di applicazione dei vari operatori:



Invece, l'applicazione del gradiente è visibile nella seguente figura. Interessante notare le singoli applicazioni, ovvero solo lungo x o y :



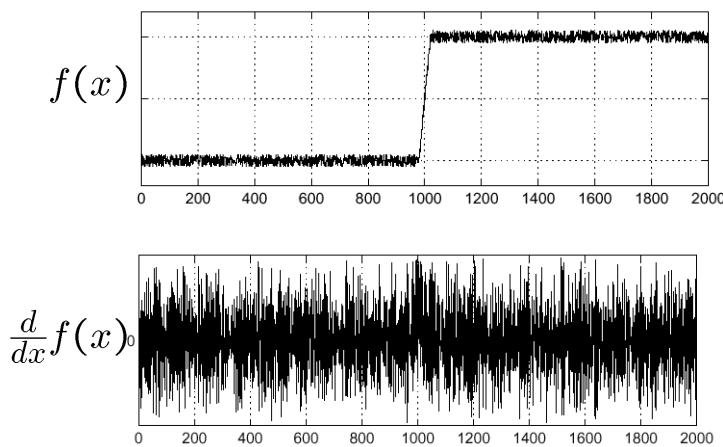
5.5 Derivata prima

5.5.1 Problemi

Gli operatori di **derivata prima** vanno applicati per ottenere il gradiente. Tuttavia, presentano dei problemi poiché **estraggono troppi punti di edge**.

5.5.2 Effetti del rumore

Si consideri una singola riga o colonna dell'immagine. Si traccia l'intensità in funzione della posizione del segnale. Si può vedere come la **derivata prima amplifichi il rumore**:



Per **risolvere** questo problema, si applica uno *smoothing* preliminare.

5.5.3 Rilevazione di *edge* o *detection* robusta

Per rilevare l'*edge* o eseguire una detection robusta, si eseguono tre fasi fondamentali:

- **Filtraggio** (*noise smoothing*): si vuole ridurre il rumore preservando gli *edge* reali;
- **Rinforzo** (*enhancement, sharpening*): facilita il rilevamento dell'*edge*, evidenziando le variazioni locali;
- **Rilevamento** (*detection*): decisione sulla presenza o meno dei punti di *edge*.

Tuttavia, si tenga in considerazione che:

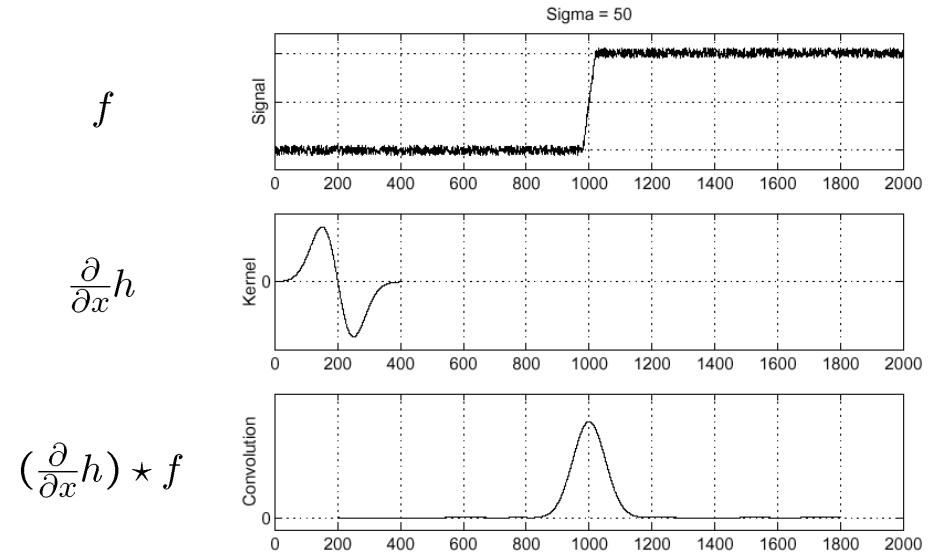
- ☛ Non tutti i punti aventi gradiente diverso da zero sono *edge*;
- ☛ È richiesto tipicamente una fase di sogliatura;
- ☛ È richiesto talvolta una fase di localizzazione a risoluzione *subpixel* e una fase di assottigliamento (*thinning*)

5.5.4 Teorema di convoluzione

Applicando il **teorema di convoluzione**:

$$\frac{\partial}{\partial x} (h * f) = \left(\frac{\partial}{\partial x} h \right) * f$$

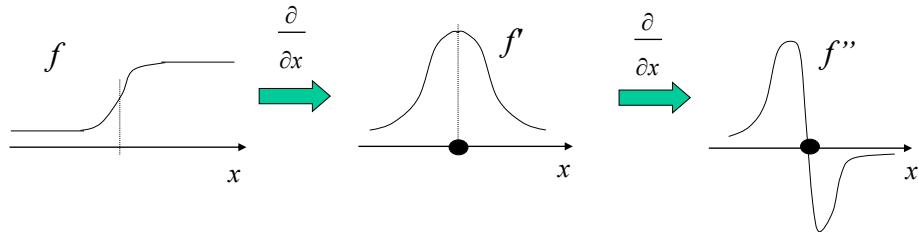
In questo modo, si risparmia un'operazione di filtraggio. Esempio:



5.6 Derivata seconda

5.6.1 Operatori di derivata seconda

Gli **operatori di derivata seconda** rilevano il punto di *edge* mediante il passaggio per lo zero (*zero crossing*) del valore calcolato, e per questo motivo sono **più precisi** (non necessitano di stabilire una soglia).



Nell'immagine è visibile l'applicazione delle varie derivate. Con la derivata prima si evidenzia la massima discontinuità tra i pixel antistanti, mentre con la derivata seconda c'è la massima variazione, ovvero il valore è prossimo allo zero.

5.6.2 Operatore Laplaciano

L'**operatore Laplaciano** equivale alla derivata seconda in 2 dimensioni:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Approssimando con l'equazione alle differenze e centrando la stima nel pixel (i, j) si ha:

$$\frac{\partial^2 f}{\partial x^2} = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

$$\frac{\partial^2 f}{\partial y^2} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

Combinando in un singolo operatore, si nota che è isotropico:

$$\nabla^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\nabla^2 = \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

$$\nabla^2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\nabla^2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

5.7 Laplaciano di Gaussiana

L'**operatore Laplaciano di Gaussiana** consiste in un filtraggio Gaussiano e Laplaciano. È il metodo migliore per estrarre gli *edge*. Con questo operatore si può osservare che:

- Il filtro di *smoothing* è Gaussiano;
- La fase di rinforzo è la derivata seconda, cioè Laplaciano;
- Il rilevamento è la fase di attraversamento per lo zero;
- La localizzazione è fatta mediante interpolazione lineare.

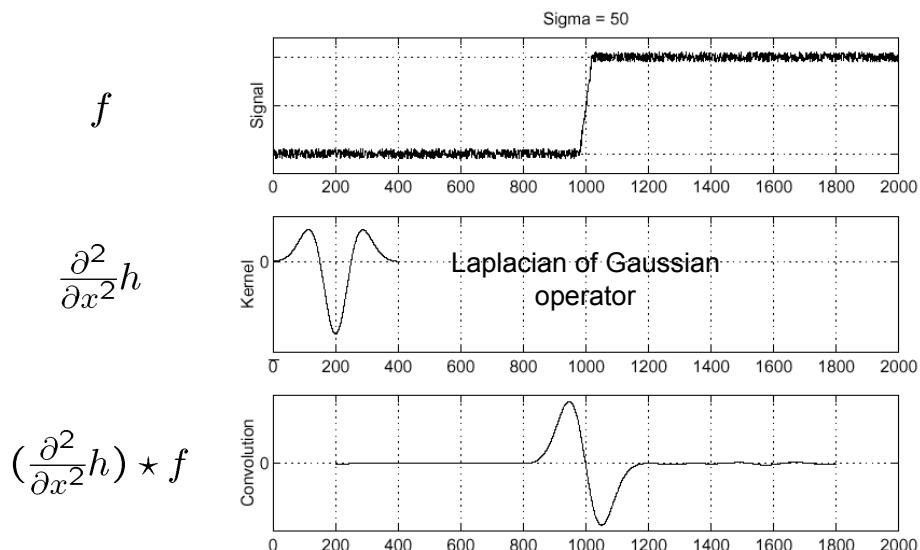
L'**applicazione** di questo filtro Gaussiano ha i seguenti fini:

- **Riduzione del rumore** per punti isolati e piccole configurazioni;
- “**Addolcimento** degli *step edge*”;
- “**Diffondere (blur)** gli *edge*”.

Considerando:

$$\frac{\partial^2}{\partial x^2} (h * f)$$

I grafici sono:



L'operatore laplaciano viene indicato con la lettera nabla al quadrato ∇^2 :

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Considerando *edge* soltanto i punti in cui:

- Laplaciano passa per lo zero;
- Gradiente locale è massimo (maggiore di una soglia quindi).

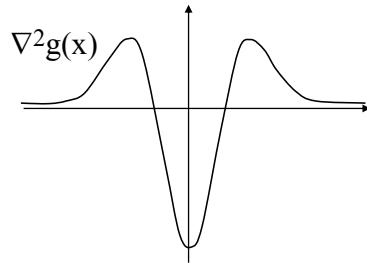
Allora l'uscita dell'operatore Laplaciano di Guassiana è $h(x, y)$:

$$h(x, y) = \nabla^2 [g(x, y) * f(x, y)] = [\nabla^2 g(x, y)] * f(x, y)$$

Dove:

$$\nabla^2 g(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{2\sigma^4} \cdot \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

Quest'ultimo viene chiamato ***mexican hat*** poiché ha la forma di cappello messicano:



Adesso, grazie all'introduzione di questo operatore, le due seguenti operazioni sono **equivalenti**:

1. Eseguire la convoluzione dell'immagine con un filtro Gaussiano di *smoothing* e calcolare il Laplaciano del risultato;
2. Eseguire la convoluzione dell'immagine con un filtro lineare, ovvero il filtro Laplaciano di Guassiana.

Si introduce un **esempio** di maschera 5×5 :

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Data la formula del filtro, con diversi valori della varianza σ , si ottengono diversi risultati:

- **σ grande:**

- ✓ Riduzione del rumore migliore;
- ✗ Perdita notevole di informazioni utili
- † Esempi: fusione di edge vicini, presenza di edge falsi, localizzazione imprecisa

- **σ piccolo:**

- ✓ Riduzione del rumore minore;
- ✗ Riduzione di perdita di informazioni utili
- † Esempio: molti edge spuri

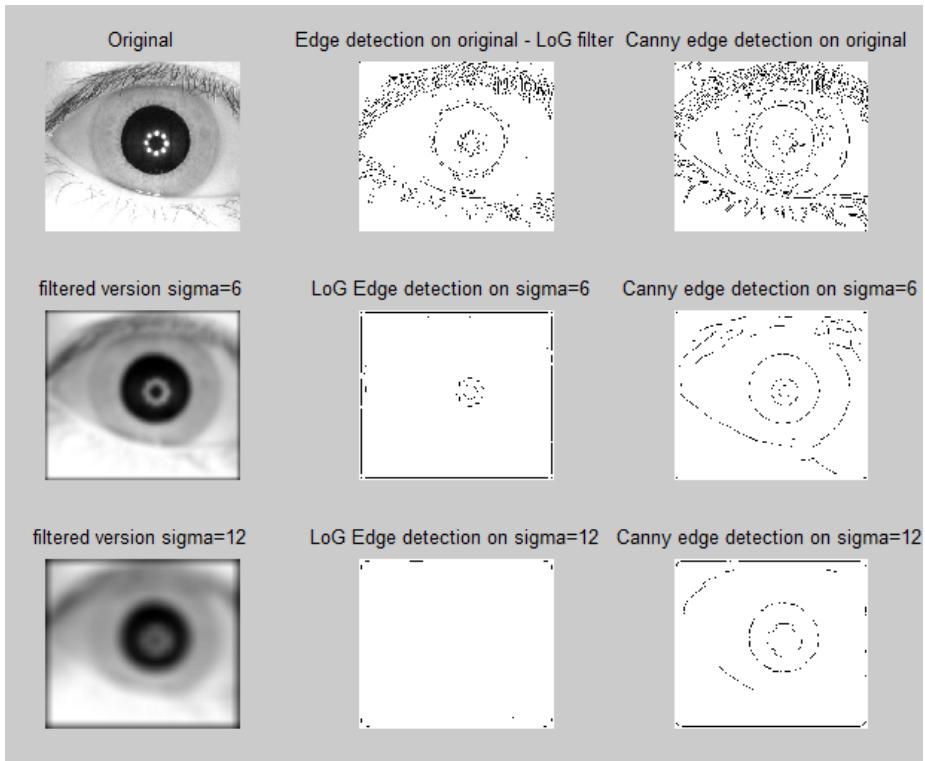
Per attenuare i difetti e cercare di ottenere un risultato ottimale, una **soluzione** comune è il **filtraggio a diversa scala** (diverse varianze):

- **σ grande:** pochi edge robusti, ma mal localizzati;
- **σ piccolo:** miglior localizzazione degli edge robusti.

Al **termine di un'applicazione di filtro Laplaciano di Gaussiana**, è necessario fondere i risultati delle varie scale ottenute, secondo alcuni criteri. Per esempio:

- Se ci sono *edge* in tutte le scale;
- Se ci sono *edge* almeno su K scale;
- E così via.

L'esempio nella prossima pagina, dovrebbe chiarire maggiormente la teoria di questo filtro:



5.8 Gradiente e Laplaciano di Gaussiana a confronto

L'operatore **gradiente** è efficace nel caso di immagini con poco rumore e transizioni nette.

Al contrario, l'operatore **Laplaciano di Gaussiana**, insieme allo *zero crossing*, offre un'alternativa quando il rumore è elevato e le transizioni non sono nette. Infatti, lo *zero crossing* viene aggiunto poiché offre un modo affidabile per localizzare l'edge. Mentre le proprietà di *smoothing* del Laplaciano di Gaussiana riducono gli effetti del rumore.

5.9 Filtro di Canny

5.9.1 Definizione

Il **filtro di Canny** è uno degli strumenti più efficienti. È il contrapposto all'operazione di Laplaciano di Gaussiana e viene considerato il “più esoso computazionalmente”. L'**algoritmo** consiste in 3 passi:

1. **Blur**, ovvero il filtro di rinforzo;
 2. **Edge thinning**, ovvero *non maxima suppression*;
 3. **Sogliatura ad isteresi + edge linking**.
-

5.9.2 Modello di *edge* per Canny

Il **modello di *edge* per Canny** è lo *step edge*, i cui descrittori sono:

- **Posizione o centro**, la posizione in cui l'*edge* è rilevato lungo la perpendicolare all'*edge* (solitamente memorizzata come un'immagine binaria);
- **Normale o orientazione o direzione**, versore della massima variazione di intensità nel punto di *edge* (perpendicolare al contorno di cui l'*edge* fa parte);
- **Forza**, modulo gradiente.

Il **modello di *step edge*** è il seguente:

$$G(x) = \begin{cases} 0 & x < 0 \\ A & x \geq 0 \end{cases}$$

Le ipotesi di base sono tre:

- Il filtro di rinforzo è lineare;
 - Il filtro deve essere ottimo per gli *edge* con rumore;
 - Il rumore è additivo, bianco e Gaussiano.
-

5.9.3 Algoritmo di Canny

L'**algoritmo di Canny** si divide in tre passaggi. Dato in ingresso l'immagine I :

1. Si applica il **filtro di rinforzo** ad I ;
2. Si applica l'algoritmo **Non-Maxima Suppression** all'output del passaggio precedente;
3. Si applica la **Sogliatura ad Isteresi** all'output del passaggio precedente.

Il filtro di rinforzo, la non-maxima suppression e la sogliatura ad isteresi, saranno argomenti trattati nei prossimi paragrafi.

5.9.4 Filtro di Rinforzo

Il **filtro di rinforzo** per essere applicato necessita di una serie di passaggi:

1. Si applica all'immagine I il filtro di *smoothing* Gaussiano:

$$J = G * I$$

Ovvero la convoluzione, con G che è la Gaussiana con media nulla e varianza σ .

2. Si calcola il gradiente $dJ = [J_x, J_y]$ e se ne stima il modulo e direzione:

$$e_s(i, j) = \sqrt{J_x^2(i, j) + J_y^2(i, j)} \implies \text{img } e_s$$

$$e_o(i, j) = \arctan \frac{J_y(i, j)}{J_x(i, j)} \implies \text{img } e_o$$

Il valore della varianza σ da utilizzare, dipende dalla lunghezza dei contorni, dal livello di rumore e dal compromesso tra localizzazione e rilevamento.

Esempio di immagine con $\sigma = 1$:



Esempio di immagine con $\sigma = 1.3$:



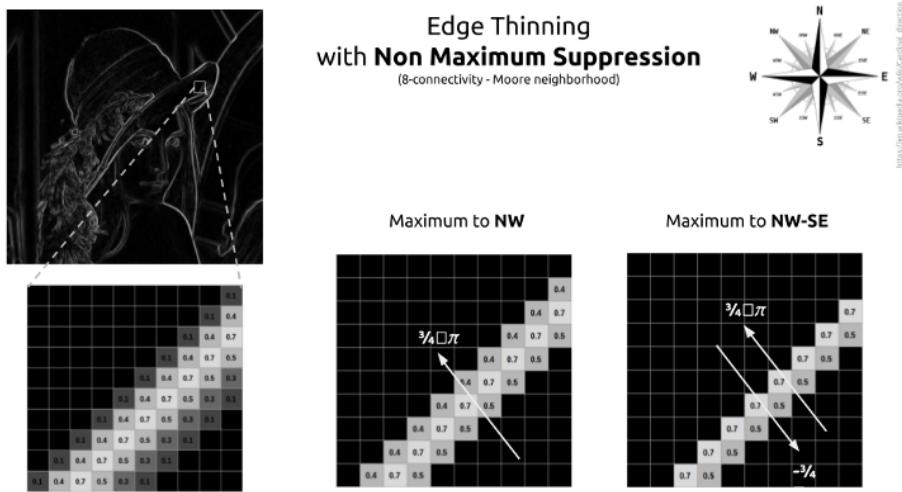
5.9.5 Non-Maxima Suppression

Il **filtro Non-Maxima Suppression** produce *edge* di un solo pixel scartando i massimi locali.

Il suo algoritmo è così strutturato:

- Riceve in input:
 - e_s , ovvero l'immagine della forza degli *edge*;
 - e_o , ovvero l'immagine con il gradiente orientato;
 - d_1, d_2, d_3, d_4 , che rappresentano le direzioni degli *edge* (per esempio, 0, 45, 90, 135).
- Per tutti i pixel con coordinate (i, j) :
 - Trovare la direzione d_k che approssima meglio la direzione del gradiente in quella posizione $e_o(i, j)$;
 - Se $e_{i,j}$ è minore di almeno uno dei suoi due vicini lungo d_k , allora assegna al risultato $I_N(i, j) = 0$, altrimenti $I_N(i, j) = e_s(i, j)$.
- Il risultato (output) è $I_N(i, j)$, ovvero l'immagine con *edge* più sottili (così da evitare *edge* troppo vicini o antistanti).

Un esempio di applicazione:



Partendo dall'immagine di sinistra e andando verso l'immagine di destra, viene eseguita un'eliminazione dei valori più piccoli (spiegato in modo approssimativo, si guardi l'algoritmo per capire meglio).

5.9.6 Sogliatura ad isteresi

La **sogliatura ad isteresi** non è altro che un confronto tra diverse soglie. Tuttavia, questa operazione nasce da un problema.

Il problema della sogliatura: l'immagine creata dalla soppressione dei non massimi (Non-Maxima Suppression, paragrafo precedente), contiene ancora punti spuri creati dal rumore.

Solitamente, viene definita una soglia TH (*threshold high*) e si scartano i pixel la cui forza è minore di TH , ma questo comporta degli svantaggi a seconda della soglia:

- Con **soglia piccola**, vengono rilevati anche gli *edge* deboli e di conseguenza si ottengono anche massimi spuri (falsi contorni);
- Con **soglia grande**, i valori dei veri massimi lungo i contorni possono fluttuare intorno alla soglia, frammentando così il contorno.

Ecco che la **soluzione** che si applica è la **soglia ad isteresi**.

L'**algoritmo** è il seguente:

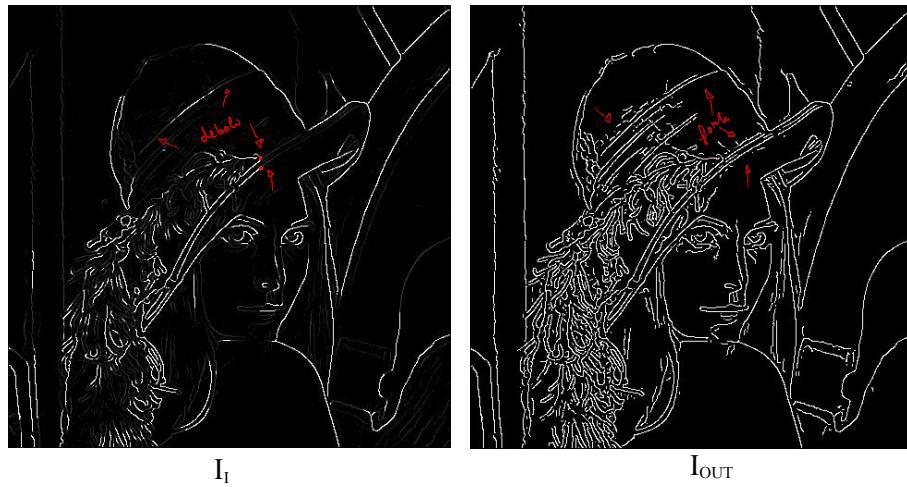
- Riceve in input:
 - I_N , ovvero l'immagine di uscita dell'algoritmo di Non-Maxima Suppression (paragrafo precedente).
- Vengono scelti due valori: soglia forte (TH , *threshold high*) e soglia debole (TL , *threshold low*)
- Viene analizzato ogni punto di *edge* a seconda che esso sia:
 - $> TH$, che corrisponde ad un punto di *edge* forte;
 - $\geq TL$ and $\leq TH$, che corrisponde ad un punto di *edge* debole;
 - $< TL$, che corrisponde ad un punto di *edge* ininfluente e da non considerare.
- Il risultato (output) è l'immagine I_I .

5.9.7 Edge linking

L'operatore **edge linking** viene utilizzato nell'algoritmo di Canny e ha il seguente algoritmo:

- Per tutti i punti di *edge* ottenuti dalla sogliatura ad isteresi, quindi dall'immagine I_I :
 - Si localizza il prossimo punto di *edge* $I_I(i, j)$ tale che $I_I(i, j) > TH$;
 - Partendo da $I_I(i, j)$ si segue la catena di massimi locali connessi nelle due direzioni perpendicolari alla normale dell'*edge* finché $I_I(i, j) > TL$ (memorizzando i risultati).
- Il risultato finale è l'immagine I_{OUT} e un insieme di liste, ognuna descrivente la posizione degli *edge*, la forza e l'orientazione dell'*edge* linkato a cui si fa riferimento.

Un **esempio** di *edge linking* con delle piccole frecce rosse per indicare gli effetti. A sinistra, l'immagine dopo la sogliatura ad isteresi e a destra l'immagine dopo l'*edge linking*:



5.10 Prestazioni di un *edge detector*

Per misurare le **prestazioni di un edge detector**, si seguono alcuni **criteri**:

- Probabilità di falsi *edge* (*false positives*);
- Probabilità di *edge* mancanti (*false negatives*);
- Errore nella stima dell'orientazione;
- Distanza media al quadrato della stima dal vero *edge*.

I **metodi** di valutazione sono principalmente due: (1) si eseguono delle prove sulle immagini sintetiche e sul rumore noto, (2) molti parametri da considerare.

5.11 Figura di merito per gli *edge*

Inoltre, si esegue una **misura quantitativa e oggettiva**, valida per ogni *edge detector*, così da ottenere una **figura di merito per gli *edge***.

Infine, esistono tre **tipi di errore**:

- *Edge* validi mancanti;
- Errore nella localizzazione;
- Classificazione di rumore come *edge*.

La **formula** della figura di merito è:

$$F_M = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + d_i \alpha^2}$$

Con:

- I_A , *edge* rilevati
- I_I , *edge* ideali
- d , distanza minima tra rilevati e ideali
- α , costante di penalizzazione

5.12 Trasformata di Hough

5.12.1 Definizione

La **trasformata di Hough** viene utilizzata dopo aver estratto gli *edge* da un'immagine ed è conosciuta come una tecnica di *object detection*⁸, basata su criteri geometrici.

Dato che l'individuazione di un oggetto è difficile per vari motivi, questa trasformata propone una soluzione. Infatti, trasforma il problema dalla ricerca di una curva, ad una banale **ricerca di massimi**.

In generale, una curva piana è definita in forma analitica tramite un insieme di parametri. Un'equazione lega tale parametri alle coordinate cartesiane con la seguente relazione:

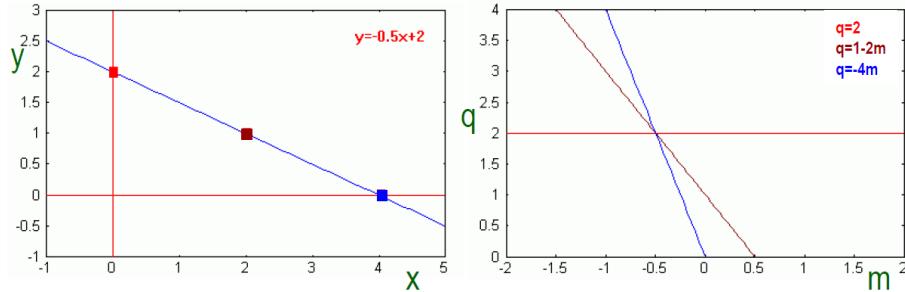
$$f((x, y), (a_1, a_2, \dots, a_n)) = 0$$

In cui (x, y) è un punto della curva nello **spazio immagine** (SI), e (a_1, a_2, \dots, a_n) è una n -upla di valori che individuano un punto nello **spazio dei parametri** (SP). Un punto nello spazio dei parametri individua univocamente una curva analitica.

Per **esempio**, la **retta** $y = mx + q$ può essere riscritta come:

$$f((x, y), (m, q)) = y - mx - q = 0$$

Quindi, fissato un punto (x_i, y_i) nello spazio immagine SI, l'equazione $q = y_i - mx_i$ descrive un insieme di curve (che rimangono ancora una retta) nello spazio dei parametri SP. Graficamente:



⁸Per *object detection* si intende l'individuazione precisa della classe di un oggetto in un'immagine

I **cerchi di raggio noto**, generano uno spazio dei parametri SP bidimensionale e la curva generata da ogni punto nello spazio immagine è essa stessa un cerchio:

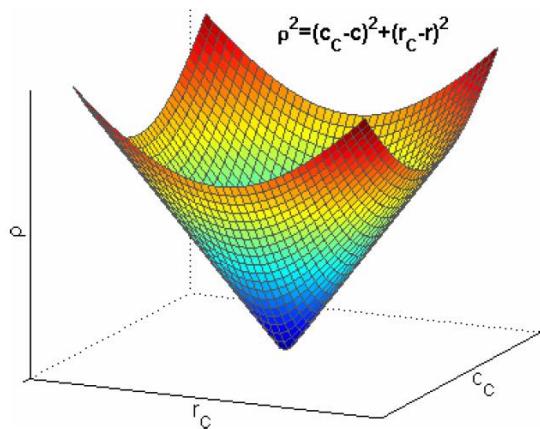
$$(y - y_c)^2 + (x - x_c)^2 = r^2$$

Lo spazio dei parametri SP è generato dalle coordinate del centro (x_c, y_c) .

Nel caso in cui il **raggio è incognito**, lo spazio dei parametri SP è tridimensionale:

$$f((c, r), (c_c, r_c), \rho) = (r - r_c)^2 + (c - c_c)^2 - \rho^2 = 0$$

In questo caso, la figura generata è un cono:



Ricapitolando:

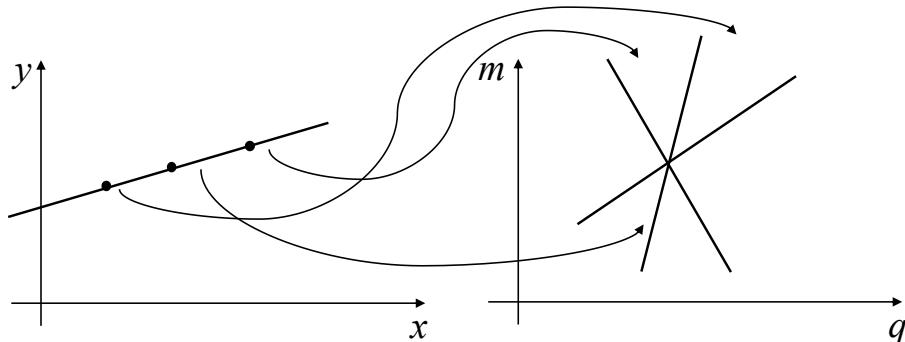
- ✓ Ogni punto nello spazio immagine SP genera una curva o superficie nello spazio dei parametri SP;
- ✓ Nello spazio immagine SP, un'intersezione di molte curve o superfici è l'indizio della presenza di una particolare istanza della curva analitica cercata;
- ✓ In generale, occorre un numero di punti almeno pari al numero dei parametri per individuare una curva, cioè le sue caratteristiche;
- ✓ La soglia alta HT permette quindi di convertire un problema di ricerca di curve in quello più semplice di ricerca di intersezioni.

5.12.2 Trasformata per le rette

Un punto P vota per il fascio di rette che passano per P . Due punti P e Q votano ciascuno per un fascio diverso, e vi è una sola retta in comune ai due fasci (che prende due voti): quella passante per P e Q .

I parametri che identificano una retta sono m e q nell'equazione $y = mx + q$ e un fascio è individuato da un insieme di valori m e q . Dunque, un punto $P = (x, y)$ vota per un insieme di valori m, q che rappresentano il fascio di rette passanti per P .

Il punto P vota per tutti i valori di p e q che soddisfano la solita equazione della retta $y = mx + q$ con x, y fissati. Il punto P vota per la retta nello spazio m, q dove il coefficiente angolare e il termine noto sono dati da x e y . Quindi, si passa da una rappresentazione nello spazio cartesiano x, y ad un'altra rappresentazione nello spazio dei parametri m, q :

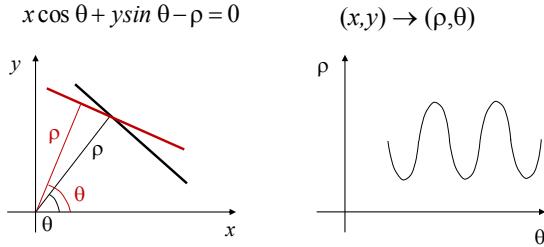


Nel caso n punti giacenti su una retta, tutti vengono mappati in un insieme di rette nello spazio dei parametri, tutte passanti per il punto (m, c) che quindi sarà maggiormente votato. A questo punto, il **problema si trasforma da line detection a peak detection** nello spazio dei parametri.

Tuttavia, questo implica:

- Delimitazione e discretizzazione dello spazio dei parametri;
- La ricerca di massimi assoluti per trovare le rette;
- Soppressione dei massimi locali per far fronte a picchi spuri nell'intorno di quelli veri.

In parole poche, viene **utilizzata la forma polare** perché più comoda, dato che: θ è delimitato, ρ si può delimitare specificando una regione circolare attorno all'origine.



Quindi, ad un fascio di rette passante per x, y , corrisponde una curva sinusoidale nello spazio dei parametri SP.

Per concludere, l'**algoritmo** corrisponde a:

- In input riceve l'immagine binaria $M \times N$;
- Viene definito lo spazio dei parametri SP (1 *edge* e 0 *no edge*):

$$\rho \in [0, \sqrt{M^2 + N^2}], \quad \theta \in [0, \pi] \rightarrow 0, 180$$

- Viene eseguita la discretizzazione di (ρ, θ) in (ρ_d, θ_d) usando come “passo” un valore accettabile a seconda del problema da risolvere (più precisione o più computazione);
- Sia A la matrice $R \times T$ risultante, inizializzata a zero;
- Allora, per ogni pixel $I(x, y) = 1$ e per $h = 1, \dots, T$:
 - Sia $\rho = x \cos(\theta_d(h)) + y \sin(\theta_d(h))$
 - Si trova l'indice k tale che $\rho_d(k)$ è l'elemento più vicino a ρ ;
 - Viene incrementato $A(k, h)$ di un'unità.
- Infine, vengono trovati tutti i massimi locali (k_ρ, h_ρ) tale che $A(k_\rho, h_\rho) > \tau$ soglia.

L'output finale è un insieme di coppie $[\rho_d(k_\rho), \theta_d(h_\rho)]$ che identifica le rette.

Alcune **considerazioni**:

- ✓ La HT (*high threshold*) non richiede una fase di *linking* o raggruppamento preliminare;
- ✓ È robusta al rumore;
- ✗ Data la presenza del rumore e dell'inesistenza della discretizzazione, non esiste un picco ben marcato. Di conseguenza, è necessario un piccolo insieme di accumulatori con valore elevato e viene usato il calcolo del centroide;
- ✗ Lo spazio dei parametri SP cresce col numero dei parametri, quindi computazionalmente oneroso per modelli complessi (risolvibile parzialmente con l'uso del gradiente).

5.12.3 Trasformata per i cerchi

L'equazione del cerchio in forma implicita è:

$$(x - a)^2 + (y - b)^2 = r^2$$

Con a, b coordinate del centro e r il raggio.

Lo spazio dei parametri SP è multidimensionale.

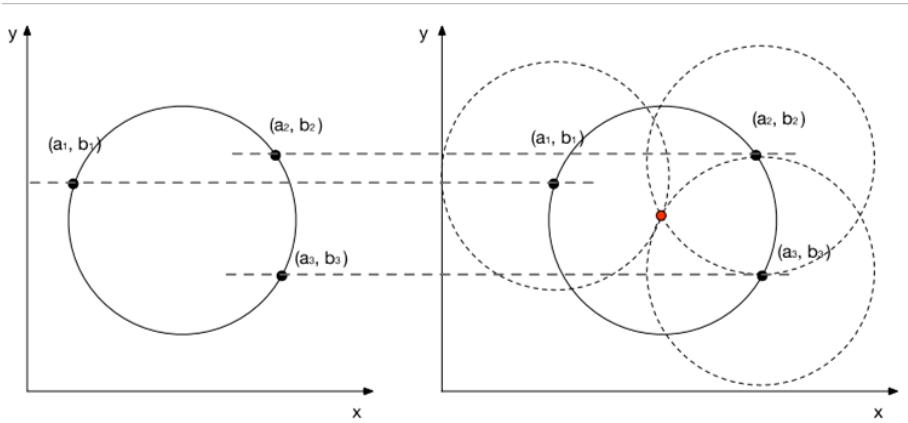
Talvolta, l'**equazione parametrica** del cerchio attraverso seni e coseni permette di rendere la trasformata **più semplice** da trattare:

$$\begin{cases} r = \frac{x - a}{\cos \theta} \\ b = y - (x - a) \tan \theta \end{cases}$$

Per **esempio**, se si è a conoscenza del raggio del cerchio r con a, b coordinate del centro:

$$(x - a)^2 + (y - b)^2 = r^2$$

Ogni punto genera un cerchio e intersezioni tra cerchi nello spazio dei parametri SP indicando il probabile centro.

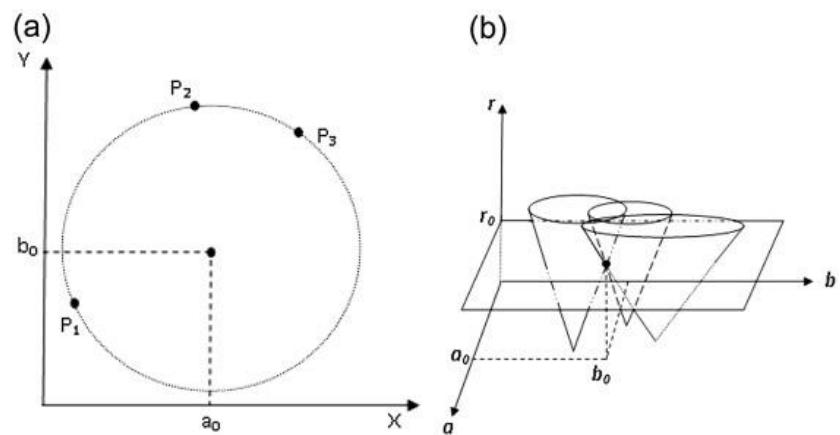


Attenzione! Se non si è a conoscenza del raggio del cerchio r , ma si è in grado di **calcolare il gradiente del cerchio** nel punto:

$$b = a \tan \theta - x \tan \theta + y$$

Che costituisce una linea nello spazio dei parametri. Inoltre, le intersezioni tra rette negli spazi dei parametri, indicano il probabile centro.

Infine, se non si è a conoscenza del cerchio e non è possibile calcolare gli angoli rispetto alla circonferenza, si utilizza l'intersezione di coni.



6 Laboratorio MATLAB

6.1 Introduzione a MATLAB

I comandi base di MATLAB non verranno spiegati poiché già spiegati ampiamente nel corso di Probabilità e Statistica.

6.1.1 Script e funzioni

Uno **script** è un insieme di comandi MATLAB. Essi possono essere dei semplici file con estensione “.m” e possono essere eseguiti (e richiamati) se e solo se sono nella stessa cartella del file chiamante.

Una **funzione** è una lista di comandi che necessita di **variabili di input** per essere eseguita e **restituisce variabili di output**. Come per gli script, l'estensione del file deve essere “.m” e il file della funzione deve iniziare con la seguente riga di codice:

```
1 function [output] = nome_function (input)
```

- **function** è la keyword che indica l'inizio della funzione;
- **output** è il nome del valore di output;
- **nome.function** è il nome della funzione che deve corrispondere al nome del file;
- **input** è il nome del valore di input.

Tutte le variabili definite internamente sono locali e di conseguenza non verranno viste al di fuori.

6.1.2 For, while, if

Il ciclo **for** in MATLAB è utilizzato per ripetere un insieme di istruzioni per un numero predeterminato di iterazioni. Deve terminare con la keyword **end** e la sua sintassi è la seguente:

```
1 for index = values
2     statements
3 end
```

Un **esempio**:

```
1 for i = 1:5
2     disp(i)
3 end
```

Con output: 1, 2, 3, 4, 5.

Il ciclo **while** e il costrutto condizionale **if** funzionano allo stesso identico modo.

6.1.3 Condizioni su vettori e matrici (`find`)

È possibile effettuare **operazioni logiche** su **vettori e matrici**. Per **esempio**:

```
1 x = [2 -1 8 0];
2 x > 0
```

L'output sarà un vettore contenente vero o falso sotto forma di numeri, ovvero 1 se il valore in quella posizione è maggiore di zero, altrimenti 0:

$$ans = [1 \ 0 \ 1 \ 0]$$

Queste condizioni di solito vengono utilizzate per modificare alcuni elementi di vettori. Per **esempio**:

```
1 x = [2 -1 8 0];
2 idx = x > 0;
3 x(idx) = 100
```

L'output del programma sarà il vettore x contenente il valore 100 nelle sole posizioni in cui la condizione ($x > 0$) era vera. Quindi:

$$x = [100 \ -1 \ 100 \ 0]$$

Invece, per **ottenere gli indici** è possibile utilizzare il comando `find`. Esso ritorna il numero di riga e colonna dei valori che hanno rispettato la condizione:

```
1 x = [2 -1; 8 0];
2 tmp = x > 0;
3 [riga, colonna] = find(tmp)
```

L'output:

$$\begin{aligned} riga &= 1 \ 2 \\ colonna &= 1 \ 1 \end{aligned}$$

6.1.4 Esercizio 1

Definire i seguenti tre vettori:

- A vettore riga che contiene i numeri pari da 2 fino a 20;
- B vettore riga con tutti i numeri da -22 a -13;
- C vettore riga con 10 valori uguali a 0.

A partire da questi, effettuare le seguenti operazioni:

- Creare una matrice **MatX** dove le righe sono costituite da A, B e C (in questo ordine);
- Verificare e salvare le dimensioni di **MatX** e il numero di elementi;
- Estrarre la sotto-matrice che contiene le prime due righe e le prime cinque colonne;
- Sostituire la seconda colonna di **MatX** con il valore 31;
- Creare una matrice **MatY** di numeri reali distribuiti in modo random (**randn**), con 4 righe e 10 colonne;
- Creare una matrice **MatZ** data dalla concatenazione di **MatX** e **MatY**;
- Verificare le dimensioni di **MatZ** ed estrarre la diagonale.

Soluzione

```
1 % A vettore riga che contiene i numeri pari da 2 fino a 20,
2 % B vettore riga con tutti i numeri da -22 a -13,
3 % C vettore riga con 10 valori uguali a 0.
4 A = [2:2:20];
5 B = [-22:-1:-13];
6 C = zeros(1,10);
7
8 % Creare una matrice MatX dove le righe sono costituite da A, B e C
     (in questo ordine)
9 MatX = [A;B;C];
10
11 % Verificare e salvare le dimensioni di MatX e il numero di
     elementi
12 [nr,nc] = size(MatX);
13 num = numel(MatX);
14
15 % Estrarre la sotto-matrice che contiene le prime due righe e le
     prime cinque colonne
16 MatXsub = MatX(1:2,1:5);
17
18 % Sostituire la seconda colonna di MatX con il valore 31
19 MatX(:,2) = 31;
20
21 % Creare una matrice MatY di numeri reali distribuiti in modo
     random (randn),
22 % con 4 righe e 10 colonne
23 MatY = randn(4,10);
24
25 % Creare una matrice MatZ data dalla concatenazione di MatX e MatY
```

```

26 MatZ = [MatX;MatY];
27
28 % Verificare le dimensioni di MatZ ed estrarre la diagonale
29 [nrz,ncz] = size(MatZ);
30 diagZ = diag(MatZ);

```

La funzione `numel` (riga 13) consente di contare il numero di elementi di un array. Dato che `MatX` è una matrice 3×10 , il valore di ritorno della funzione è 30.

6.1.5 Esercizio 2

- Generare un numero casuale con il comando `randn` (distribuzione normale standard);
- Assegnare alla variabile y il valore 1 se tale numero è compreso tra -1 e $+1$ (media \pm deviazione standard), 0 altrimenti;
- Se ripeto il procedimento 10000 volte, quante volte il numero casuale cade nell'intervallo $[-1 1]$?
- (EXTRA) Provare a risolvere l'esercizio anche senza usare cicli (suggerimento: consultate l'help della funzione `randn`).

Soluzione

```

1 % Generare un numero casuale con il comando randn (distribuzione
2 % normale standard).
3 x = randn;
4
5 % Assegnare alla variabile y il valore 1 se tale numero e' compreso
6 % tra -1 e 1 (media +- deviazione standard), 0 altrimenti.
7 if x < 1 && x > -1
8     y = 1;
9 else
10    y = 0;
11 end
12
13 % visualizziamo il numero generato
14 fprintf('Numero generato: %f\n',x) % funzione fprintf per stampare
15
16 % visualizziamo y
17 fprintf('E'' compreso fra -1 e 1?')
18 y
19
20 % - Se ripeto il procedimento 10000 volte, quante volte il numero
21 % casuale cade nell'intervallo [-1 1]?
22 cnt = 0;
23 for i = 1:10000
24     x = randn;
25     if x < 1 && x > -1
26         cnt = cnt+1;
27     end
28 end
29 fprintf('Numero di volte in cui il numero cade fra -1 e 1: %d\n',
30        cnt)

```

```

31 % EXTRA: Provare a risolvere l'esercizio anche senza usare cicli
32 % (suggerimento: consultate l'help della funzione randn)
33 x = randn([1 10000]); % Genero un vettore con 10000 numeri casuali
34 y = (x < 1) & (x > -1);
35 cnt = sum(y);
36 fprintf('Numero di volte in cui il numero cade fra -1 e 1: %d\n',
          cnt)

```

L'esercizio extra utilizza una sintassi più minimale e più efficace della precedente. Nonostante la leggibilità potrebbe perdersi, la riga 33 genera un vettore di 10'000 elementi; nella variabile *y* viene salvato il vettore contenente i valori 1 nel caso in cui sia vera la condizione, 0 altrimenti. Grazie a quest'ultima intuizione, facendo la somma dei valori 1, è possibile contare quanti valori hanno soddisfatto tale condizione.

6.1.6 Esercizio 3

Creare una funzione chiamata **Mymean** che dato un vettore o una matrice in ingresso restituisca il valore medio. Si ricorda che la funzione **Mymean** deve essere definita in un file che si chiama **Mymean.m** e deve iniziare con la seguente riga:

```
1 function [output] = Mymean (input)
```

Dove **input** e **output** sono rispettivamente l'ingresso e l'uscita della funzione. In particolare, nel caso di vettori la funzione **Mymean** restituisce un singolo valore medio, mentre per le matrici un vettore riga contenente il valore medio di ogni colonna.

Controllare che la funzione dia il risultato atteso (confronto con il risultato della funzione **mean**) di Matlab) con in ingresso un vettore riga, un vettore colonna e una matrice. Per esempio:

```

1 vec = [1:2:30];
2 vec_media = Mymean(vec);
3 media_Matlab = mean(vec);
4 confronto = [vec_media; media_Matlab]

```

Soluzione

Il codice della funzione che richiama **Mymean**:

```

1 % vettore riga
2 vec = [1:2:30];
3 vec_media = MYmean(vec);
4 media_Matlab = mean(vec);
5 confronto = [vec_media; media_Matlab];

6
7 % vettore colonna
8 vec1 = [2:2:30]';
9 vec_media1 = MYmean(vec1);
10 media_Matlab1 = mean(vec1);
11 confronto1 = [vec_media1; media_Matlab1];

12
13 % matrice
14 A = [vec' vec1];
15 mat_media = MYmean(A);
16 media_Matlab = mean(A);
17 confronto1 = [mat_media; media_Matlab];

```

E il codice della funzione Mymean:

```
1 function y = MYmean(x)
2 % Calcola il valor medio.
3 % Per i vettori, MYmean(x) restituisce il valor medio.
4 % Per le matrici, MYmean(x) restituisce un vettore riga contenente
5 % il valor medio di ogni colonna.
6 [m,n] = size(x);
7
8 % A questo punto devo controllare se e' un vettore riga (1xn,
9 % quindi m=1)
10 % oppure un vettore colonna (mx1, quindi n=1)
11 if m == 1 || n == 1
12     % vettore
13     y = sum(x)/length(x);
14 else
15     % matrice
16     % sum fa la somma colonna per colonna
17     y = sum(x);
18     % divido per il numero di righe
19     y = y/m;
20 end
```

6.2 Concetti avanzati di MATLAB

6.2.1 Debug

MATLAB consente di eseguire un debug del codice ottimo. Per entrare nella modalità debug è necessario inserire un breakpoint cliccando sul numero di riga del codice interessata (classico stile di debug); all'esecuzione del codice, MATLAB si fermerà alla linea in cui è stato inserito il breakpoint.

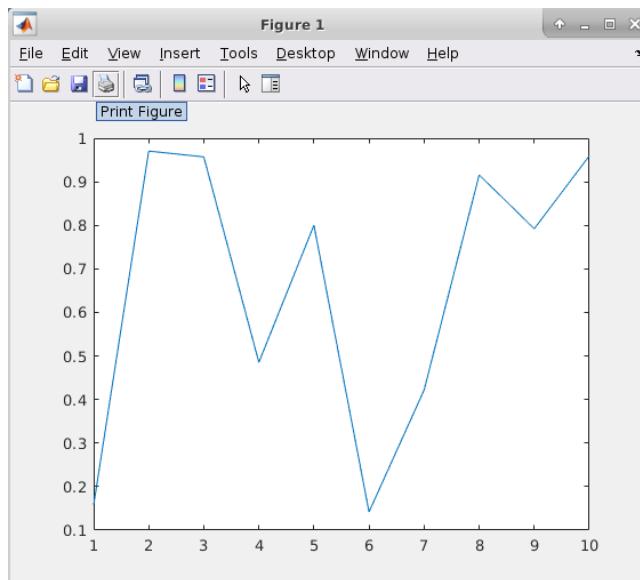
Comando	Descrizione
<code>dbstop if error;</code>	Entra in modalità di debug nel momento in cui riscontra un errore nel codice, in corrispondenza della riga che ha prodotto l'errore. Risulta utile per ispezionare il workspace alla ricerca del motivo dell'errore.
<code>dbstop if warning;</code>	Stessa descrizione di <code>dbstop if error</code> con la differenza che è uno <i>warning</i> ad attivare la modalità di debug.
<code>dbquit;</code>	Esce dalla modalità di debug.
<code>dbclear all;</code>	Rimuove tutti i breakpoint.
<code>CTRL + r</code>	Commenta la riga corrente o la porzione di codice selezionata.
<code>CTRL + t</code>	Elimina il primo commento sulla sinistra, se presente, o della porzione di codice selezionata.
<code>CTRL + i</code>	Identifica la riga di codice o la porzione di codice selezionata guardando l'intero <i>scope</i> dello script o funzione in cui ci si trova.

6.2.2 Rappresentazione grafica dei segnali

Le figure possono essere inizializzate con il comando `figure(number)`.

Il **line plot** è il tipo di grafico più utilizzato per rappresentare i segnali e richiede due vettori della stessa dimensione. Un vettore x che rappresenta le coordinate orizzontali di ogni punto e un vettore y che rappresenta i valori corrispondenti nell'asse delle ordinate. Per **esempio**:

```
1 A = [1:10];
2 B = rand(1,10);
3 plot(A,B)
```



È possibile specificare il colore della linea del grafico inserendo un terzo parametro nella funzione `plot`. Per **esempio**, si vuole un grafico identico a quello sopra ma con la linea rossa:

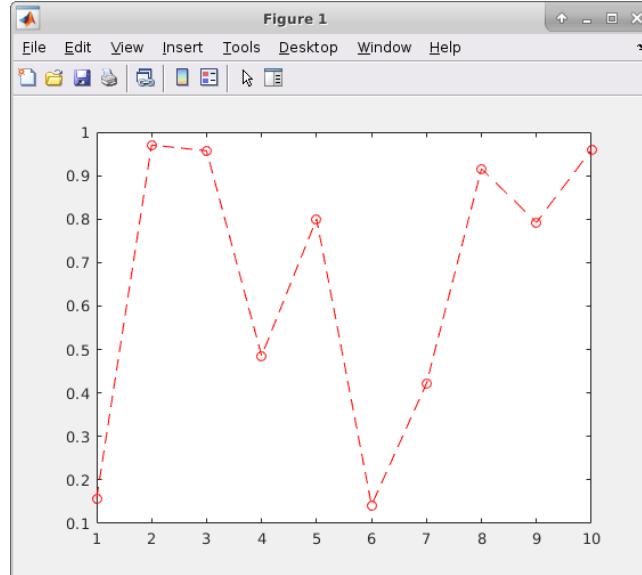
```
1 plot(A,B,'r')
```

I colori di MATLAB sono 8:

Comando	Colore
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

È possibile specificare anche la tipologia di linea e maker del grafico, per **esempio**:

```
1 plot(A,B,'--or')
```



Le varie tipologie di linee e markers:

Comando	Tipo di linea/marker
.	point
o	circle
x	x-mark
+	plus
*	star
s	square
d	diamond
v	triangle (down)
^	triangle (up)
<	triangle (left)
>	triangle (right)
p	pentagram
h	hexagram
-	solid
:	dotted
-.	dashdot
--	dashed
(none)	no line

Altri comandi che vengono utilizzati per rappresentare i grafici sono:

Comando	Descrizione
<code>title('titolo')</code>	Per settare il titolo
<code>xlabel('name')</code>	Per settare il nome dell'asse x
<code>ylabel('name')</code>	Per settare il nome dell'asse y
<code>axis([xmin xmax ymin ymax])</code>	Per settare minimo/massimo per asse
<code>xlim([], ylim([]))</code>	Per settare minimo/massimo per un asse
<code>legend('text')</code>	Per visualizzare una legenda a lato
<code>grid on/off</code>	Per visualizzare (o no) la griglia nel grafico
<code>hold on</code>	Per visualizzare più grafici sovrapposti
<code>clf</code>	Per pulire il contenuto di una figura
<code>stem</code>	Per visualizzare la sequenza dei dati Y come steli che si estendono per tutta la lunghezza X
<code>subplot</code>	Per creare più immagini all'interno di una singola figura

6.2.3 Rappresentazione dei suoni

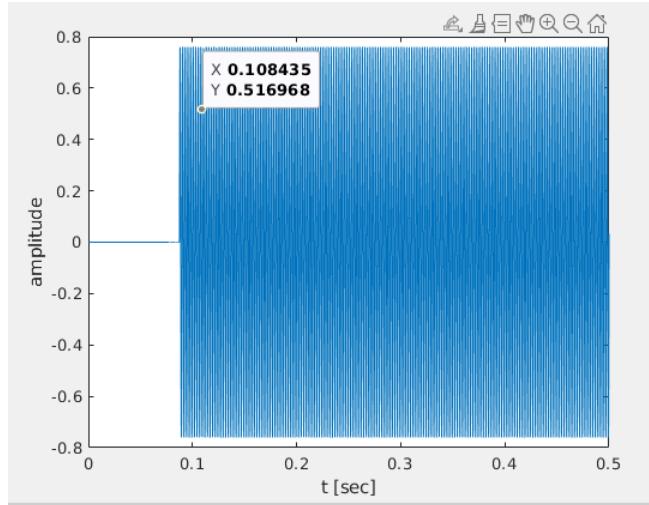
Per leggere un file contenente un suono è necessario il seguente comando:

```
1 [Y, FS] = audioread(FILENAME)
```

Consente di leggere un file audio, specificato nella stringa FILENAME, e di restituire i dati campionati in Y e la frequenza di campionamento FS, in Hertz.

Per visualizzare i suoni è possibile utilizzare le rappresentazioni grafiche ampiamente spiegate nel paragrafo 6.2.2. Un **esempio** di codice e del suo relativo grafico (si ricorda che l'operazione ./ prende ogni valore dell'array e lo divide per l'operando):

```
1 t = 1:size(y(1:Fs/2,1),1);
2 t = t ./ Fs;
3 figure; plot(t, y(1:Fs/2, 1))
4 xlabel('t [sec]')
5 ylabel('amplitude')
```



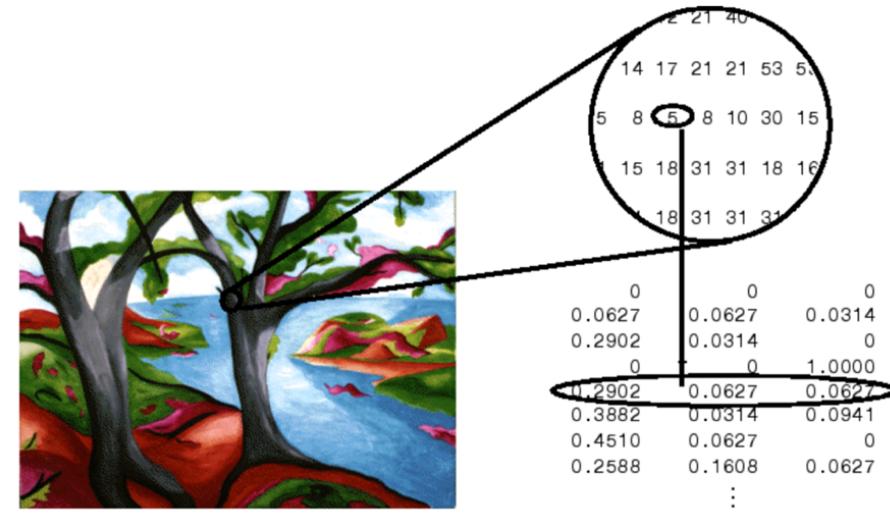
6.2.4 Rappresentazione delle immagini

In MATLAB esistono due tipologie di immagini:

- **Indicizzata**, una matrice di dati i cui valori rappresentano un “puntatore” al colore vero, contenuto in una mappa di colore.

Le immagini indicizzate sono **matrici di dati** chiamata di solito X e **una matrici di colori** chiamata map . Quest’ultima è un array ($m \times 3$) di double contenente valori a virgola mobile nell’intervallo $[0, 1]$ e **ogni riga specifica i componenti rosso, verde e blu di ogni singolo colore**.

Un’immagine indicizzata utilizza la **mappatura diretta dei valori dei pixel ai valori della mappa di colori**. Quindi, il colore di ciascun pixel dell’immagine viene determinato mappando il valore di X al corrispondente colore nella mappa di colori (i valori di X quindi devono essere numeri interi). Per **esempio**, l’immagine `trees.tif`:

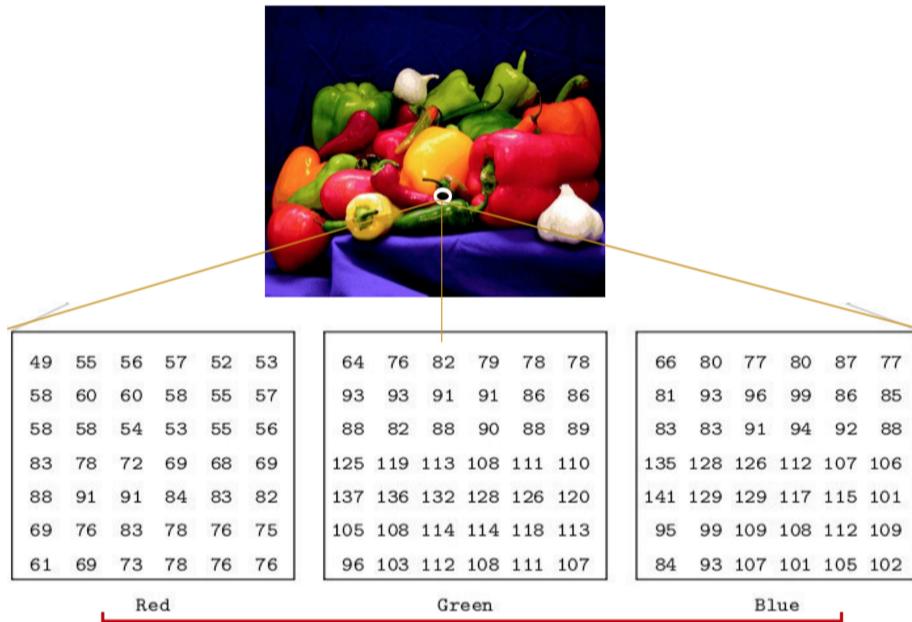


- **Di intensità**, una matrice di dati i cui valori rappresentano già i colori (in particolare rappresentano intensità all'interno di un intervallo).

Le immagini di intensità sono **matrici di dati I** , i cui valori rappresentano intensità all'interno di un intervallo:

- $M \times N$ (singolo canale): il valore di ogni pixel indica il suo livello di grigio;
- $M \times N \times K$ (3 canali): i tre valori di ogni pixel indicano il colore (tipicamente secondo la codifica RGB).

Per **esempio**, l'immagine `peppers.png`:



Per leggere un file contenente un'immagine in scala di grigi o a colori, e ottenere informazioni su di essa, viene messo a disposizione il comando `imread`:

```

1 I = imread(filename, fmt);
2 [I, map] = imread(...);
3 [I, map] = imread(filename);
4 [I, map] = imread(URL, ...);

```

Il parametro `filename` indica il nome del file e `fmt` il formato dell'immagine. A seconda del tipo di immagine, il comando ritorna valori differenti:

- Immagini **indicizzate**, il comando ritorna l'immagine nella variabile `I`, e la mappa di colore in `map`.
- Immagini di **intensità**, il comando ritorna l'immagine nella variabile `I`, mentre la mappa `map` è nulla.

Per entrambi i tipi di immagine, la matrice immagine I ha alcune caratteristiche a seconda dell'immagine in scala di grigio o a seconda dell'immagine di intensità a colori:

- Immagine in **scala di grigio o binaria**, quindi ha dimensione $M \times N$ e il prodotto sono il numero di pixel;
- Immagine di **intensità a colori**, che possono essere di dimensione $M \times N \times 3$ nel caso di modelli RGB o HSV, oppure di dimensione $M \times N \times 4$ nel caso di modelli CMYK (tipico dei file con estensione .tiff).

Per **visualizzare un'immagine** in scala di grigi o a colori contenuta in una matrice I , viene utilizzato il seguente comando:

```
1 imshow(I)
```

Aggiungendo alcuni parametri è possibile ottenere qualche informazioni in più:

- Con le **immagini indicizzate**, aggiungendo il parametro `map`, viene visualizzata un'immagine indicizzata con la relativa mappa di colore contenuta nella variabile `map`.
- Con **immagini in scala di grigi** vengono visualizzati i pixels con i valori all'interno dell'intervallo $[low \ high]$, gli altri valori saranno sostituiti con il colore nero, se il loro valore è minore di low , altrimenti, se maggiore, con il colore bianco.

Per **esempio**:

```
1 I = imread('cameraman.tif');
2 figure(1)
3 subplot(1,2,1), imshow(I)
4 subplot(1,2,2), imshow(I, [0 80])
```

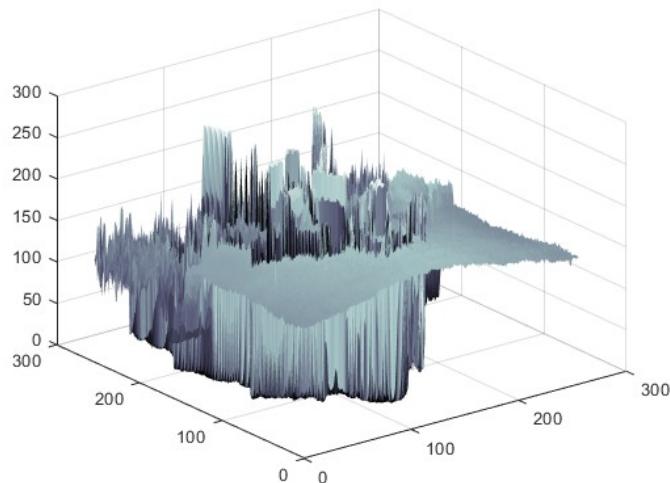


Un ulteriore metodo per **visualizzare un'immagine in scala di grigi o a colori contenuta in una matrice** A è quello di utilizzare il comando `imagesc(A)`. L'immagine ottenuta viene visualizzata utilizzando tutto il range di colori compreso nella mappa di colore (colormap). Il suo scopo è visualizzare una generica informazione bidimensionale, massimizzando l'utilizzo della mappa cromatica.

6.2.5 Ulteriori comandi utili per le immagini

Il comando `surf(I)` consente di visualizzare un'immagine `I` come un rilievo geografico, con valli e picchi. Risulta efficace per capire l'analisi in frequenza, per vedere l'effetto di operazioni quali estrazioni di edge, segmentazioni, e altro. Un **esempio**:

```
1 I = imread('cameraman.tif');
2 figure(1)
3 surf(I)
4 shading flat
5 colormap bone
```



Girando la figura:



Con il comando `imresize(A, scale)` è possibile **riscalare un'immagine** (a colori o in scala di grigi), rimpicciolendola oppure ingrandendola a seconda del valore del parametro `scale`. Supponendo:

```
1 B = imresize(A, scale)
```

Se `scale` ha valore:

- Compreso tra 0 e 1, allora $B < A$
- Maggiore di 1, allora $B > A$

Con il comando `imrotate(I, angle)` è possibile **ruotare un'immagine** in senso orario con valori di `angle` negativi, in senso antiorario con valori di `angle` positivi. Ovviamente `angle` è espresso in gradi.

Con il comando `imcrop(I)` è possibile visualizzare un'immagine e aprire un tool interattivo per selezionarne una porzione.

Con il comando `rgb2gray(RGB)` è possibile **convertire un'immagine da RGB a scala di grigi**.

Con il comando `imwrite(I, map, 'filename', 'fmt')` è possibile scrivere un file contenente un'immagine in scala di grigi o a colori. Il parametro `map` è necessario solo per le immagini indicizzate.

6.2.6 Esercizio 1

Prendere la foto di Paperino oppure fatevi una foto al volto. Copiate questa foto nella directory di lavoro, e caricatela attraverso MATLAB. Attraverso opportune indicizzazioni della matrice in cui è contenuta la foto, sostituite ai pixel che rappresentano gli occhi dei pixel neri, facendo comparire una sorta di occhiali da sole.

Si ricorda che il valore nero si ottiene con una terna $RGB = [0\ 0\ 0]$. Visualizzate l'immagine originale e quella modificata attraverso il comando `surf`, in due plot separati nella stessa figura. Per visualizzare con `surf` occorre trasformare la foto in scala di grigio. Per una visualizzazione ottimale, usare anche il comando “`shading flat`”.

Soluzione

Il codice dell'esercizio è il seguente:

```
1 A = imread('Paperino.jpg');
2 figure(1);
3 imagesc(A);
4 axis square
5 title('Immagine Originale')
6
7 % creo una copia
8 B = A;
9 % accedo alle posizioni per disegnare gli "occhiali"
10 B(163:209,90:136,:)=0;
11 B(163:209,164:211,:)=0;
12 B(185:188,137:162,:)=0;
13
14 % visualizzo
15 figure(2);
16 imagesc(B);
17 axis square
18 title('Immagine Modificata')
19
20 % - Visualizzate l'immagine originale e quella modificata
21 % attraverso il comando surf, in due plot separati nella
22 % stessa figura. Per visualizzare con surf occorre
23 % trasformare la foto in scala di grigio
24 figure(3);
25 subplot(1,2,1)
26 surf(rgb2gray(A))
27 shading('flat')
28 title('Immagine Originale')
29 subplot(1,2,2)
30 surf(rgb2gray(B))
31 shading('flat')
32 title('Immagine Modificata')
```

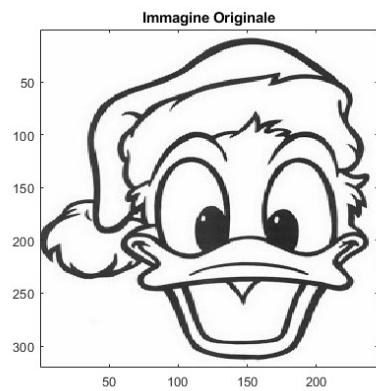


Figura 45: figure(1)

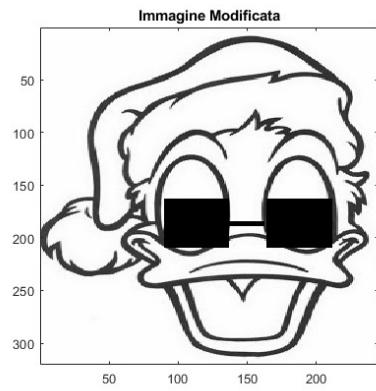


Figura 46: figure(2)

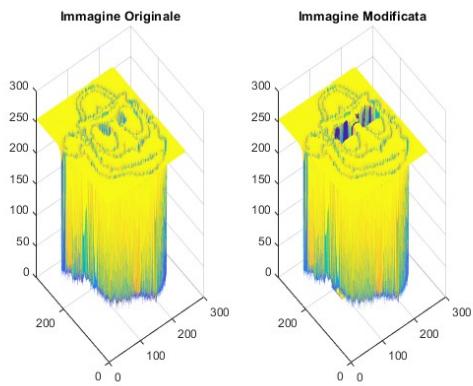


Figura 47: figure(3)

6.2.7 Esercizio 2

Realizzare una funzione che, data l'immagine a livelli di grigio `moon.tif`, conti quanti pixel (= entries i,j all'interno della matrice) assumono un particolare valore di grigio, per tutti i valori di grigio compresi tra 0 e 255. Il risultato sarà un vettore di naturali di dimensionalità (256, 1) chiamato istogramma. Provare a visualizzare questo vettore usando il comando `bar`.

Soluzione

Il codice dell'esercizio è il seguente:

```
1 % Carico immagine
2 clear all
3 close all
4 clc
5
6 I = imread('moon.tif');
7
8 % visualizzo l'immagine
9 figure(1)
10 imshow(I, []), colorbar
11 title ('Immagine a livelli di grigi')
12
13 % estraggo le dimensioni
14 [m,n] = size(I);
15
16 % versione piu' lunga: scorro l'immagine
17 % inizializzo il vettore di conteggio
18 count = zeros(256,1); %256 livelli di grigio, da 0 a 255
19 for i = 1:m
20     for j = 1:n
21         val = I(i,j);
22         count(val+1) = count(val+1) +1;
23         % devo mettere "+1" perche' in matlab
24         % gli indici dei vettori partono da 1
25     end
26 end
27 figure(2)
28 bar(count)
29 xlabel('Valori')
30 ylabel('Numero di pixels')
31
32 % VERSIONE PIU' EFFICIENTE: scorro il vettore
33 % dei livelli di grigio
34 % 256 livelli di grigio, da 0 a 255
35 vettore_count = zeros(256,1);
36 for index = 0:1:255
37     r = find(I(:)==index);
38     vettore_count(index+1,1) = length(r);
39     % devo mettere "+1" perche' in matlab
40     % gli indici dei vettori partono da 1
41 end
42
43 figure(3)
44 bar(vettore_count)
45 xlabel('Valori')
46 ylabel('Numero di pixels')
```

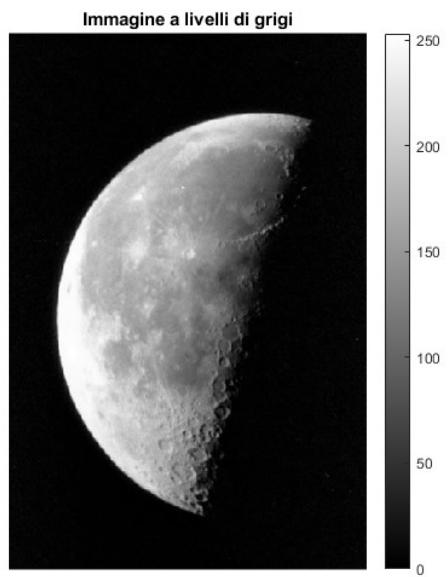


Figura 48: figure(1)

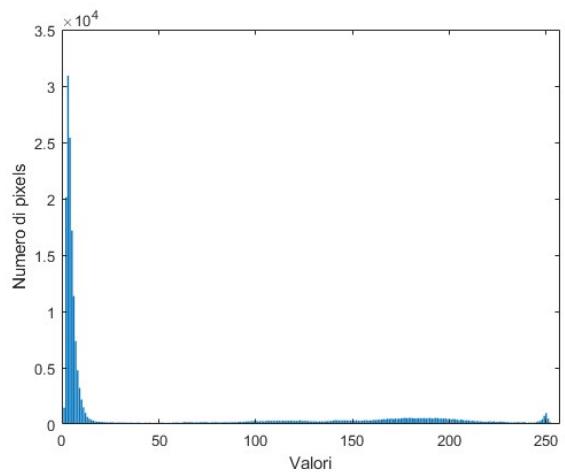
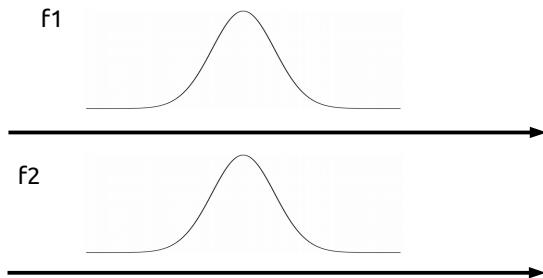


Figura 49: figure(2) e figure(3)

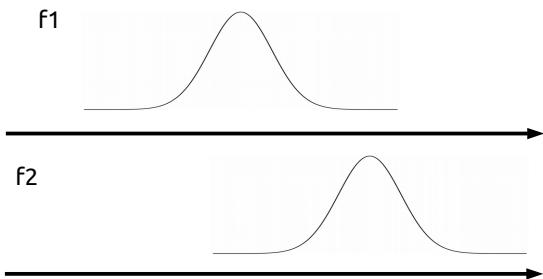
6.3 Cross Correlazione 1D

6.3.1 Cross correlazione

Molto brevemente, la **correlazione** ha l'obiettivo di **misurare se due segnali sono correlati**, cioè se si comportano nello stesso modo (per esempio se sono simili). Un **esempio** sono questi due segnali che hanno una correlazione molto alta poiché si comportano nello stesso modo:



Al contrario, nel seguente caso la correlazione è bassa poiché i due segnali, nel tempo, si comportano in modo diverso. In particolare, il picco del segnale f_2 è traslato verso destra.



Per scoprire se uno dei due segnali, soprattutto quando viene **traslato**, ha una **buona correlazione con l'altro**, viene utilizzata la **cross correlazione**. In parole povere, viene usata per capire se due segnali sono simili anche in periodi di tempo differenti. Quindi, la figura in alto, nonostante una correlazione bassa, ha una cross correlazione molto alta poiché i due segnali si comportano nello stesso modo in periodi di tempo differenti.

Si riportano qua di seguito alcune formule con i relativi paragrafi in cui è presente la spiegazione teorica:

Nome segnale	Formula
Cross correlazione 1D (segnali continui)	$f_1 \otimes f_2(t) = \int_{-\infty}^{+\infty} \tilde{f}_1(\tau) f_2(\tau - t) d\tau$
Cross correlazione 1D normalizzata	$f_1 \bar{\otimes} f_2(t) = \frac{\int_{-\infty}^{+\infty} \tilde{f}_1(\tau) f_2(\tau - t) d\tau}{\sqrt{E_{f_1} E_{f_2}}}$
Cross correlazione 1D (segnali discreti)	$x_1 \otimes x_2(n) = \sum_{k=-\infty}^{+\infty} \tilde{x}_1(k) x_2(k-n)$

6.3.2 Applicazione della cross correlazione

Si supponga di voler calcolare in MATLAB la cross correlazione tra due segnali discreti f_1 e f_2 , con rispettiva dimensione di $M = 5$ e $N = 3$.

Quindi, la cross correlazione con un **lag pari a zero** è una moltiplicazione punto a punto con somma:

$$\dots \quad 0 \quad 0 \quad 0 \quad \boxed{ } \quad 0 \quad 0 \quad 0 \quad \dots$$



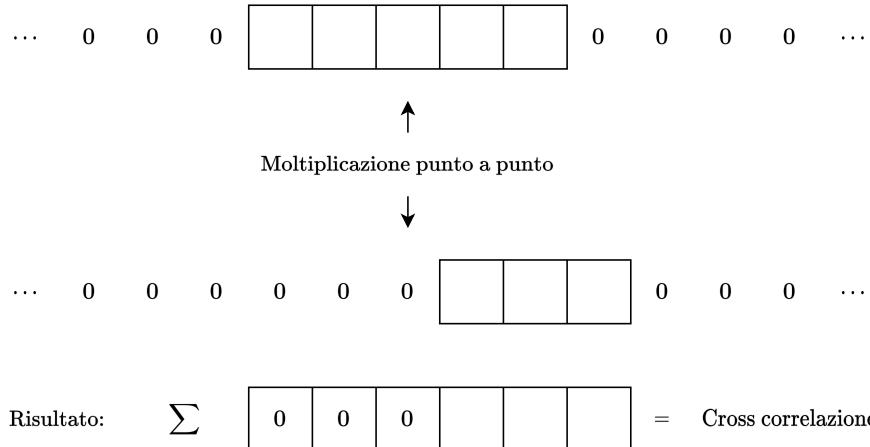
Moltiplicazione punto a punto



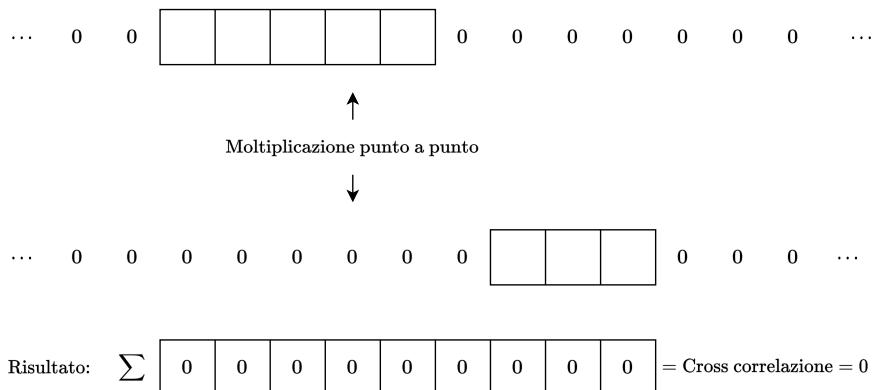
$$\dots \quad 0 \quad 0 \quad 0 \quad 0 \quad \boxed{ } \quad 0 \quad 0 \quad 0 \quad 0 \quad \dots$$

Risultato: $\sum \boxed{0 0} = \text{Cross correlazione}$

Invece, il calcolo si differenzia leggermente con la **presenza di un lag (offset)**. Viene spostato il segnale f_2 ed eseguita la moltiplicazione punto a punto con somma:

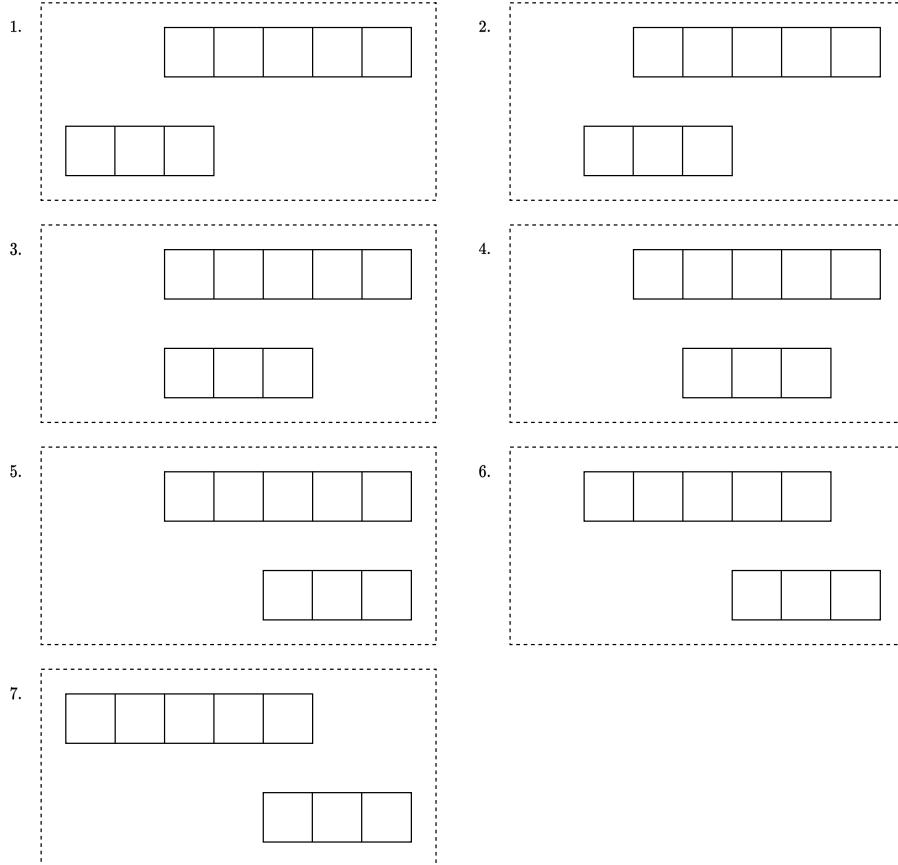


Inoltre, nonostante la cross correlazione sia definita per tutti i possibili valori di lag, in molti casi la cross **correlazione è pari a zero**, ovvero dove non c'è sovrapposizione:



6.3.3 Cross correlazione nella pratica

MATLAB mette a disposizione il comando `xcorr` per eseguire la cross correlazione. In particolare, esso calcola il vettore di cross correlazione **solo** per i lag per cui c'è sovrapposizione. Quindi, per **esempio**, i vari calcoli che esegue con due segnali discreti di dimensione $M = 5$ e $N = 3$:



In totale il vettore risultato avrà 7 posizioni: $M + N - 1 = 5 + 3 - 1 = 7$.

6.3.4 Esercizio 1

Implementare a mano la cross correlazione 1D, partendo e completando lo script presente nel file `Lezione3_EserciziPrincipali.m`.

Soluzione

Il codice dell'esercizio è il seguente:

```
1 %%  
2 %%% ESERCIZIO 1  
3 % Implementare a mano la cross correlazione 1D, partendo e  
4 % completando lo script sottostante  
5 clear all  
6 close all  
7 clc  
8  
9 f1 = [1 1 1 1 1 1 1 1]; %box  
10 f2 = [1 2 3 4 5 6 7 8]; %triangolo  
11  
12 M = length(f1);  
13 N = length(f2);  
14  
15 % NOTA: Assumiamo per questo esercizio che f1 e f2  
16 % abbiano la stessa dimensione: in caso contrario si puo' fare  
17 % zero-padding, come spiegato qui di seguito:  
18 %  
19 % Zero padding: operazione per rendere uguali le dimensioni dei  
20 % vettori (segnali) in ingresso  
21 % if N>M  
22 %   % concatena due vettori: cat(DIM,A,B)  
23 %   f1 = cat(2,f1,zeros(1,N-M));  
24 %   M=N;  
25 % elseif N<M  
26 %   f2 = cat(2,f2,zeros(1,M-N));  
27 % end  
28 %  
29  
30  
31 figure; set(gcf,'name','Cross Correlazione','IntegerHandle','off');  
32 subplot(511); stem(f1); title('f1')  
33 subplot(512); stem(f2); title('f2')  
34  
35 % Primo confronto: un bin di sovrapposizione  
36 % si aggiungono zeri a destra e a sinistra  
37 tf1 = [zeros(1,N-1),f1,zeros(1,N-1)];  
38 tf2 = [f2,zeros(1,2*N-2)];  
39  
40 lag = [-N+1:N-1];  
41 MYf1xf2 = [];  
42 for i=1:2*N-1  
43     subplot(513); stem(tf1); title('f1 allineato')  
44     subplot(514); stem(tf2); title('f2 allineato')  
45     % calcolare il valore della cross correlazione  
46     MYf1xf2 = [MYf1xf2 sum(tf1.*tf2)];  
47     % spostare f2 verso destra aggiungendo uno zero davanti e  
48     % rimuovendolo in fondo  
49     tf2(end) = [];  
50     tf2 = [0 tf2];  
51     % Oppure si ricostruisce da zero  
52     % tf2 = [zeros(1,i) f2 zeros(1,2*M-2-i)];  
53     % Ultima possibilita': si puo' fare in modo compatto
```

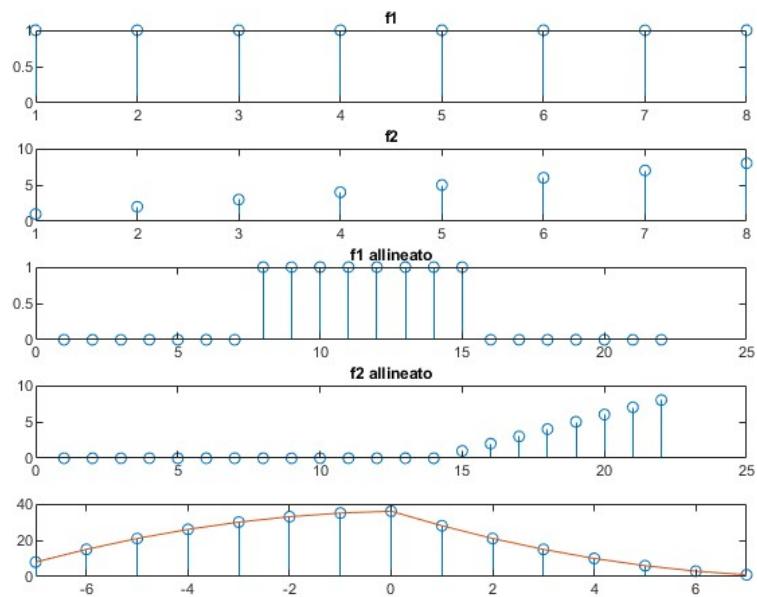
```

54 % con circshift
55 % tf2 = circshift(tf2,1,2);
56
57 % visualizzare il vettore di crosscorrelazione
58 % calcolato fino adesso
59 subplot(515); stem(lag(1:i),MYf1xf2); xlim([-N+1 N-1]);
60
61 % il programma si ferma per un secondo
62 pause(1);
63 end
64
65 hold on; subplot(515); plot(lag(1:i),MYf1xf2);

```

- (9-28) Si assume che i due segnali siano della stessa dimensione, in caso contrario viene eseguito lo *zero-padding*, ovvero vengono aggiunti una serie di zeri.
- (31-33) Creazione della prima finestra di grafici con organizzazione dei titoli e delle misure.
- (35-38) Vengono preparati gli array `tf1` e `tf2`. Il primo è il segnale che rimane fermo, mentre il secondo è il segnale in movimento che piano piano (ad ogni ciclo) si sposta verso destra aggiungendo zeri a sinistra.
- (40-63) Il ciclo `for` continua ad andare finché non è stato ripetuto $2 * N - 1$ volte. Ad ogni iterazione, viene salvato nell'array `MYf1xf2` il risultato della cross correlazione e viene spostato il secondo segnale eliminando l'ultimo elemento e aggiungendo uno zero all'inizio. Infine vengono aggiornate le figure.

Il risultato è il seguente:



6.3.5 Esercizio 2

Cross-correlazione su segnali audio: riconoscimento del suono attraverso la cross-correlazione:

- Caricare i primi 20 secondi dei segnali audio “funky.mp3”, “lost.mp3”, “Diana.mp3”, “never.mp3”, “T69.mp3”;
- Caricare il segnale audio “Test.wav”;
- Confrontate l'esempio di test con le varie canzoni della galleria usando la cross correlazione: da quale canzone proviene?

Suggerimento: cercare il segnale che contiene la cross correlazione più grande (si parla dello script presente nel file “Lezione3_EserciziPrincipali.m”).

Soluzione

Il codice dell'esercizio è il seguente:

```
1 %%  
2 %%% ESERCIZIO 2  
3 % Cross-correlazione su segnali audio: riconoscimento  
4 % del suono attraverso la cross-correlazione.  
5 %  
6 % - Caricare i primi 20 secondi dei segnali audio  
7 %   'funky.mp3', 'lost.mp3', 'Diana.mp3',  
8 %   'never.mp3', 'T69.mp3'  
9 % - Caricare il segnale audio 'Test.wav'  
10 % - Confrontate l'esempio di test con le varie canzoni della  
11 %   galleria usando la cross correlazione:  
12 %   da quale canzone proviene?  
13 % - Suggerimento: cercare il segnale che contiene la  
14 %   crosscorrelazione piu' grande  
15  
16 clear all  
17 close all  
18 clc  
19  
20 [Y1,fs1] = audioread('funky.mp3',[1,96000*20]);  
21 [Y2,fs2] = audioread('lost.mp3',[1,96000*20]);  
22 [Y3,fs3] = audioread('Diana.mp3',[1,96000*20]);  
23 [Y4,fs4] = audioread('never.mp3',[1,96000*20]);  
24 [Y5,fs5] = audioread('T69.mp3',[1,96000*20]);  
25  
26 test = audioread('Test.wav');  
27  
28 figure; set(gcf, 'name', 'Dataset canzoni', 'IntegerHandle', 'off');  
29 subplot(2,3,1); plot(Y1(1:96000*3,1));  
30 subplot(2,3,2); plot(Y2(1:96000*3,1));  
31 subplot(2,3,3); plot(Y3(1:96000*3,1));  
32 subplot(2,3,4); plot(Y4(1:96000*3,1));  
33 subplot(2,3,5); plot(Y5(1:96000*3,1));  
34 subplot(2,3,6); plot(test(1:96000*3,1));  
35  
36 % Nota: segnale audio ha due canali (stereo),  
37 % per la cross correlazione consideriamo solo il primo  
38 % Array di celle: un metodo piu' veloce per raccogliere sequenze  
39 % di lunghezza diversa.  
40 gallery{1}=Y1(:,1);  
41 gallery{2}=Y2(:,1);
```

```

42 gallery{3}=Y3(:,1);
43 gallery{4}=Y4(:,1);
44 gallery{5}=Y5(:,1);
45
46 maxcc = zeros(5,1);
47 for g=1:5
48     [xc{g},lagc{g}]= xcorr(gallery{g},test(:,1));
49     maxcc(g) = max(abs(xc{g}));
50 end
51
52 figure; set(gcf,'name','Risultati di matching', ...
53     'IntegerHandle','off');
54 for g=1:5
55     subplot(2,3,g); plot(lagc{g},xc{g});
56 end
57
58 % trovo il segnale con la miglior cross correlazione
59 [maxcorr,bestmatch]=max(maxcc);
60
61 % trovo il lag della massima correlazione nel segnale
62 % col match migliore
63 [~,maxli]=max(xc{bestmatch});
64
65 % visualizzo il segnale originale
66 figure
67 plot(test(:,1),'-r','LineWidth',2);
68 hold on
69 % visualizzo il best match
70 plot(gallery{bestmatch} ...
71     (lagc{bestmatch}(maxli)+1:lagc{bestmatch}(maxli)+480001,1), ...
72     '-k','LineWidth',1);

```

- (20-34) Viene rappresentato graficamente il primo risultato che non è altro che una serie di grafici rappresentanti le frequenze delle canzoni caricate
- (36-44) Dato che l'audio ha due canali stereo, viene eseguita la cross correlazione solo sul primo segnale. I dati vengono salvati all'interno dell'array `gallery`. Tramite il costrutto `{value}`, si indica che la cella in posizione `value` deve contenere il valore assegnatole.
- (46-50) Viene eseguita la cross correlazione e in `xc` vengono salvati i calcoli effettuati, mentre in `lagc` viene salvato il lag di ogni cross correlazione effettuata. Infatti, le dimensioni dei vettori presenti in ogni cella di `xc` e `lagc` sono uguali.
Inoltre, ogni volta viene eseguito il controllo della cross correlazione effettuata e viene acquisito il valore assoluto massimo.
- (52-56) Viene creato il grafico per rappresentare i segnali.
- (58-72) Viene preso il massimo assoluto tra tutti i massimi trovati nella cross correlazione, insieme al suo indice ovviamente;
Viene effettuato l'accesso alla lista dei valori di lag e preso il lag relativo al massimo assoluto;
Viene visualizzato infine il segnale in un grafico.

Il risultato è il seguente:

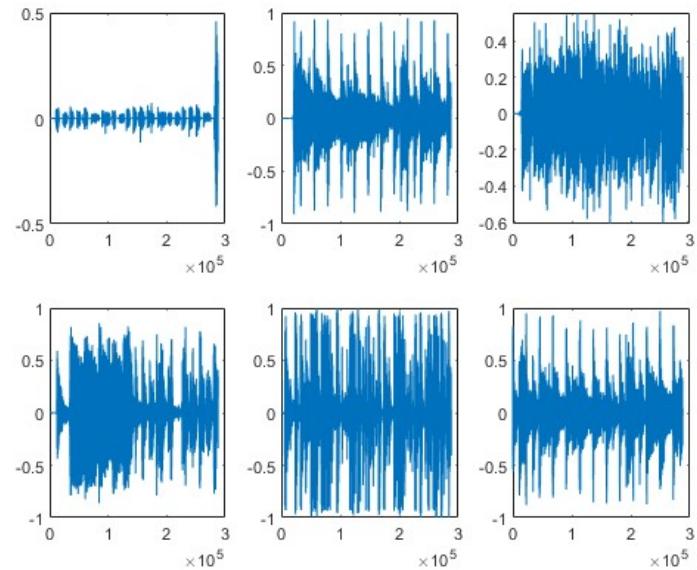


Figura 50: Dataset canzoni.

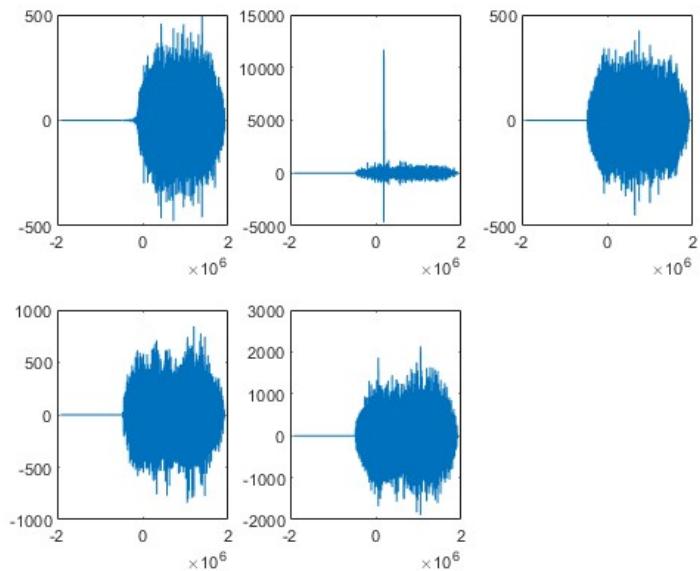


Figura 51: Risultati di matching.

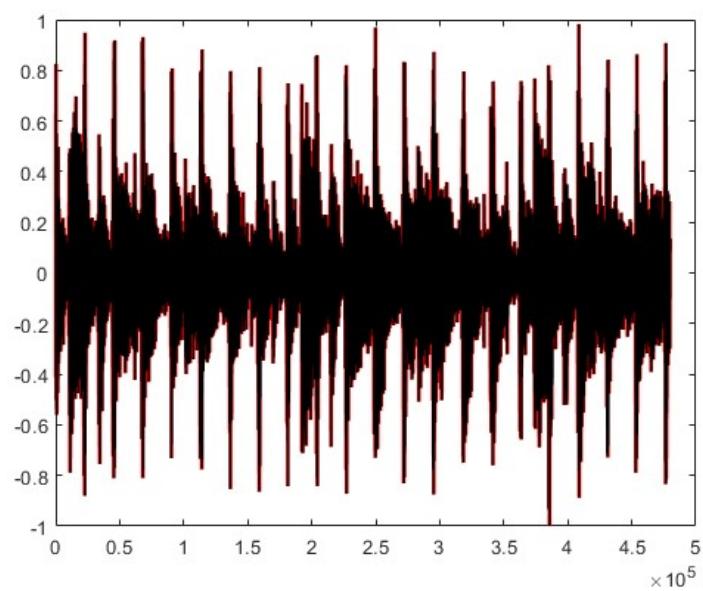


Figura 52: Segnale originale.

6.4 Cross correlazione 2D

6.4.1 Estensione della cross correlazione 1D

A differenza della cross correlazione 1D, la **cross correlazione 2D** presenta due valori di lag:

- Lag di **riga** (spostamento nelle righe)
- Lag di **colonna** (spostamento nelle colonne)

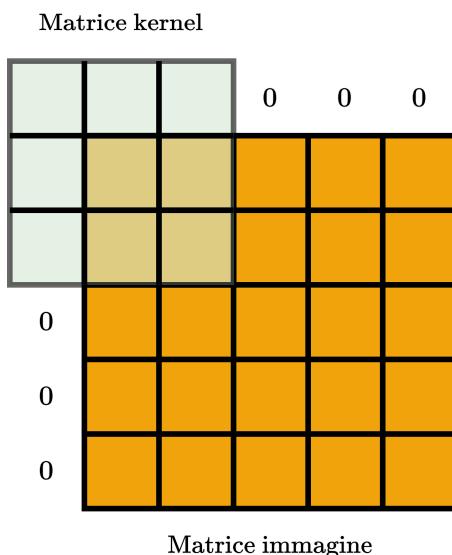
Date le due dimensioni, la cross correlazione riguarda quindi le matrici e non più singoli vettori. Tipicamente la **matrice con dimensione più piccola** viene chiamata **kernel**, mentre l'altra viene chiamata **immagine**. La sua formula è la seguente:

$$x_1 \otimes x_2 (m, n) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} x_1(u, v) x_2(u - m, v - n)$$

La cross correlazione 2D viene **calcolata** spostando la matrice kernel di un offset di riga m e di un offset di colonna n ; viene effettuato zero padding; infine, viene eseguita la moltiplicazione punto a punto con somma. In questo modo, viene rappresentato il valore di cross-correlazione 2D per il valore (m, n) .

6.4.2 Applicazione della cross correlazione 2D

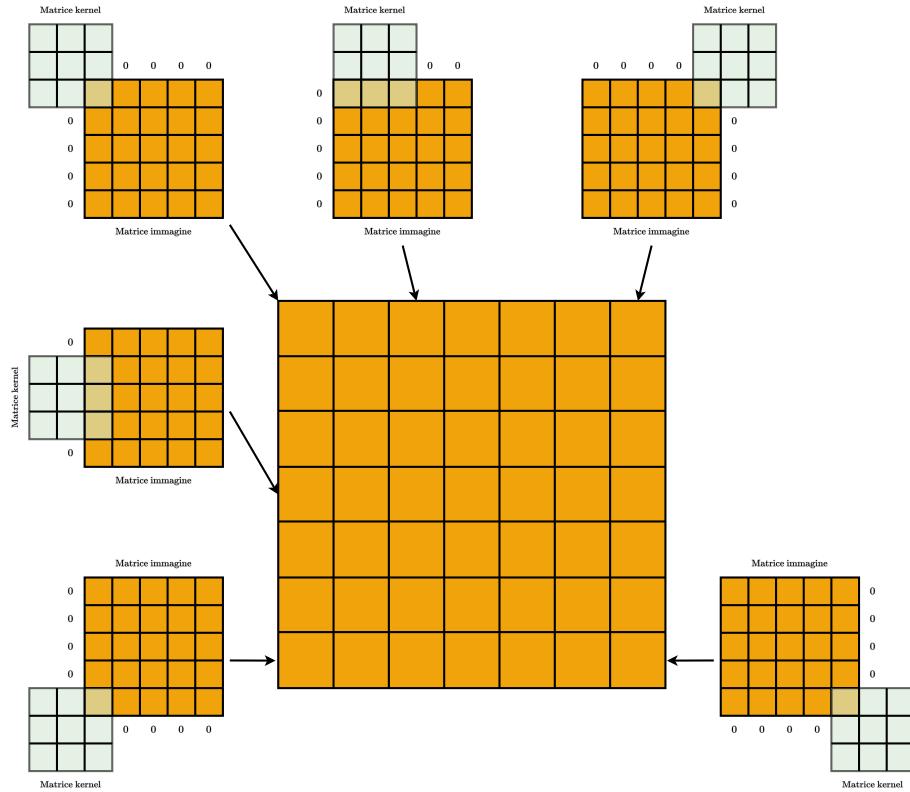
I passaggi da eseguire sono i seguenti. Si sposta la matrice kernel, si esegue lo zero padding e infine si esegue la moltiplicazione punto a punto con somma. Così si ottiene la cross correlazione per un determinato lag:



La matrice di correlazione risultante ha dimensione:

$$(R1 + R2 - 1) \times (C1 + C2 - 1)$$

In cui $R1, C1$ sono le **dimensioni della matrice immagine** e $R2, C2$ sono le **dimensioni del kernel**.



6.4.3 Ottimizzazione della cross correlazione 2D

Esiste una versione ottimizzata della cross correlazione 2D. Innanzitutto viene effettuato lo zero padding della matrice **immagine**. Quindi, vengono aggiunte $R2 - 1$ colonne di zeri a destra e a sinistra; vengono aggiunte $C2 - 1$ righe di zeri sopra e sotto:

$$\begin{array}{cccccccccc}
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & 0 & 0 \\
 0 & 0 & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & 0 & 0 \\
 0 & 0 & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & 0 & 0 \\
 0 & 0 & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & 0 & 0 \\
 0 & 0 & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & 0 & 0 \\
 0 & 0 & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & \boxed{\text{yellow}} & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array}$$

Per ogni valore di lag, viene posizionata la matrice di kernel in quella determinata posizione; viene estratta la corrispondente parte della matrice immagine tramite lo zero padding; infine, viene applicata la moltiplicazione punto a punto con somma:

$$\begin{array}{ccc}
 \text{Matrice kernel} & & \text{Matrice kernel} \\
 \begin{array}{ccccccccc}
 0 & 0 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array} & \times & \begin{array}{ccccc}
 \boxed{0} & \boxed{0} & \boxed{0} \\
 \boxed{0} & \boxed{0} & \boxed{0} \\
 \boxed{0} & \boxed{0} & \boxed{0}
 \end{array} \\
 & & \sum
 \end{array}$$

Ripetendo questa operazione per tutti i possibili lag, viene ottenuta la matrice di cross correlazione.

6.4.4 Esercizio 1

Usare la cross correlazione 2D per trovare la posizione del *template* nell'immagine. In particolare, si richiede di calcolare di quanto (righe-colonne) il *template* è stato traslato rispetto all'angolo in alto a sinistra dell'immagine.

Suggerimento: calcolare la cross correlazione (`xcorr2`) tra l'immagine e il template (kernel); estrarre le coordinate del massimo; recuperare la posizione del kernel. Controllare anche l'help della funzione `xcorr2`.

Soluzione

Il codice dell'esercizio è il seguente:

```
1 %% Esercizio 1
2 % Matching 2D
3 % Usare la cross correlazione 2D per trovare la posizione
4 % del template nell'immagine
5 % In particolare si richiede di calcolare di quanto
6 % (righe-colonne) il template e' stato traslato rispetto
7 % all'angolo in alto a sinistra dell'immagine
8 %
9 % Suggerimento: calcolare la cross correlazione (xcorr2)
10 % e il template (kernel), estrarre le coordinate del massimo e
11 % recuperare la posizione del kernel
12 %
13 % Controllare anche l'help della funzione xcorr2
14
15 clear all
16 close all
17
18 % Template: una croce
19 template = 0.2*ones(55);
20 template(29:31,15:45) = 0.6;
21 template(15:45,29:31) = 0.6;
22
23
24 % Immagine: si posiziona il template con un offset
25 immagine = 0.2*ones(111);
26 offset = [10 40];
27 immagine(offset(1):offset(1)+size(template,1)-1, ...
28         offset(2):offset(2)+size(template,2)-1) = template;
29
30 figure
31 imshow(template,[])
32 title('Template')
33
34 figure
35 imshow(immagine,[])
36 title('Immagine')
37
38
39 cc = xcorr2(immagine,template);
40 figure, imagesc(cc)
41
42
43 % si trovano le coordinate del massimo
44 maxcc = max(cc(:)); % massimo
45 [r,c] = find(cc == maxcc); % coordinate del massimo
46
```

```

47
48 % la posizione 1,1 del kernel si trova
49 % Riga: r-R2+1
50 % Colonna: c-C2+1
51
52 [R2,C2] = size(template);
53 corr_offset = [r-R2+1, c-C2+1];
54
55 fprintf('Offset originale: %d-%d\n',offset)
56 fprintf('Offset calcolato con la cross-correlazione: %d-%d\n', ...
57     corr_offset)

```

- (18-36) Vengono create due immagini, la prima (template) che rappresenta una croce bianca centrata su sfondo nero e la seconda (immagine) che rappresenta una croce bianca in alto a destra su sfondo nero.
- (39-40) La cross correlazione 2D è molto semplice. Il comando `xcorr2` restituisce una matrice con i valori della cross correlazione appena effettuata. Successivamente, il tutto viene rappresentato in una figura.
- (43-57) Viene ottenuto il valore massimo all'interno della matrice della cross correlazione e successivamente vengono ottenute le relative coordinate.

Viene calcolata la dimensione (righe × colonne) della figura template, ovvero della croce bianca centrata. Queste dimensioni sono fondamentali per calcolare l'offset calcolato con la cross correlazione 2D. In particolare, calcolando $r - R2 + 1$, cioè riga del valore massimo della cross correlazione meno il numero di righe della figura template più uno, viene ottenuto l'offset applicato durante la cross correlazione 2D sulle righe. Lo stesso ragionamento vale per le colonne.

Il risultato:



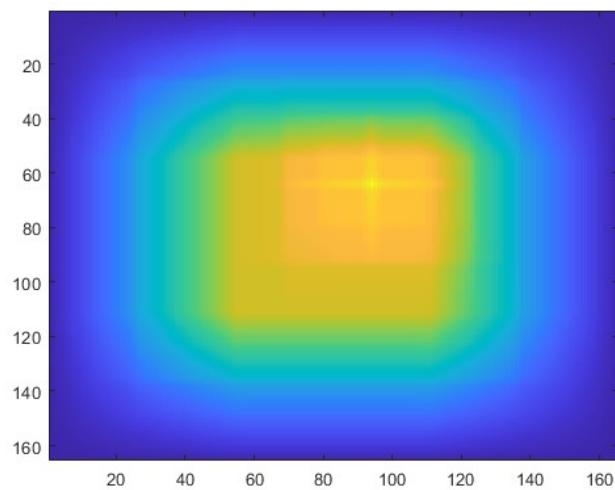


Figura 53: Figura della cross correlazione 2D con colori scalati.

6.4.5 Esercizio 2

Calcolare manualmente al cross correlazione 2D tra le matrici X_1 e X_2 definite nel file “Lezione4_EserciziPrincipali.m”. Confrontare con il risultato del comando matlab `xcorr2(X1, X2)`. Usare la versione ottimizzata descritta nelle diapositive precedenti.

Soluzione

Il codice dell'esercizio è il seguente:

```
1 %% Esercizio 2
2 % Implementazione manuale della cross-correlazione 2D
3 % Calcolare manualmente la cross correlazione 2D tra le
4 % matrici X1 e X2. Confrontare con il risultato del
5 % comando matlab xcorr2(X1,X2)
6 %
7 % Suggerimento: usare la versione piu' ottimizzata:
8 % si fa zero padding della matrice immagine (la matrice piu'
9 % grande) e si fa scorrere il kernel
10 %

11
12 clear all
13 close all
14
15 X1 =[1      2      5      2      5;
16      3      4      5      4      2;
17      5      2      3      2      2;
18      2      3      2      4      2;
19      3      4      2      2      3];
20
21
22 X2 =[3      3      2;
23      2      3      4;
24      4      5      4];
25
26
27 [R1, C1] = size(X1);
28 [R2, C2] = size(X2);
29
30 % Zero padding: aggiungere (R2-1) colonne di zeri a dx e a sx e
31 % (C2-1) righe di zeri sopra e sotto
32
33 % In due passi: prima si aggiungono le colonne
34 paddedX1 = [zeros(R1,C2-1) X1 zeros(R1,C2-1)];
35 % In pratica:
36 % paddedX1 = [zeros(5,2) X1 zeros(5,2)];
37
38 % ora le righe
39 paddedX1 = [zeros(R2-1,C1+2*(C2-1)); paddedX1; ...
40             zeros(R2-1,C1+2*(C2-1))];
41 % In pratica:
42 % paddedX1 = [zeros(2,5+4); paddedX1; zeros(2,5+4)];
43
44
45 % si crea la matrice di cross-correlazione
46 % dimensione 7x7: (5+3-1)x(5+3-1)
47 MyCrossCorr = zeros(R1+R2-1,C1+C2-1);
48
49 [r,c] = size(MyCrossCorr); % valori da calcolare
50
```

```

51 % doppio ciclo for
52 for i = 1:r
53     for j = 1:c
54         % si estraе la parte di paddedX1
55         extractedX1 = paddedX1(i:i+R2-1,j:j+C2-1);
56         % si moltiplica punto a punto per il kernel
57         tmp = extractedX1.*X2;
58         % si somma e si memorizza in posizione i,j
59         MyCrossCorr(i,j) = sum(tmp,'all');
60     end
61 end
62
63 % confronto con il risultato
64 MatlabCrossCorr = xcorr2(X1,X2);
65
66 % check
67 sum(abs(MyCrossCorr - MatlabCrossCorr), 'all')

```

- (15-28) Creazione delle due matrici (segnali) e ottenimento delle rispettive dimensioni.
- (30-42) La versione ottimizzata prevede l'implementazione dello zero padding come passo iniziale. Quindi, vengono aggiunte le colonne semplicemente creando una nuova matrice affiancando degli zeri; vengono aggiunte le righe nello stesso modo ma inserendo dei punti e virgola così da andare a capo.
- (45-61) Viene inizializzata la matrice di cross correlazione con numero di righe pari alla somma delle righe dei due segnali meno uno e analogamente per le colonne lo stesso calcolo.

La cross correlazione 2D viene implementata con un doppio ciclo for, il primo utile per scorrere le righe, il secondo utile per scorrere le colonne. Ad ogni iterazione del ciclo, viene estratta un pezzo di matrice dalla matrice nella quale è stata fatta lo zero padding. La dimensione da estrarre è calcolata:

- Righe: dalla riga attuale in cui è il ciclo, fino alla riga attuale in cui è il ciclo più il numero di righe della matrice kernel meno uno;
- Colonne: dalla colonna attuale in cui è il ciclo, fino alla colonna attuale in cui è il ciclo più il numero di colonne della matrice kernel meno uno.

Una volta estratti i valori, viene eseguita la moltiplicazione punto a punto tra i valori estratti e la matrice kernel.

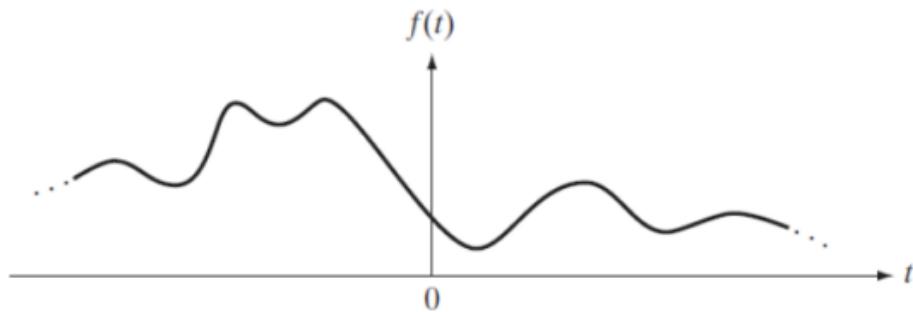
La cross correlazione 2D si conclude sommando tra di loro tutti i valori trovati durante la moltiplicazione. Così facendo si trova un valore che viene inserito all'interno della matrice della cross correlazione 2D.

- (63-67) Il codice si conclude eseguendo la cross correlazione 2D con il comando di MATLAB. Nel caso in cui il risultato tra le due matrici (matrice calcolata con la cross correlazione manuale e matrice calcolata con la cross correlazione automatica) sia uguale a zero, allora il metodo ha funzionato.

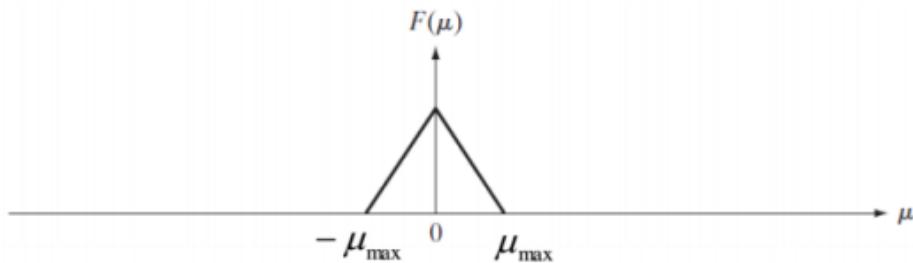
6.5 Trasformata di Fourier Discreta 1D

6.5.1 Spiegazione teorica sulla TdF discreta nei calcolatori

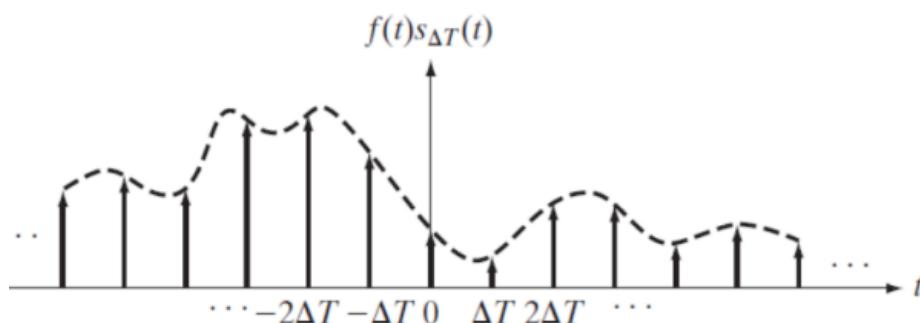
Con la trasformata di Fourier è possibile descrivere un segnale sia nel dominio del tempo che nel dominio delle frequenze. Ricordando che un segnale nel dominio del tempo:



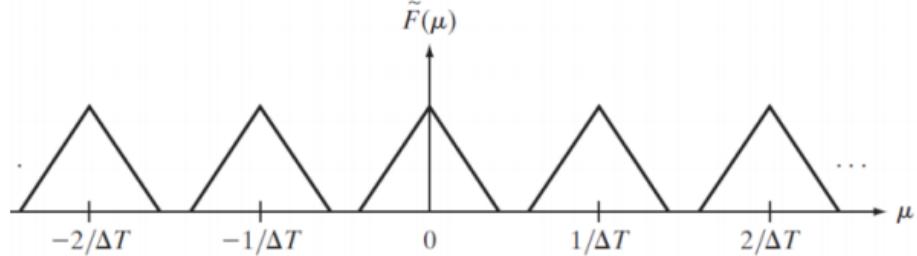
Mentre nel dominio delle frequenze:



Per convertire un segnale dal dominio del tempo al dominio delle frequenze, è necessario utilizzare prima il campionamento e successivamente la trasformata di Fourier discreta. Il campionamento di un segnale originale viene eseguito prendendo un punto ΔT ogni f_s , cioè ogni frequenza di campionamento $\frac{1}{\Delta T}$:

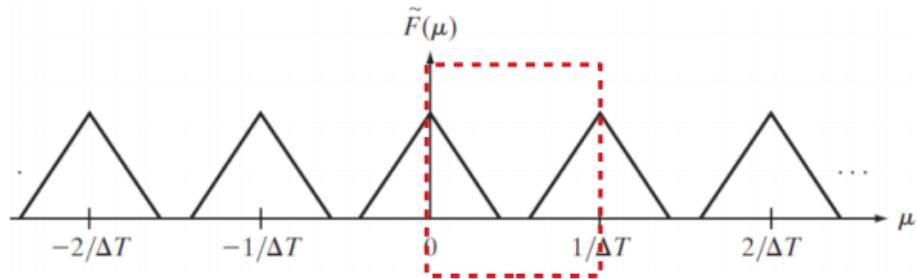


Successivamente, viene eseguita la trasformata di Fourier discreta, ovvero viene replicato lo spettro di frequenza ogni f_s :

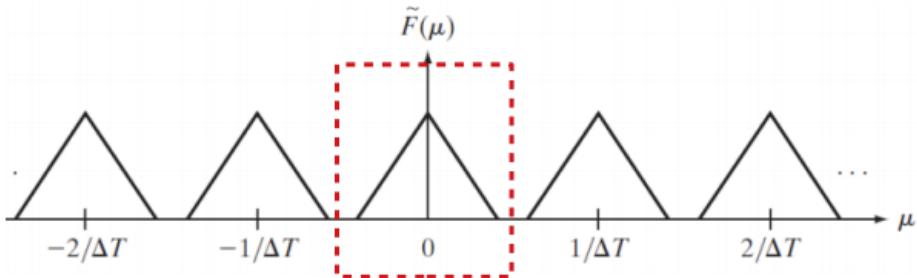


In MATLAB, vi sono due comandi principali per ottenere un pezzo della Trasformata di Fourier (un pezzo perché in teoria la Trasformata continua all'infinito!):

- `fft` ritorna il pezzo indicato in figura. Se il segnale originale ha N punti, allora si hanno N bin in frequenza, ovvero da 0 a f_s con salti di $\frac{f_s}{N}$:



- `fftshift` è possibile ottenere N bins in un intervallo da $-\frac{f_s}{2}$ a $+\frac{f_s}{2}$ con salti di $\frac{f_s}{N}$:



6.5.2 Aliasing

Il fenomeno di **aliasing** si manifesta quando non viene soddisfatto il teorema del campionamento, ovvero quando la frequenza di campionamento non è maggiore alla frequenza massima del segnale per due. Il fenomeno dell'aliasing non consente di ricostruire il segnale.

6.5.3 Esempi di TdF discreta 1D

Nel seguente esempio viene definita una sinusoida a 20 Hz, viene campionato un segnale a 100 Hz e infine, viene osservato lo spettro di magnitudo con picco a 20 Hz:

```
1 %% Esempio 1 - SINUSOIDA A 20 Hz 1-D CAMPIONATA A 100 Hz
2 % Obiettivo di questo esempio e' mostrare come si possono
3 % generare segnali e controllarne lo spettro. In particolare,
4 % si definisce una sinusoida a 20 Hz, si campiona a 100Hz e
5 % si osserva lo spettro di magnitudo con picco a 20 Hz.
6 % Controllare anche lo spettro di fase.

7
8
9 clear all
10 close all
11 clc
12
13 % FUNZIONI RICHIESTE: fft, fftshift, abs, angle
14
15 % Definisco le principali variabili di interesse e il segnale
16 mu = 20;          % frequenza del segnale sinusoidale
17 mu_s = 100;        % frequenza di campionamento
18 Dt = 1/mu_s;       % delta T visto a lezione
19 t = 0:Dt:1-Dt;    % prendo un secondo di durata
20 N = length(t);    % numero di campioni
21
22 f = sin(2*pi*mu*t); % segnale sinusoidale
23
24 figure
25 subplot(221)
26 plot(t,f,'-b.', 'MarkerSize',9)
27 xlabel('tempo (sec.)')
28 ylabel('f(t)')
29 title('Segnale campionato');

30
31 % Calcolo la trasformata di Fourier e
32 % definisco il vettore delle frequenze
33 % Fast Fourier Transform e' l'implementazione della DFT
34 F = fft(f);
35 % Se il segnale originale ha N punti ho N bin in frequenza:
36 % da 0 alla frequenza di campionamento (mu_s) con step mu_s/N
37 mu_sampling = mu_s/N; % passo
38 % campioni nello spazio delle frequenze
39 mu = 0:mu_sampling:mu_s-mu_sampling;
40
41 subplot(222)
42 stem(mu,abs(F));
43 xlabel('frequenza (Hz)')
44 ylabel('|F|')
45 grid
46 title('DFT (abs) senza riordinamento');

47
48 % Esegue operazione di centratura dello spettro e
49 % visualizzo spettro centrato
50 Fs = fftshift(F); % centratura
51 mu_max = mu_s/2; % frequenza di Nyquist ?
52 % nuovo vettore frequenze
53 mu = -mu_max:mu_sampling:mu_max-mu_sampling;
54
55 subplot(223)
56 stem(mu,abs(Fs))
57 xlabel('frequenza (Hz)')
```

```

58 ylabel ('|Fs|')
59 grid
60 title('DFT (abs) con riordinamento');
61
62 % Eseguo pulizia per eliminare le componenti
63 % con magnitudo bassa e calcolo la fase
64 th = 1e-6;
65 Fs(abs(Fs) < th) = 0;
66
67 subplot(224)
68 stem(mu,angle(Fs)/pi) % Posso moltiplicare per 180 per avere gradi
69 xlabel ('frequenza (Hz)')
70 ylabel ('fase/\pi')
71 grid
72 title('DFT (fase) con riordinamento');
73
74
75 %%%%%% RIASSUNTO:
76 % Passi:
77 % calcolo il vettore dei tempi e delle frequenze
78 % (N valori da -mu_s/2 a mu_s/2)
79 % calcolo la fft e la fft riordinata
80 % calcolo il vettore frequenze:
81 %         fft riordinata: (N valori da -mu_s/2 a mu_s/2-step)
82 %         fft non riordinata: (N valori da 0 a mu_s-step)
83 %         step = mu_s/N
84 % visualizzo lo spettro di ampiezza
85 % Se segnale rumoroso: pulisco lo spettro e
86 % tolgo le frequenze troppo basse
87 % visualizzo lo spettro di fase
88 %%%%%%%%%%%%%%
89
90
91 %%%%%%
92 % Altro esempio con il seguente segnale:
93 % f = cos(2*pi*10*t) - sin(2*pi*40*t)
94 % cosa ottengo?
95
96 f = cos(2*pi*10*t) - sin(2*pi*40*t); % segnale sinusoidale
97
98 figure
99 subplot(221)
100 plot(t,f,'-b.', 'MarkerSize',9)
101 xlabel ('tempo (sec.)')
102 ylabel ('f(t)')
103 title('Segnale campionato');
104
105 % Calcolo la trasformata di Fourier e
106 % definisco il vettore delle frequenze
107 % Fast Fourier Transform e' l'implementazione della DFT
108 F = fft(f);
109 mu_sampling = mu_s/N; % passo
110 % campioni nello spazio delle frequenze
111 mu = 0:mu_sampling:mu_s-mu_sampling;
112
113 subplot(222)
114 stem(mu,abs(F));
115 xlabel ('frequenza (Hz)')
116 ylabel ('|F|')
117 grid
118 title('DFT (abs) senza riordinamento');
119

```

```

120 % Eseguo operazione di centratura dello spettro e
121 % visualizzo spettro centrato
122 Fs = fftshift(F); % centratura
123 mu_max = mu_s/2; % frequenza di Nyquist ?
124 % nuovo vettore frequenze
125 mu = -mu_max:mu_sampling:mu_max-mu_sampling;
126
127 subplot(223)
128 stem(mu,abs(Fs))
129 xlabel ('frequenza (Hz)')
130 ylabel ('|Fs|')
131 grid
132 title('DFT (abs) con riordinamento');
133
134 % Eseguo pulizia per eliminare le componenti
135 % con magnitudo bassa e calcolo la fase
136 th = 1e-6;
137 Fs(abs(Fs) < th) = 0;
138
139 subplot(224)
140 stem(mu,angle(Fs)/pi) % Posso moltiplicare per 180 per avere gradi
141 xlabel ('frequenza (Hz)')
142 ylabel ('fase/\pi')
143 grid
144 title('DFT (fase) con riordinamento');

```

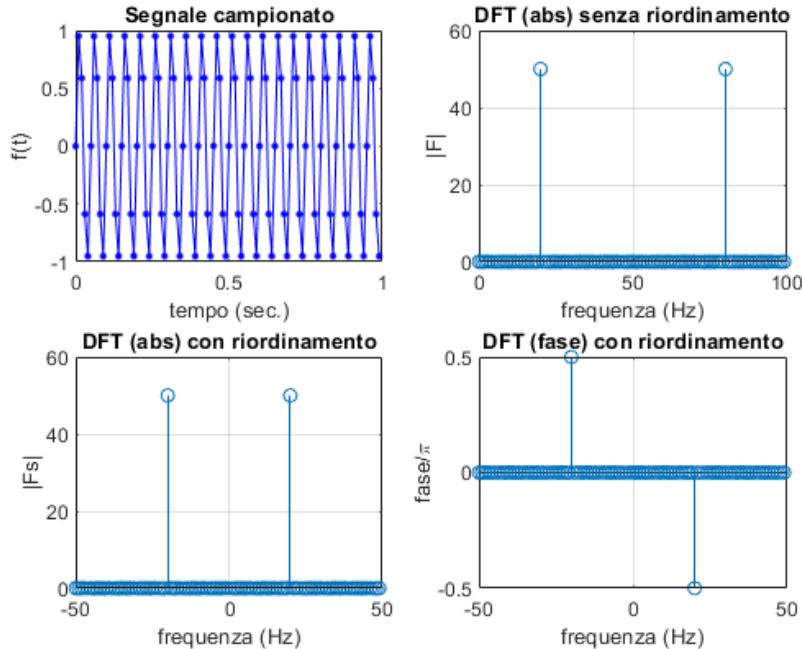


Figura 54: Operazioni eseguite sul segnale $\sin(2\pi\mu t)$ (riga 22), ovvero $\sin(2\pi\mu t)$.

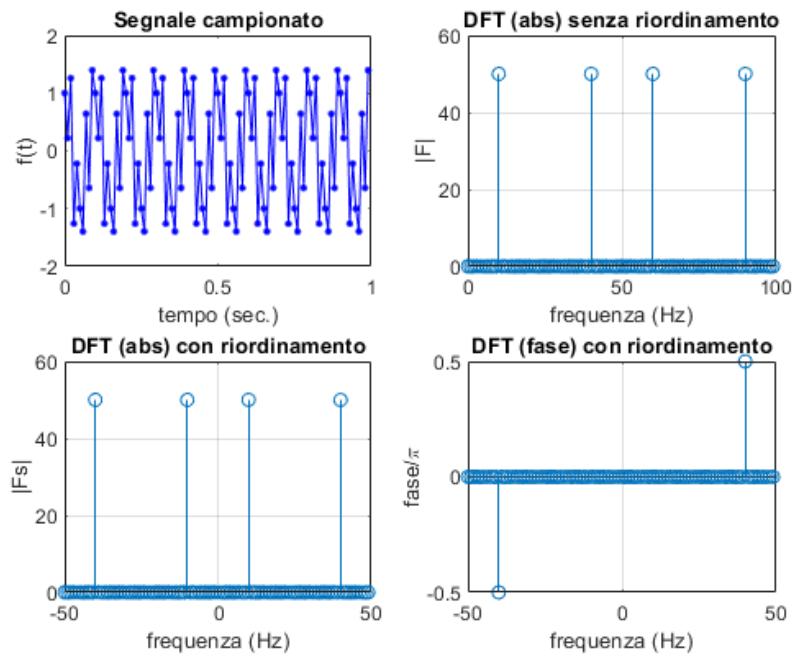
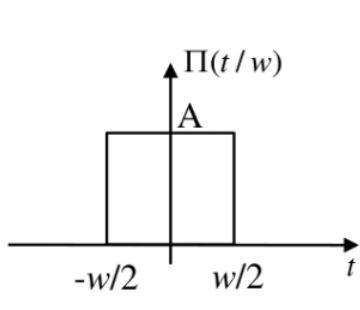


Figura 55: Operazioni eseguite sul segnale $\cos(2\pi 10t) - \sin(2\pi 40t)$ (riga 96), ovvero $\cos(2\pi 10t) - \sin(2\pi 40t)$.

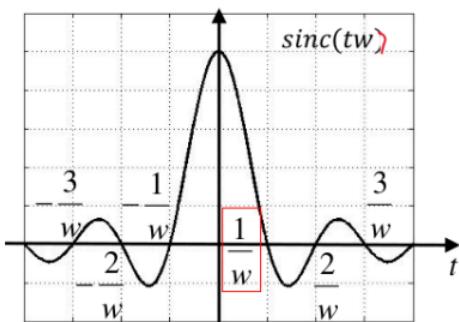
6.5.4 Esercizio 1

Analizzare tramite la Trasformata di Fourier Discreta un segnale BOX:

- Creare un'onda rettangolare di 1 secondo con una frequenza di campionamento di 500 Hz e una lunghezza di 0.2 secondi (funzione `rectpuls`).
- Calcolare la TdF discreta e visualizzarne lo spettro di ampiezza e di fase (con riordinamento).
- Controllare che il risultato ottenuto per lo spettro di ampiezza corrisponda a quanto spiegato in teoria.



Funzione Box



Spettro di ampiezza

Soluzione

Il codice della soluzione è il seguente:

```

1 %% Esercizio 1
2 % Analizzare tramite la Trasformata di Fourier Discreta
3 % un segnale BOX:
4 % - Creare un'onda rettangolare di 1 secondo con una frequenza di
5 % campionamento di 500 Hz e una lunghezza di 0.2 s
6 % (funzione rectpuls - si veda l'help)
7 % - Calcolare la DFT e visualizzarne lo spettro di
8 % ampiezza e di fase (con riordinamento)
9 % - Controllare che il risultato ottenuto per lo spettro
10 % di ampiezza corrisponda a quanto spiegato in teoria
11 %
12 % (se si vuole): Provare ad effettuare la stessa analisi
13 % su un segnale audio registrato direttamente in MATLAB,
14 % della durata di 4 secondi
15
16 % Funzione BoX
17
18 clear all
19 close all
20
21 mu_s = 500; % Frequenza di campionamento
22 T = 0.2;      % lunghezza del rect in secondi
23
24 % Vettore dei tempi per il segnale rect
25 time = -0.5-1/mu_s:1/mu_s:0.5;
26 x = rectpuls(time,T); % Genero l'onda rettangolare

```

```

27 % Guardare il doc di rectpuls
28
29 figure(1)
30 plot(time,x,'k');
31 title(['Rectangular Pulse ampiezza = ', num2str(T), 's']);
32 xlabel('tempo(s)');
33 ylabel('ampiezza');
34
35
36 % Passi:
37 % calcolo il vettore frequenze (N valori da -mu_s/2 a mu_s/2)
38 % calcolo la fft e la fft riordinata
39 % visualizzo lo spettro di ampiezza
40 % visualizzo lo spettro di fase
41
42
43 N = length(x); % N campioni, N freqs nella fft
44 mu_sampling = mu_s/N; % una ogni mu_s/N
45 mu_max = mu_s/2; % metà' a sx e metà' a dx dello zero
46
47 % vettore delle frequenze: N valori da -mu_s/2 a mu_s/2
48 mu = -mu_max:mu_sampling:mu_max-mu_sampling;
49
50 X = fft(x);
51
52 % Eseguo operazione di centratura dello spettro
53 % e calcolo spettro di ampiezza
54 shifted = fftshift((X));
55 X1 = abs(shifted);
56
57 figure(2)
58 plot(mu,X1,'r');
59 title('Spettro di Magnitudo');
60 xlabel('frequenza (Hz)')
61 ylabel('magnitudo |X(f)|');
62
63 figure(3)
64 plot(mu,180*angle(X)/pi,'r');
65 title('Spettro di Fase');
66 xlabel('frequenze (Hz)')
67 ylabel('fase/\pi');
68
69
70
71 % Seconda parte (se si vuole): stessa analisi su un segnale audio
72 % registrato direttamente in MATLAB, della durata di 4 secondi
73
74 % Registro segnale audio
75 recObj = audiorecorder;
76 disp('Start')
77 recordblocking(recObj,4);
78 disp ('End')
79
80 play(recObj)
81 myVoice = getaudiodata(recObj);
82
83 % in recObj posso recuperare la frequenza di campionamento
84 % e il numero di campioni
85 mu_s = recObj.SampleRate; % Frequenza di campionamento
86 N = recObj.TotalSamples; % Numero di campioni
87
88

```

```

89 % Passi:
90 % calcolo il vettore frequenze (N valori da -mu_s/2 a mu_s/2)
91 % calcolo la fft e la fft riordinata
92 % visualizzo lo spettro di ampiezza
93 % pulisco lo spettro e tolgo le frequenze troppo basse
94 % visualizzo lo spettro di fase
95
96 mu_m = mu_s/2;
97 freq = -mu_m:mu_s/N:mu_m-mu_s/N; % vettore frequenze
98 F = fft(myVoice);
99 FF = fftshift(F);
100
101 figure(1)
102 subplot(131)
103 plot([1:N],myVoice);
104 xlabel('#campioni')
105 ylabel('ampiezza')
106 title ('Segnale registrato')
107
108 subplot(132)
109 plot(freq,(abs(FF)));
110 xlabel('frequenze (Hz)')
111 ylabel('magnitudo')
112 title ('DFT (abs) del segnale ')
113
114 % Per la fase: ho un risultato molto rumoroso,
115 % quindi devo applicare un filtraggio restrittivo.
116 % decido di pulire lo spettro cancellando i residui di
117 % frequenza sotto un quarto del massimo valore
118 th = max(abs(FF))/4;
119 FF(abs(FF) < th) = 0;
120 theta = angle(FF);
121
122 subplot(133);
123 stem(freq,180*(theta)/pi)
124 xlabel ('frequenza (Hz)')
125 ylabel ('fase/\pi')
126 grid
127 title('Spettro di fase');

```

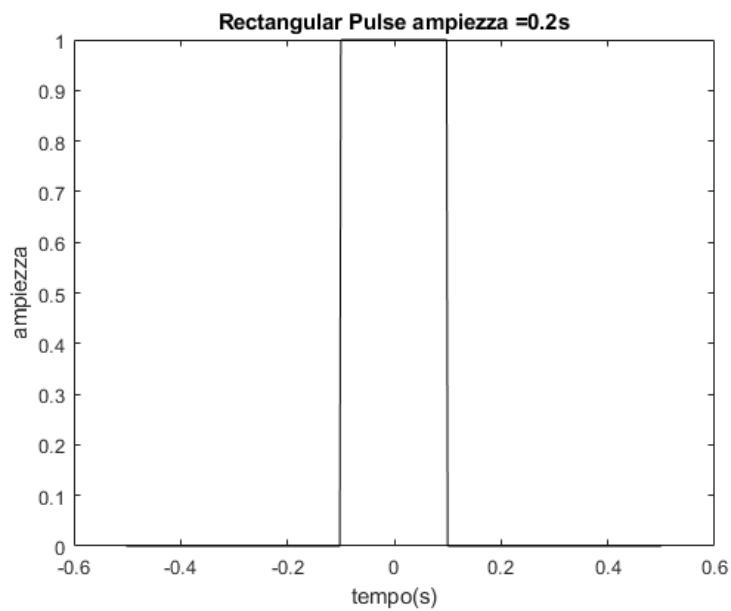


Figura 56: Rappresentazione del segnale box.

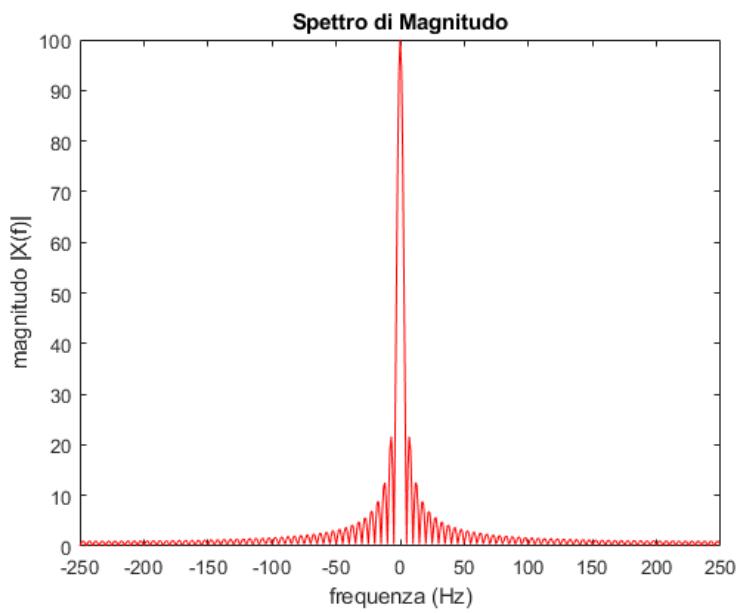


Figura 57: Spettro di magnitudo di una box.

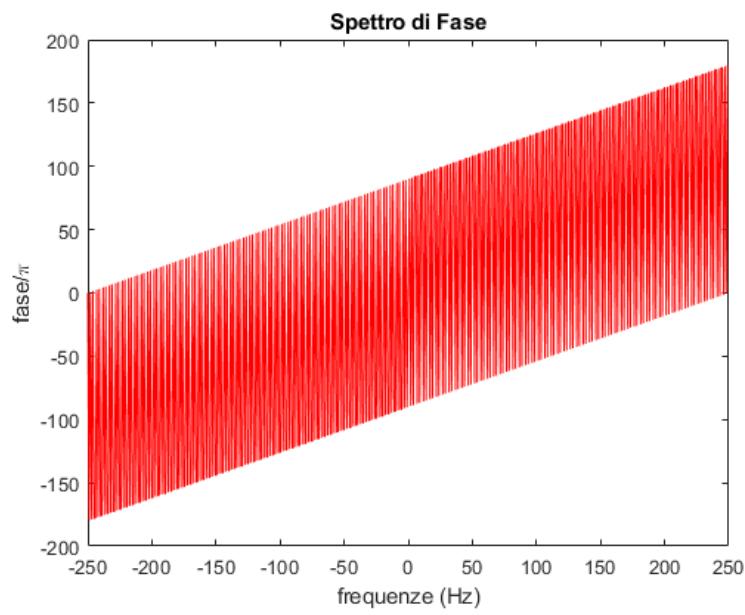


Figura 58: Spettro di fase di una box.

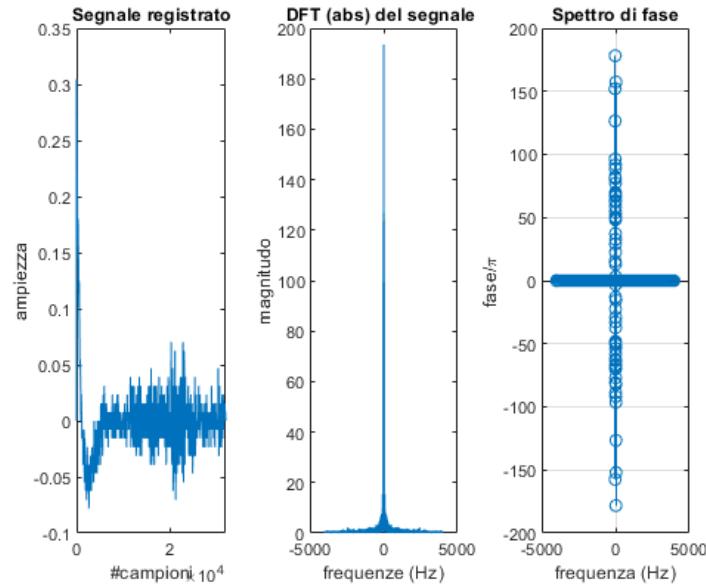


Figura 59: Stessa analisi eseguita sulla box, ma in questo caso su un segnale audio.

6.5.5 Esercizio 2

Verificare il fenomeno dell'aliasing. In particolare:

- Partire dal segnale sinusoidale $\sin(2\pi \cdot fsig \cdot t)$, dove $fsig = 10$ è la frequenza del segnale.
- Campionare un secondo di segnale ad una determinata frequenza ed effettuare l'analisi di Fourier (provare con le seguenti frequenze: [200, 100, 40, 20, 15, 14, 10]).
- Per quali di queste avviene il fenomeno dell'aliasing (cioè non riesco a ricostruire lo spettro)? È corretto rispetto alla teoria?

Soluzione

Il codice della soluzione è il seguente:

```
1 %% Esercizio 2
2 % Verificare il fenomeno dell'aliasing.
3 % In particolare:
4 % - partire dal segnale sinusoidale sin(2*pi*fsig*t),
5 %   dove fsig = 10 e' la frequenza del segnale
6 % - campionare un secondo di segnale ad una determinata
7 %   frequenza ed effettuare l'analisi di Fourier
8 % - Provare con le seguenti frequenze:
9 %   [200, 100, 40, 20, 15, 14, 10]
10 % Per quali di queste avviene il fenomeno dell'aliasing?
11 % E' corretto rispetto alla teoria?
12
13
14 clear all
15 close all
16 clc
17
18 % creo il segnale campionato con frequenza Fs
19 % Suggerimento:
20 % Fisso Fs
21 % Calcolo DeltaT (1/Fs)
22 % Creo un vettore t per i tempi che va da 0 a 1 (1 secondo)
23 % con passo DeltaT creo il segnale con fsig = 10
24 % y = sin(2*pi*fsig*t);
25
26 for Fs = [200, 100, 40, 20, 15, 14, 10]
27     Dt = 1/Fs;           % deltaT, passo
28     t = 0:Dt:1-Dt;       % prendo un secondo di durata
29     fsig = 10;           % frequenza sinusoidale
30     y = sin(2*pi*fsig*t); % segnale
31
32
33 % effettuo l'analisi di Fourier e
34 % visualizzo lo spettro riordinato
35
36 N = length(t);           % numero di campioni
37 Y = fft(y);
38 mu_samplig = Fs/N;
39 mu = -Fs/2:mu_samplig:Fs/2-mu_samplig;
40 YY = fftshift(Y);
41
42 close
43
```

```
44     figure(1)
45     subplot(211)
46     stem(t,y,'filled')
47     hold on;
48     plot(t,y)
49     title(['Segnale campionato con Fs = ',num2str(Fs)])
50
51
52     subplot(212)
53     stem(mu,abs(YY),'filled')
54     title('DFT del segnale')
55     xlabel('frequenze (Hz)')
56     ylabel('magnitudo')
57
58     pause
59 end
```

6.6 Operatori puntuali

6.6.1 Operazioni puntuale di rinforzo

Un **operatore puntuale** è un operatore che prende in **input il valore di un pixel** e **restituisce un pixel con un altro valore**. Il valore risultante dipende esclusivamente dal valore del pixel in ingresso. L'**obiettivo principale** di questi operatori è **variare il contrasto**.

Gli operatori puntuali di rinforzo lavorano con la **lookup table** (LUT), in cui la prima riga è degli input e la seconda degli output:

r	0	1	2	3	4	5	6	...
s	T(0)	T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	...

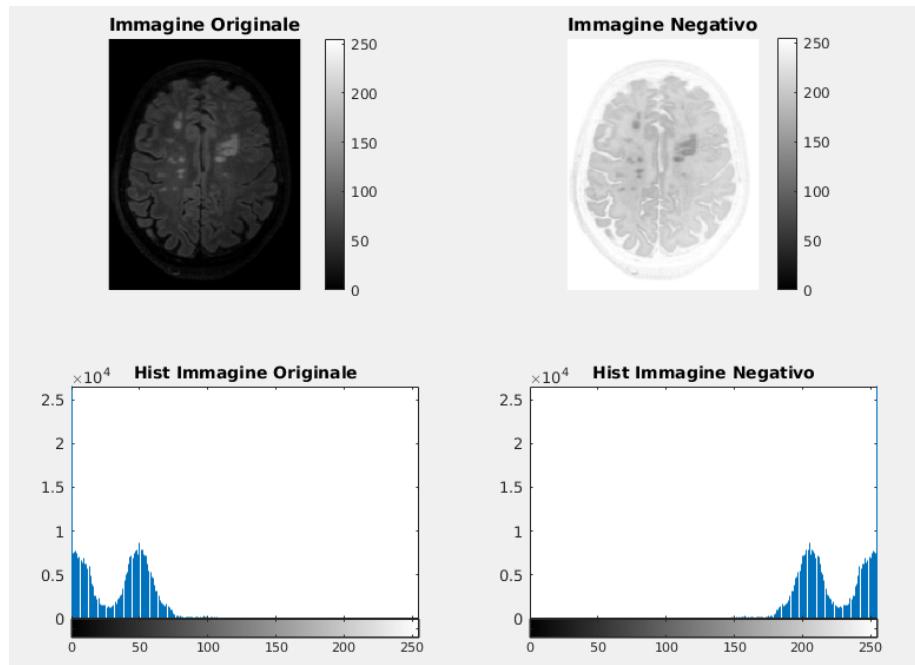
Quindi, se un pixel ha come livello di grigio il valore 3, allora viene sostituito con T(3). Questa operazione deve essere eseguita per tutti i pixel dell'immagine.

6.6.2 Negativo

L'operazione di **negativo** per i livelli di grigio è dettato dalla seguente formula:

$$s = 255 - r$$

E tale operazione viene utilizzata quando ci sono **dettagli di grigio immersi in zone nere che si vogliono evidenziare** (percettivamente è meglio avere dettagli scuri su regioni chiare).



Un esempio in MATLAB:

```

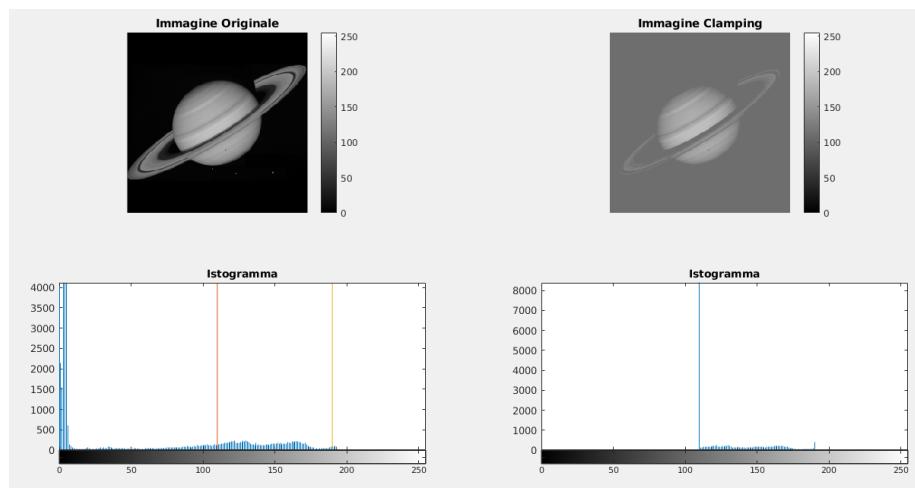
1 %% Operazione puntuale: Negativo
2 % Caricare l'immagine "FLAIR.png" e una volta convertita in scala
3 % di grigi da RGB farne il negativo. Visualizzare in un'unica
4 % figura con piu' subplot le due immagini e i corrispondenti
5 % istogrammi.
6
7 clear all;
8 close all;
9 clc;
10
11 MRI = imread('FLAIR.png');
12 MRI = rgb2gray(MRI);
13 MRI_neg = 255-MRI;
14
15 figure;
16 subplot(221);imshow(MRI); colorbar, title ('Immagine Originale')
17 subplot(222);imshow(MRI_neg);colorbar, title ('Immagine Negativo')
18 subplot(223);imhist(MRI); title ('Hist Immagine Originale')
19 subplot(224);imhist(MRI_neg); title ('Hist Immagine Negativo')
```

6.6.3 Clamping

Il **clamping** ha l'obiettivo di limitare le intensità ad un range definito $[a, b]$:

$$T(r) = \begin{cases} a & \text{se } r < a \\ r & \text{se } a \leq r \leq b \\ b & \text{se } r > b \end{cases}$$

Viene utilizzato nel caso in cui ci siano dei pixel di rumore molto chiari o molto scuri che devono essere mascherati, quindi sostituendoli con un altro valore.



Un esempio in MATLAB:

```

1 %% Operazione puntuale: Clamping
2 % Caricare l'immagine di saturno ("saturn2") dal file "imdemos.mat"
3 % (il file contiene una serie di immagini in B/N, e si trova dentro
4 % l'image processing toolbox)
5 % Fare l'operazione di clamping e visualizzare la LUT
6 % relativa a questa operazione.
7 % Infine, visualizzare le due immagini (originale/modificata) e i
8 % corrispondenti istogrammi
9 clear all;
10 close all;
11 clc;
12 load ('imdemos.mat','saturn2')
13 I = saturn2;
14
15 % Creo una LUT per operazione di clamping e la visualizzo
16 a = 110;
17 b = 190;
18 LUT=[];
19 for i = 0:255
20     r = i;
21     if r<a
22         LUT(i+1) = a; % MATLAB inizia l'indicizzazione a 1,
23                     % mentre le intensita' iniziano da 0
24     elseif r<=b & r>=a
25         LUT(i+1) = r;
26     elseif r>b
27         LUT(i+1) = b;

```

```

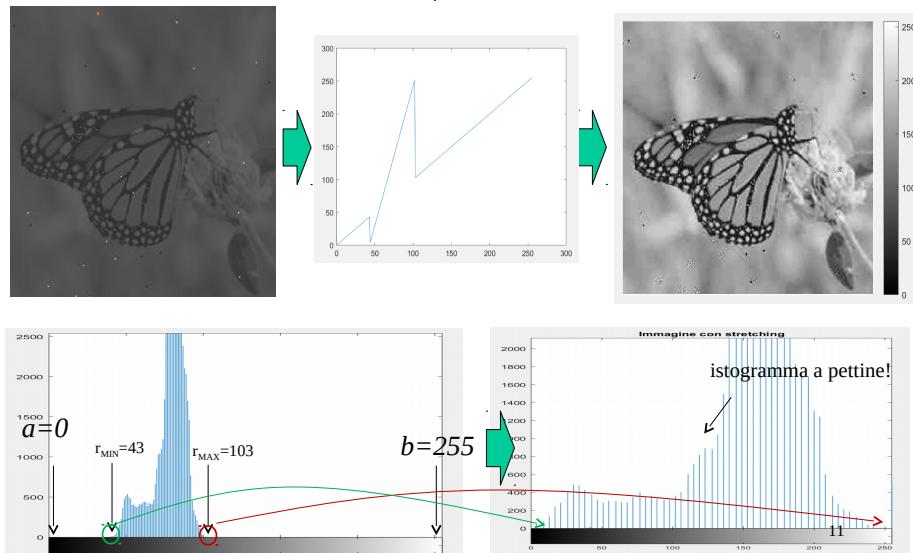
28     end
29 end
30
31 figure;
32 plot([0:255],LUT), xlim([0 255]), grid on, title ('Clamping LUT')
33 xlabel('Valori originali')
34 ylabel('Valori per Clamping LUT')
35
36 % Applico la LUT e visualizzo l'immagine dopo
37 % l'operazione di clamping
38 [r,c] = size(I);
39 Inew = zeros(size(I));
40 for i = 1:r
41     for j = 1:c
42         ldg = I(i,j);
43         newldg = LUT(ldg+1);% MATLAB inizia l'indicizzazione a 1,
44                             % mentre le intensita' iniziano da 0
45         Inew(i,j) = newldg;
46     end
47 end
48 Inew = uint8(Inew);
49 % I due cicli for si possono scrivere in forma compatta:
50 Inew2 = uint8(LUT(I+1)); % MATLAB inizia l'indicizzazione a 1,
51                             % mentre le intensita' iniziano da 0
52 % % To check
53 % sum(abs(Inew2-Inew), 'all')
54 % % ans =
55 % %      0
56
57 figure;
58 subplot(221);imshow(saturn2); colorbar, title ('Immagine Originale'
    )
59 subplot(223);imhist(saturn2); title ('Istogramma')
60 hold on; plot([a a],[0 max(imhist(saturn2))]);
61 plot([b b],[0 max(imhist(saturn2))])
62 subplot(222);imshow(Inew); colorbar, title ('Immagine Clamping')
63 subplot(224);imhist(Inew); title ('Istogramma')

```

6.6.4 Stretching/Shrinking

Lo **stretching** (stiramento) e lo **shrinking** (comprime) eseguono **attività di stiramento/comprime delle intensità** di un range $[r_{\min}, r_{\max}]$ portando l'intervallo ad uno definito $[a, b]$. In generale, viene usato per migliorare il contrasto eseguendo un'operazione di ampliamento o riduzione dello spettro dei livelli di grigio:

$$s = \left[\frac{r - r_{\min}}{r_{\max} - r_{\min}} \right] [b - a] + a$$



6.6.5 Trasformazione non lineare

La **trasformazione non lineare** viene utilizzata per effettuare una **trasformazione non uniforme che agisce differentemente sui livelli di grigio**. Nel caso in cui si voglia mappare fasce strette di valori dell'immagine originale in fasce più ampie, viene applicata la trasformazione non lineare così da aumentare il range del contrasto rendendo l'interpretazione umana più informativa.

Due casi in cui la trasformazione non lineare può essere utile:

- Immagine **sottoesposta**: i particolari interessanti sono poco evidenti e concentrati nelle **zone scure**.

In questo caso, con la trasformazione non lineare è possibile espandere la dinamica associata ai livelli scuri e comprimere quella dei livelli chiari;

- Immagine **sovraesposta**: i particolari interessanti sono poco evidenti e concentrati nelle **zone chiare**.

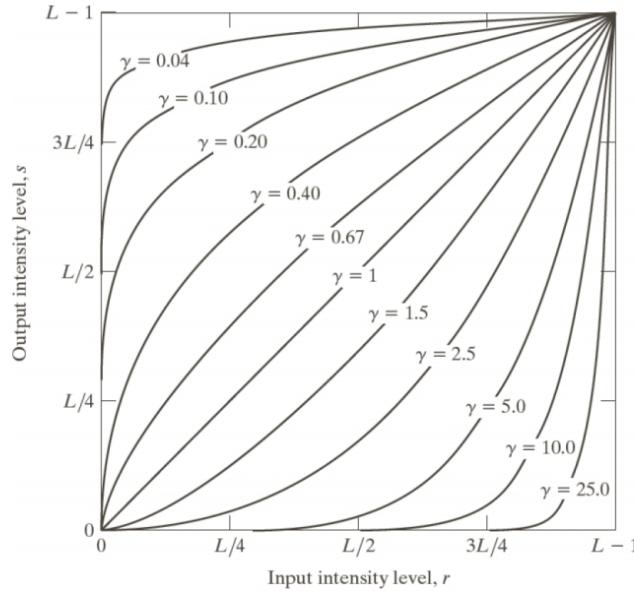
In questo caso, con la trasformazione non lineare è possibile espandere la dinamica associata ai livelli chiari e comprimere quella dei livelli scuri.

Vi è anche un'altra formula che è **ltrasformazione potenza** con la seguente formula:

$$s = cr^\gamma$$

In cui $c, \gamma > 0$ e dove c rappresenta la scelta **in dipendenza da** γ in modo da normalizzare i valori di s nell'intervallo $[0, 255]$. In altre parole:

$$\tilde{s} = r^\gamma \quad s = \left[\frac{\tilde{s} - \tilde{s}_{\min}}{\tilde{s}_{\max} - \tilde{s}_{\min}} \right] [MAX - MIN] + MIN$$



La scelta di γ influisce sul risultato:

- $\gamma < 1$ si ha un'espansione di bassi valori di r e una compressione per alti valori di r . I **dettagli** sono nei livelli di **grigio scuro**. L'**obiettivo** è quello di aumentare il range dei pixel scuri, ottenendo così uno schiarimento dell'immagine:



- $\gamma > 1$ si ha una compressione per bassi di valori di r , mentre un'espansione per alti valori di r . I **dettagli** sono nei livelli di **grigio chiari**. L'**obiettivo** è quello di aumentare il range dei pixel chiari, ottenendo un generale oscuramento dell'immagine:



Un esempio in MATLAB:

```

1 %% Operazione puntuale: Trasformazione Non lineare (Log/Potenza)
2 % ---%
3 % La trasformazione lineare espande in modo uniforme la dinamica
4 % originale dell'immagine, producendo un effetto globale di
5 % miglioramento del contrasto.
6 % In alcuni casi pero' si ha l'esigenza di effettuare una
7 % trasformazione non uniforme, che agisca differentemente sui
8 % livelli di grigio. In particolare:
9 % Immagine sottoesposta = i particolari interessanti sono poco
10 % evidenti e concentrati nelle zone scure;
11 % in tal caso possiamo espandere la
12 % dinamica
13 % associata ai livelli scuri e comprimere
14 % quella dei livelli chiari;
15 % Immagine sovraesposta = i particolari interessanti sono poco
16 % evidenti e concentrati nelle zone chiare;
17 % in tal caso possiamo espandere la
18 % dinamica

```

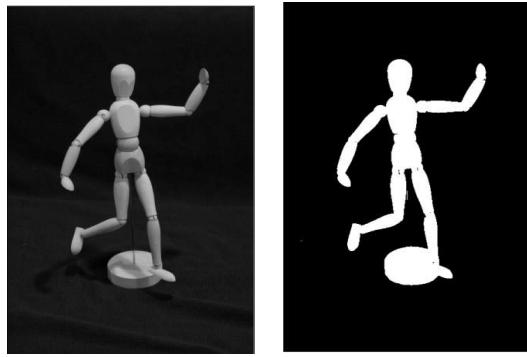
```

16 %                                              associata ai livelli chiari e comprimere
17 %                                              quella dei livelli scuri.
18 % L'espansione non uniforme della dinamica dell'immagine si puo'
19 % ottenere mediante elevazione a potenza dei livelli originari.
20 clear all;
21 close all;
22 clc;
23
24 % carico immagine "Spine.png" e applico trasformazione
25 % a potenza per migliore il contrasto
26 img = imread('Spine.png');
27 img = (rgb2gray(img));
28 figure, imshow(img), colorbar
29 figure, imhist(img), colorbar
30
31 % I dettagli che mi interessano sono nei livelli di grigio scuri --
32 % --> gamma <1
33 gamma = 0.4;
34 s_tilde = double(img).^gamma;
35 s_tilde_min = double(min(s_tilde(:)));
36 s_tilde_max = double(max(s_tilde(:)));
37 MAX = 255;
38 MIN = 0;
39 s = ((s_tilde - s_tilde_min)./(s_tilde_max-s_tilde_min))*(MAX-MIN)
    + MIN;
40
41 s = uint8(s);
42 figure, imshow(s), colorbar
43 figure, imhist(s), colorbar
44
45
46 %
47 % carico immagine "satellite.jpg" e applico trasformazione
48 % di potenza per migliore il contrasto
49 img = imread('satellite.jpg');
50 img = (rgb2gray(img));
51 figure, imshow(img), colorbar
52 figure, imhist(img), colorbar
53
54 % I dettagli che mi interessano sono nei livelli di grigio chiari
    --
55 % --> gamma >1
56 gamma = 4;
57 s_tilde = double(img).^gamma;
58 s_tilde_min = double(min(s_tilde(:)));
59 s_tilde_max = double(max(s_tilde(:)));
60 MAX = 255;
61 MIN = 0;
62 s = ((s_tilde - s_tilde_min)./(s_tilde_max-s_tilde_min))*(MAX-MIN)
    + MIN;
63
64 s = uint8(s);
65
66 figure, imshow(s), colorbar
67 figure, imhist(s), colorbar

```

6.6.6 Binarizzazione

La **binarizzazione** produce un'immagine che ha solo due livelli: nero e bianco. Si ottiene scegliendo una soglia T e impostando al colore nero tutti i pixel il cui valore è minore a T e al colore bianco tutti i pixel rimanenti. La binarizzazione solitamente viene usata per discriminare un oggetto dalla scena. Tuttavia, la difficoltà risiede nel saper scegliere una soglia T ragionevole.



Un esempio in MATLAB:

```
1 %% Operazione puntuale: Binarizzazione ---
2 % Produce una immagine che ha solo due livelli: nero e bianco.
3 % Si ottiene scegliendo una soglia T e mettendo a nero tutti i
4 % pixel il cui valore e' minore a T e a bianco tutti gli altri.
5 % La difficolta' risiede nel saper scegliere la soglia T
6 % piu' ragionevole.
7 % Ci sono metodi per scegliere automaticamente la
8 % soglia (Metodo di Otsu)
9 clear all;
10 close all;
11 clc;
12
13
14 I = imread('coins.png');
15 figure
16 set(gcf,'name','Binarizzazione di immagini')
17 subplot(121); imshow(I); title('Immagine originale');
18
19
20 T = 80; % soglia
21 BW = zeros(size(I)); % immagine binaria
22 BW(I>T) = 1;
23 subplot(122); imshow(BW); title('Immagine binarizzata');
24
25 % Nota: si puo' ottenere lo stesso risultato utilizzando
26 % il comando imbinarize.
27 % La soglia deve essere tra 0 e 1
28 % Si veda doc imbinarize
29
30 % BW2 = imbinarize(I,T/255);
31 % % check che siano uguali
32 % sum(abs(BW-BW2),'all')
33 % % ans =
34 % % 0
```

6.6.7 Esercizio 1

Implementare l'operazione puntuale di stretching/shrinking e applicarla all'immagine presente nel file "fog.mat":

- Caricare il file "fog.mat" (auto nella nebbia);
- Visualizzare l'immagine e il corrispondente istogramma;
- Individuare i limiti r_{min} , r_{max} per l'operazione di stretching/shrinking;
- Costruire la LUT (Look Up Table) per l'operazione di stretching/shrinking.
- Applicare la LUT all'immagine originale e visualizzare il risultato ottenuto: come cambia? Si riescono a visualizzare meglio i numeri della targa?
- Visualizzare l'istogramma dell'immagine risultato: come cambia rispetto all'istogramma dell'immagine originale?

Soluzione

Il codice MATLAB dell'esercizio:

```
1 %% Esercizio 1
2 % Implementare l'operazione puntuale di stretching/shrinking e
3 % applicarla all'immagine presente nel file "fog.mat"
4 % - caricare il file "fog.mat" (auto nella nebbia)
5 % - visualizzare l'immagine e il corrispondente istogramma
6 % - individuare i limiti rmin rmax per l'operazione
7 %   di stretching/shrinking
8 % - costruire la LUT (Look Up Table) corrispondente
9 % - applicare la LUT all'immagine originale e visualizzare
10 %   il risultato ottenuto: come cambia? Si riesce a
11 %   visualizzare meglio i numeri della targa?
12 % - visualizzare l'istogramma dell'immagine risultato:
13 %   come cambia rispetto all'istogramma dell'immagine
14 %   originale?
15 %
16 % Suggerimento: fare attenzione che in MATLAB gli array partono
17 % da 1, mentre i livelli di grigio vanno da 0 a 255 (La LUT e' un
18 % array con elementi in posizione da 1 a 256 ma con valori
19 % da 0 a 255).
20
21
22 clear all;
23 close all;
24 clc;
25
26 % Auto nella nebbia:
27 % come fare a visualizzare meglio i numeri della targa?
28 % Data l'operazione puntuale costruita, visualizzarne la
29 % Look Up Table (LUT)
30 load fog;
31 figure; set(gcf,'name','Immagine originale auto nella nebbia')
32 imshow(img); colorbar, title('Immagine originale')
33 figure; set(gcf,'name','Istogramma originale auto nella nebbia')
34 imhist(img); title ('Istogramma')
35
36 % Individuo i limiti massimo e minimo dell'istogramma ,
```

```

37 % noto che tali limiti non sono dovuti al rumore (non sono elementi
38 % distaccati dal corpo principale dell'istogramma)
39 r_max = double(max(img(:))); % 153
40 r_min = double(min(img(:)));// 128
41 % voglio usare tutti i livelli di grigio a disposizione, ovvero 256
42 a = 0;
43 b = 255;
44
45 % Creo una LUT per operazione di stretching
46 LUT=[];
47 for i = 0:255
48     r = i;
49     if r<=r_max & r>=r_min
50         s = ((r-r_min)/(r_max-r_min))*(b-a)+a;
51     else
52         s = r; % fuori dal range non ho modifiche
53     end
54     LUT(i+1) = s;
55 end
56
57
58 newimg = LUT(img+1); % come prima: applico la LUT
59 newimg = uint8(newimg); % rounding
60 figure; set(gcf,'name','Immagine auto nella nebbia trattata')
61 imshow(newimg); colorbar; title('Immagine con stretching')
62 figure; set(gcf,'name','Istogramma auto nella nebbia trattata')
63 imhist(newimg); title('Istogramma stretchato')
64
65 % NOTA IMPORTANTE: l'operazione di stretching si puo' fare con
66 % imadjust vedi doc imadjust
67 % L'operazione fatta sopra si ottiene come:
68 % newimg2 = imadjust(img,[r_min,r_max]./255,[a,b]./255);
69 % % check
70 % sum(abs(newimg-newimg2),'all')
71 % % ans =
72 % %      0

```



Figura 60: Immagine originale.

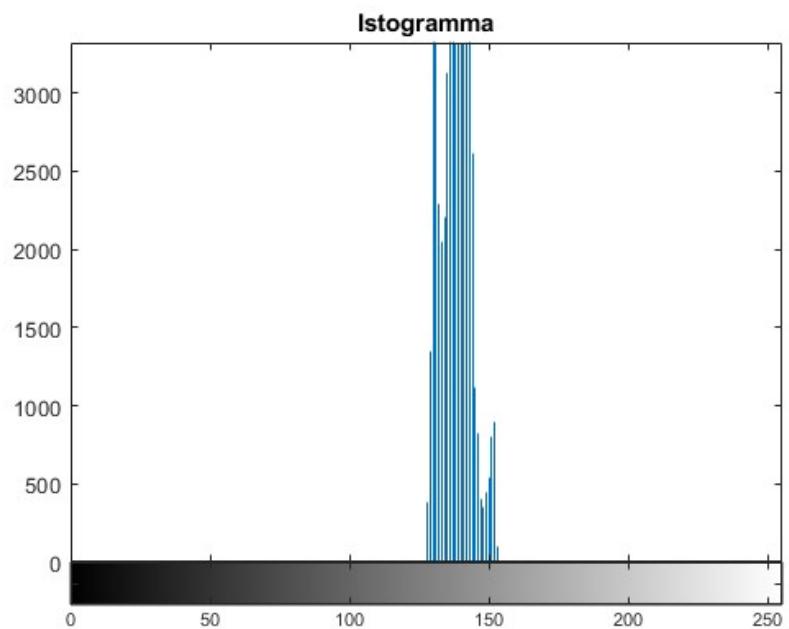


Figura 61: Istogramma dell'immagine originale.



Figura 62: Immagine dopo aver applicato lo stretching.

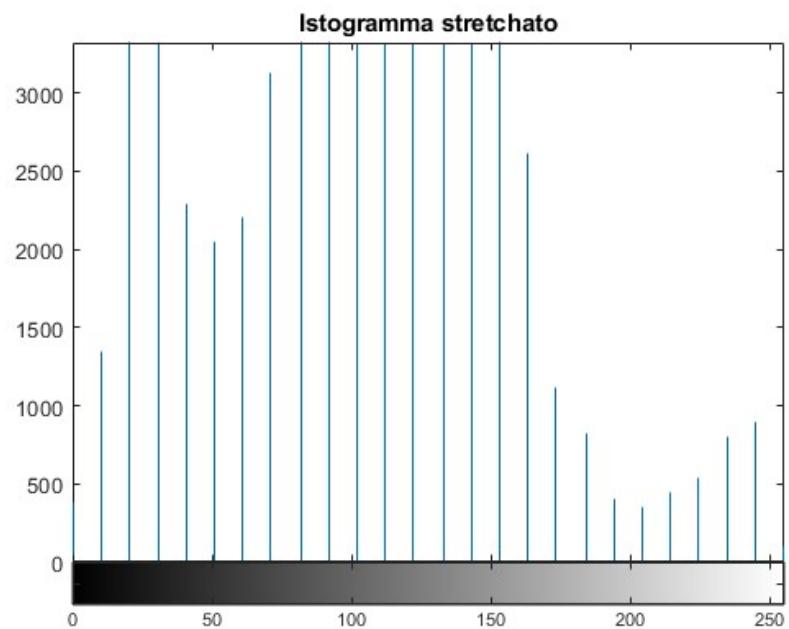


Figura 63: Istogramma dell'immagine modificata.

6.6.8 Esercizio 2

Caricare l'immagine “noisy.png”. Visualizzare l'immagine e il relativo istogramma. Cercare di migliorare l'immagine riducendo il rumore con un'operazione puntuale: qual'è l'operazione più adatta?

Soluzione

Il codice MATLAB dell'esercizio:

```
1 %% Esercizio 2
2 % - Caricare l'immagine 'noisy.png'
3 % - Visualizzare l'immagine e il relativo istogramma
4 % - Cercare di migliorare l'immagine riducendo il rumore
5 %   con un'operazione puntuale:
6 %   qual'è l'operazione piu' adatta?
7
8 clear all
9 close all
10 clc
11
12
13
14 I = imread('noisy.png');
15 figure, imshow(I), title('Immagine originale')
16
17 % Soluzione: clamping
18 % Visualizzo l'istogramma per capire i limiti
19 figure, imhist(I), title('Istogramma Immagine originale')
20
21 a = 40;
22 b = 100;
23 LUT=[];
24 for i = 0:255
25     r = i;
26     if r<a
27         LUT(i+1) = a;
28     elseif r<=b & r>=a
29         LUT(i+1) = r;
30     elseif r>b
31         LUT(i+1) = b;
32     end
33 end
34 % visualizzo la LUT
35 figure;
36 plot([0:255],LUT), xlim([0 255]), grid on
37
38 % clamping
39 I_clip = uint8(LUT(I+1));
40 % visualizzo l'immagine risultante
41 figure;
42 imshow(I_clip)
43
44 % NOTA: il risultato non e' perfetto, vedremo nella lezione
45 % sulle operazioni locali che esiste un metodo migliore
```



Figura 64: Immagine originale.

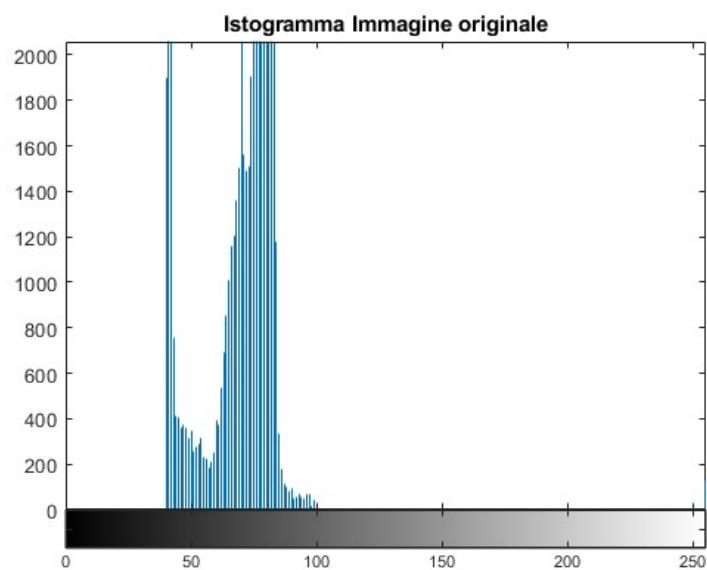


Figura 65: Istogramma immagine originale.

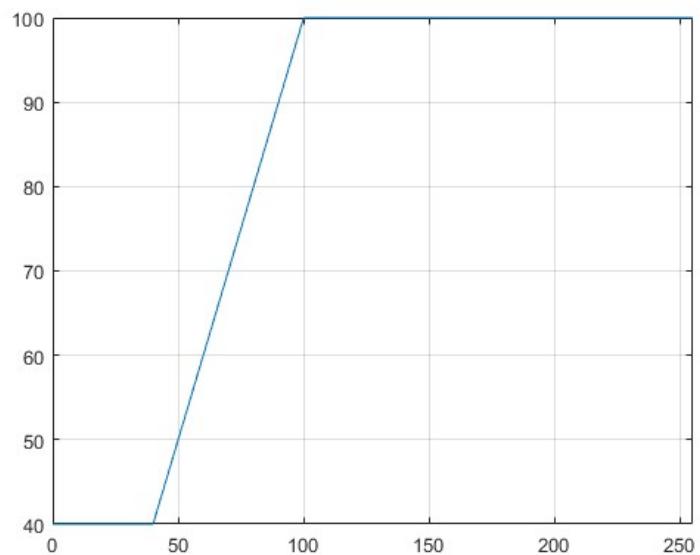


Figura 66: LUT.



Figura 67: Immagine dopo aver ridotto il rumore con il clamping.

6.7 Operatori locali

6.7.1 Filtraggio spaziale

Il valore di uscita di ogni pixel, dopo aver usato uno degli **operatori locali**, dipende da un limitato intorno del corrispondente punto in input. Vengono utilizzati per migliorare la qualità delle immagini o per estrarre delle informazioni dall'immagine.

Spesso tale filtraggio viene ottenuto facendo la convoluzione tra l'immagine ed una matrice, chiamata maschera o kernel. In questo caso, si parla di filtri lineari.

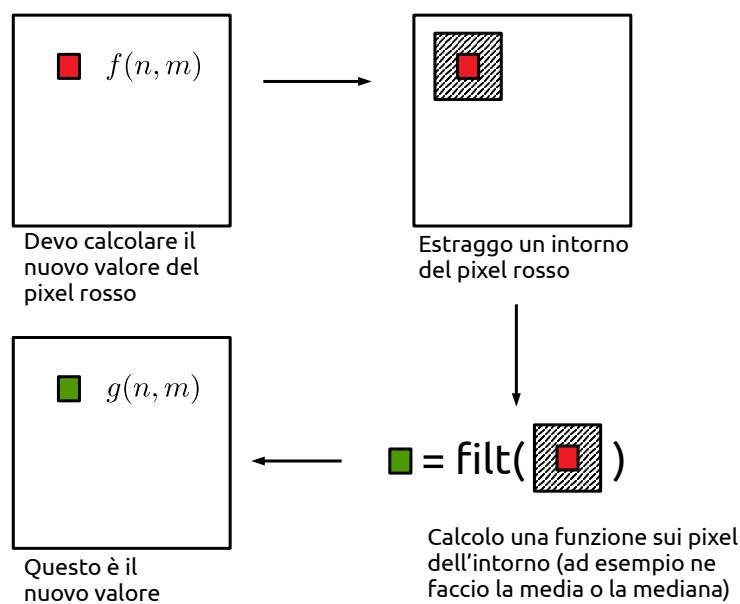


Figura 68: Funzionamento generale degli operatori locali.

Per applicare un operatore locale, devono essere decisi alcuni valori:

- La **dimensione dell'intorno**, che tipicamente è $K \times K$ con K dispari;
- La **funzione con cui viene calcolato il nuovo valore del pixel**.

6.7.2 Tipologie di filtro

Esistono diverse tipologie di filtro che a loro volta si dividono in macro categorie:

- Filtri per smoothing (tolgono il rumore):
 - Filtro media
 - Filtro Gaussiano
 - Filtro mediano
 - Filtri per sharpening (aumentano il gradi di dettaglio delle immagini):
 - Filtro laplaciano
-

6.7.3 Filtro media

Il **filtro di media** è un **filtraggio lineare**, poiché il nuovo valore del pixel è la **media** dei valori dei pixel dell'intorno. Gli effetti di questo filtro sono i seguenti:

- Rimuove il rumore Gaussiano;
- L'immagine risultato è “addolcita” rispetto ai bordi degli oggetti delle immagini;
- Maggiore è l'ampiezza K della maschera, più severo sarà l'effetto della media sulla struttura dell'immagine.

Un **esempio** in MATLAB:

```
1 %% Esempio 1: Filtro Media
2 % Per creare un filtro locale media: fspecial
3 % si veda doc fspecial
4 % Creo un filtro media 3x3
5 H = fspecial('average',3);
6 % Per applicare il filtro all'immagine: imfilter
7 Km = imfilter(J,H,'replicate');
8 % 'replicate' e' un'opzione per gestire il "problema dei bordi"
9 % Visualizzo il risultato
10 subplot(223); imagesc(Km), colormap gray, axis image; title(
11     'Smoothing Media'), colorbar
12 % Calcolo SNR
13 SNRMmse = sum(Km(:).^2)/sum((Km(:)-I(:)).^2);
14 SNRMvar = var(I(:))/var(Km(:));
15 fprintf('SNR img noisy + Mean smoothing \nMSE: %g\nVAR: %g\n',
16 SNRMmse, SNRMvar);
```

6.7.4 Filtro gaussiano

Il **filtro Gaussiano** è simile al filtro di media poiché rimuove il rumore gaussiano. Questo filtro è una **media pesata**, in cui i pesi più vicini al centro hanno valori più alti (i valori vicini al pixel centrale hanno una priorità più elevata). Per **esempio**:

$$\frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Il suo compito è quello di effettuare uno *smoothing* più lieve, preservano di contorni meglio di quanto faccia il filtraggio di media. Tuttavia, ha una riduzione del rumore minore rispetto al filtro di media.

La formula che lega l'ampiezza della maschera (W) e sigma (deviazione standard del filtro):

$$W = 5 \cdot \sigma$$

Un **esempio** in MATLAB:

```

1 %% Esempio 2: Filtro Gaussiano
2 % Per creare un filtro locale gaussiano: fspecial
3 % filtro gaussiano KxK con standard deviation 0.6
4 H = fspecial('gaussian',3,0.6);
5 % Importante: relazione tra K (W nella teoria) e sigma:
6 % W = 5*sigma
7 Kg = imfilter(J,H,'replicate');
8 subplot(224); imagesc(Kg), colormap gray, axis image, title('
    Smoothing Gaussiano'), colorbar
9 SNRGmse = sum(Kg(:).^2)/sum((Kg(:)-I(:)).^2);
10 SNRGvar = var(I(:))/var(Kg(:));
11 fprintf('SNR img noisy + Gaussian smoothing \nMSE: %g\nVAR: %g\n',
    SNRGmse,SNRGvar);

```

6.7.5 Filtro mediano

Il **filtro mediano** è considerato un filtraggio **non lineare** poiché il **nuovo** valore del pixel è la **mediana dei valori dei pixel dell'intorno**. Si ricorda che per ottenere la mediana è necessario ordinare i valori e prendere il valore centrale. Essa rappresenta una **stima robusta della media e serve per rimuovere rumore impulsivo**. Un **esempio** in MATLAB:

```
1 %% Esempio 3: Filtro Mediano
2 % In questo esempio si vedra' come e' possibile eliminare il rumore
3 % impulsivo utilizzando un filtro di smoothing mediano.
4 clear all
5 close all
6 clc
7
8 load imdemos
9 I = saturn2;
10 % aggiungo rumore sale e pepe all'immagine originale
11 J = imnoise(I,'salt & pepper',0.02);
12 figure; set(gcf,'name','Studio sintetico rumore e filtraggio');
13 subplot(221); imagesc(I), colormap gray, axis image; title('
    Originale')
14 colorbar
15 subplot(222); imagesc(J), colormap gray, axis image; title('Noisy')
16 colorbar
17 % casting per il calcolo del SNR
18 I = double(I);
19 J = double(J);
20 SNR1mse = sum(J(:).^2)/sum((J(:)-I(:)).^2);
21 SNR1var = var(I(:))/var(J(:));
22 fprintf('SNR img noisy (Salt-pepper)\nMSE: %g\nVAR: %g\n\n',SNR1mse
    ,SNR1var);
23
24
25 % per applicare il filtro mediano si usa
26 % la funzione medfilt2
27 Kmed = medfilt2(J);
28 % senza parametri si applica un filtro 3x3
29 subplot(223); imagesc(Kmed), colormap gray, axis image; title('
    Smoothing Mediano'), colorbar
30
31 SNRMedmse = sum(Kmed(:).^2)/sum((Kmed(:)-I(:)).^2);
32 SNRMedvar = var(I(:))/var(Kmed(:));
33 fprintf('SNR img noisy (Salt-pepper) + Median smoothing\nMSE: %g\
    nVAR: %g\n\n',SNRMedmse,SNRMedvar);
```

6.7.6 Filtro laplaciano

Il **filtro laplaciano** è un operatore di *sharpening* e ha l'obiettivo di evidenziare i dettagli come *post processing* dopo filtraggi di *smoothing*. Questo perché i filtraggi di *smoothing* eliminano i dettagli.

Il filtro laplaciano è un filtro lineare in cui la **maschera**, identificata con la lettera H , è **caratterizzata da coefficienti di un segno vicino al centro e di segno opposto nella periferia esterna**. Per **esempio**:

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Per definire il filtro Laplaciano occorre definire un **parametro** α il cui significato è legato all'**importanza che si vuole dare agli edge**:

- $\alpha = 0$ —> Verticali e orizzontali
- $\alpha = 1$ —> Diagonali
- $\alpha = 0.5$ —> Tutti gli *edge*

Per **esempio**:

$$h = \frac{1}{\alpha + 1} \begin{bmatrix} -\alpha & \alpha - 1 & -\alpha \\ \alpha - 1 & \alpha + 5 & \alpha - 1 \\ -\alpha & \alpha - 1 & -\alpha \end{bmatrix}$$

Per ottenere l'immagine con più dettagli è possibile applicare il filtro ***Basic Highpass Spatial filtering***. Per farlo, viene eseguita una somma tra l'immagine originale e l'immagine filtrata con il laplaciano:

$$g(n, m) = f(n, m) + c \cdot \underbrace{h * f(n, m)}_{\text{Immagine filtrata con il laplaciano}}$$

In cui c indica il fattore di amplificazione ed è positivo se il valore centrale della maschera H è positivo, altrimenti è negativo. Un **esempio** in MATLAB:

```

1 %% Esempio 4 - Filtraggio Locale: Sharpening ---
2 % Esempio di applicazione di sharpening.
3
4 clear all; close all; clc
5
6 A = imread('moon.tif');
7 figure; imshow(A); title ('Immagine partenza')
8
9 alpha = 0.2; %valore di default
10 H = fspecial('laplacian',alpha);
11 B = imfilter(A,H);
12 figure; imshow(B); title ([ 'Filtraggio sharpening con Laplaciano
    Alpha = ', num2str(alpha)])
13
14 % Basic highpass spatial filtering
15 % L'immagine finale e' l'immagine originale
16 % sommata con l'immagine filtrata con il laplaciano
17 cost = -1; % valore centrale di H e' negativo
18 C = double(A) + cost*double(B);
19 figure; imagesc(uint8(C)), colormap gray, axis image; title(
    'Immagine finale (basic highpass)')
```

6.7.7 Signal to noise ratio

Il **signal to noise ratio** viene utilizzato per stimare la **quantità di rumore in un'immagine**. Esistono due versioni:

- La formula con ***mean square***:

$$SNR_{\text{ms}} = \frac{\sum_{n=1}^N \sum_{m=1}^M \tilde{f}(n, m)^2}{\sum_{n=1}^N \sum_{m=1}^M [\tilde{f}(n, m) - f(n, m)]^2}$$

In cui f è l'immagine originale e \tilde{f} è l'immagine con rumore.

- La formula con la **varianza**:

$$SNR_{\text{var}} = \frac{\sigma_s^2}{\sigma_n^2}$$

Dove σ_s^2 è la deviazione standard dell'immagine senza rumore, e σ_n^2 è quella dell'immagine con il rumore.

Un **esempio** in MATLAB:

```

1 %% ESEMPI: creazione e applicazione diversi di filtri di smoothing
2 % e calcolo SNR
3 %
4 % Definisco l'immagine simulata I a cui poi aggiungo rumore
5 % gaussiano
6
7 clear all
8 close all
9 clc
10
11 I = ones(128,128).*128;
12 I(1:64,1:64) = 96;
13 I(65:128,65:128) = 160;
14 J = I + normrnd(0,5,128,128);
15 figure; set(gcf, 'name', 'Studio sintetico rumore e filtraggio');
16 subplot(221); imagesc(I), colormap gray, axis image; title('
    Originale')
17 colorbar
18 subplot(222); imagesc(J), colormap gray, axis image; title('Noisy')
19 colorbar
20
21 % Calcolo il SNR_{MSE} e SNR_{VAR} nell'immagine di partenza
22 % f = I (immagine senza rumore)
23 % ftilde = J (immagine con rumore)
24 num = sum(J(:).^2);
25 den = sum((J(:)-I(:)).^2);
26 SNR1mse = num/den;
27 SNR1var = var(I(:))/var(J(:));
28 % Attenzione:
29 % Se l'immagine e' uint8 occorre fare un casting per avere la
30 % precisione nei calcoli (e la funzione "var" prende in
31 % ingresso solo single/double):
32 % I = double(I);
33 % J = double(J);
34 % In questo esempio non ci sono problemi perche' e' un

```

```
35 % immagine sintetica definita come double  
36  
37 fprintf('SNR img noisy (Gauss)\nMSE: %g\nVAR: %g\n',SNR1mse,SNR1var  
);
```

6.7.8 Esercizio 1

Applicare diversi filtri di smoothing alle due immagini reali definite nello script Lezione7_EserciziPrincipali.m (“peacock” con rumore gaussiano e “peppers” con rumore impulsivo). Confrontare i risultati sia in modo qualitativo (quali ritornano un’immagine migliore?) e quantitativo (quali ritornano il valore di SNR più alto?). Utilizzare SNR versione VAR. I filtri da provare:

- Media: $3 \times 3, 5 \times 5, 9 \times 9$
- Gaussiano: $3 \times 3, 5 \times 5, 9 \times 9$
- Mediano: $3 \times 3, 5 \times 5, 9 \times 9$

Soluzione

Il codice risultante:

```
1 %% Esercizio 1. Smoothing
2 %
3 % - Applicare diversi filtri di smoothing alle due immagini reali
4 % definite sotto ("peacock" con rumore gaussiano e "peppers" con
5 % rumore impulsivo);
6 % - Confrontare i risultati sia in modo qualitativo (quali
7 % ritornano un'immagine migliore?) e quantitativo (quali
8 % ritornano il valore di SNR piu' alto?)
9 % -- Utilizzare SNR versione VAR;
10 % - Filtri da provare:
11 %   - media: 3x3, 5x5, 9x9
12 %   - gaussiano: 3x3, 5x5, 9x9 (attenzione a stimare correttamente
13 %                           la deviazione standard del filtro con la regola)
14 %   - mediano: 3x3, 5x5, 9x9
15 %
16
17
18 clear all;
19 close all;
20 clc;
21
22
23 % Immagine 1. peacock con rumore gaussiano
24 I1 = rgb2gray(imread('peacock.jpg'));
25 J1 = imnoise(I1,'gaussian',0.01);
26 figure, imshow(I1); set(gcf,'name','Originale');
27 figure, imshow(J1); set(gcf,'name','Rumore Gaussiano');
28
29 % % Immagine 2. pepper con rumore impulsivo
30 load imdemos
31 I2 = pepper;
32 % aggiungo rumore sale e pepe all'immagine originale
33 J2 = imnoise(I2,'salt & pepper',0.05);
34 figure, imshow(I2); set(gcf,'name','Originale');
35 figure, imshow(J2); set(gcf,'name','Rumore Gaussiano');
36
37
38 for esempio = 1:2
39     if esempio == 1
40         % Soluzione per I1,J1
41         I = I1;
42         J = J1;
43     else
```

```

44      % Soluzione per I2,J2
45      I = I2;
46      J = J2;
47  end
48  fprintf ('\n\nIMMAGINE %d\n',esempio);
49
50
51  % Step 1: calcolo SNR immagine originale con rumore
52  % Casting per il calcolo del SNR
53  I = double(I);
54  J = double(J);
55
56  SNRivar = var(I(:))/var(J(:));
57
58  % Step 2: Effettuo tutti i filtraggi: salvo i risultati in un
59  % array di celle
60  % Media
61  H = fspecial('average',3);
62  Filtered{1} = imfilter(J,H);
63  meth{1} = 'Media 3x3' ; % salvo il nome del filtro
64
65  H = fspecial('average',5);
66  Filtered{2} = imfilter(J,H);
67  meth{2} = 'Media 5x5' ; % salvo il nome del filtro
68
69  H = fspecial('average',9);
70  Filtered{3} = imfilter(J,H);
71  meth{3} = 'Media 9x9' ;
72
73
74  % Gaussiano
75  H = fspecial('gaussian',3,3/5);
76  Filtered{4} = imfilter(J,H);
77  meth{4} = 'Gaussiano 3x3';
78
79  H = fspecial('gaussian',5,5/5);
80  Filtered{5} = imfilter(J,H);
81  meth{5} = 'Gaussiano 5x5';
82
83  H = fspecial('gaussian',9,9/5);
84  Filtered{6} = imfilter(J,H);
85  meth{6} = 'Gaussiano 9x9';
86
87  % Mediano
88  Filtered{7} = medfilt2(J,[3 3]);
89  meth{7} = 'Mediano 3x3' ;
90
91  Filtered{8} = medfilt2(J,[5 5]);
92  meth{8} = 'Mediano 5x5' ;
93
94  Filtered{9} = medfilt2(J,[9 9]);
95  meth{9} = 'Mediano 9x9' ;
96
97  % calcolo e visualizzo i diversi SNR
98  fprintf ('SNR prima del filtraggio: VAR: %g \n',SNRivar);
99  for i = 1:9
100    F = Filtered{i};
101    SNRFiltvar = var(I(:))/var(F(:));
102    fprintf ('SNR dopo filtro %s: VAR: %g \n',meth{i},SNRFiltvar
103  );
104  end

```

```
104 % visualizzo le soluzioni
105 figure, set(gcf, 'name', 'Filtrate');
106 for i = 1:9
107     F = Filtered{i};
108     subplot(3,3,i)
109     imshow(uint8(F))
110     title(meth{i})
111 end
112
113 end
```

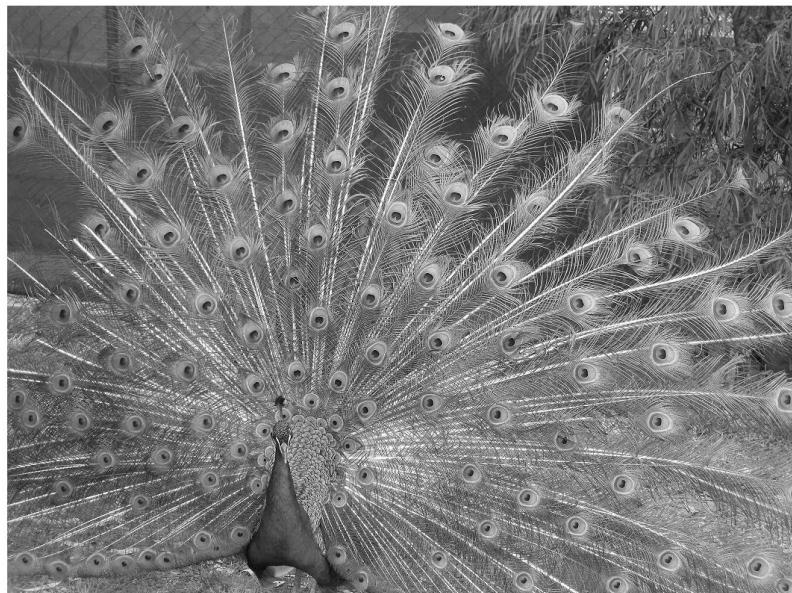


Figura 69: Immagine originale.

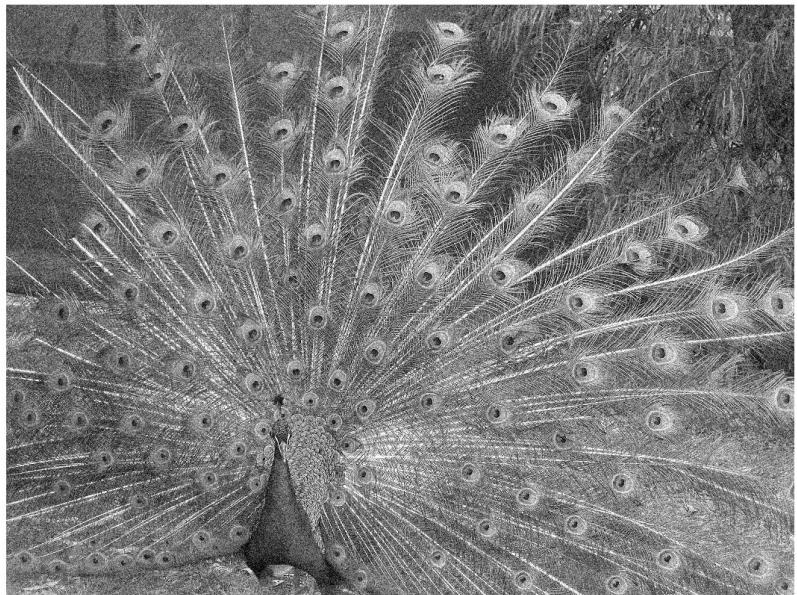


Figura 70: Immagine con rumore gaussiano.



Figura 71: Immagine originale.



Figura 72: Immagine con rumore gaussiano.

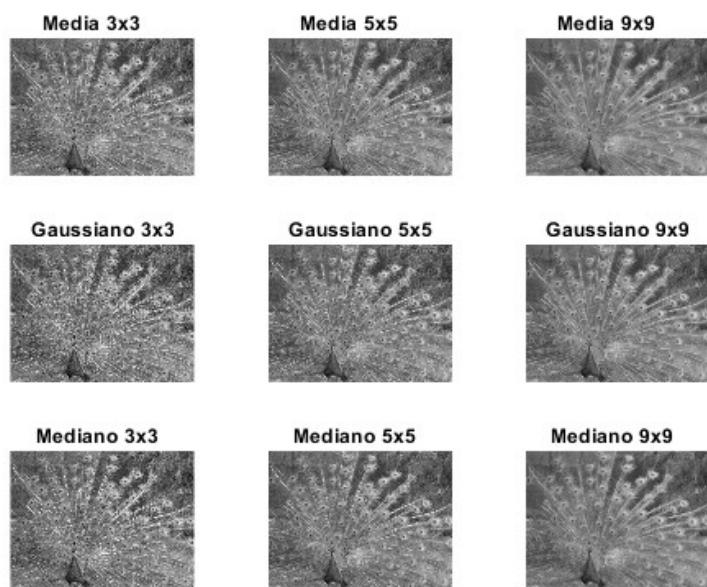


Figura 73: Filtri a confronto.

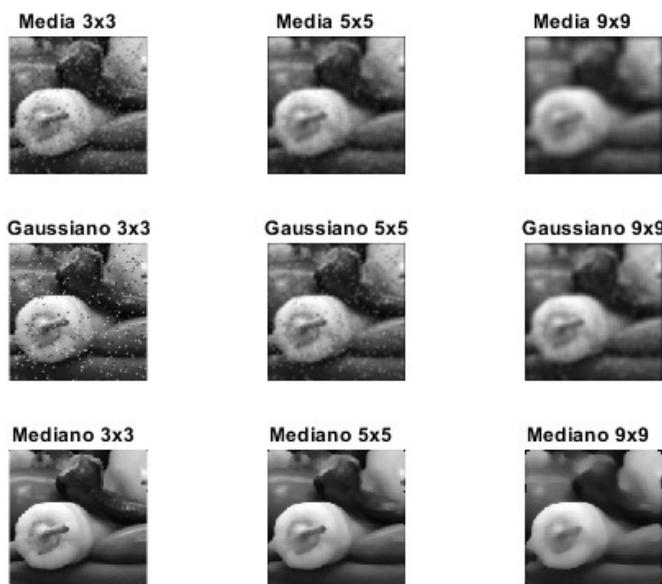


Figura 74: Altri filtri a confronto.

6.7.9 Esercizio 2

Applicare il filtraggio di sharpening all'immagine "Pavone" corrotta con rumore Gaussiano. Provare ad applicarlo direttamente all'immagine con rumore oppure dopo aver applicato un filtraggio media o gaussiano. Che differenza si nota?

Soluzione

Il codice risultante:

```
1 %% Esercizio 2. Sharpening
2 %
3 % - Applicare il filtraggio di sharpening all'immagine Pavone
4 % corrotta con rumore Gaussiano
5 % - Provare ad applicarlo direttamente all'immagine con rumore
6 % oppure dopo aver applicato un filtraggio media o gaussiano.
7 % Che differenza si nota?
8
9 clear all; close all; clc
10
11 I = rgb2gray(imread('peacock.jpg'));
12 figure, imshow(I); set(gcf,'name','Originale');
13
14 J = imnoise(I,'gaussian',0.01);
15 figure, imshow(J); set(gcf,'name','Rumore Gaussiano');
16
17 I = double(I);
18 J = double(J);
19
20 % Filtro sharpening
21 Hlap = fspecial('laplacian',0.5);
22 K1 = imfilter(J,Hlap);
23 cost = -1;
24 K1final = double(J) + cost*double(K1);
25 figure; set(gcf,'name','Pavone sharpened ');
26 imshow(uint8(K1final),[]); colorbar;
27
28
29 % Filtro smoothing media e poi sharpening
30 H = fspecial('average',9);
31 Km = imfilter(J,H);
32 Hlap = fspecial('laplacian',0.5);
33 Kml = imfilter(Km,Hlap);
34 cost = -1;
35 Kmlfinal = double(Km) + cost*double(Kml);
36 figure; set(gcf,'name','Pavone media smoothing: sharpened ');
37 imshow(uint8(Kmlfinal),[]); colorbar;
38
39 % Filtro smoothing gaussiano e poi sharpening
40 Msize = 9;
41 sigma = Msize/5;
42 H = fspecial('gaussian',9,sigma);
43 Km = imfilter(J,H);
44 Kgl = imfilter(Km,Hlap);
45 cost = -1;
46 Kglfinal = double(Km) + cost*double(Kgl);
47 figure;
48 set(gcf,'name','Pavone gaussian smoothing: sharpened ');
49 imshow(uint8(Kglfinal),[]); colorbar;
```



Figura 75: Immagine originale.

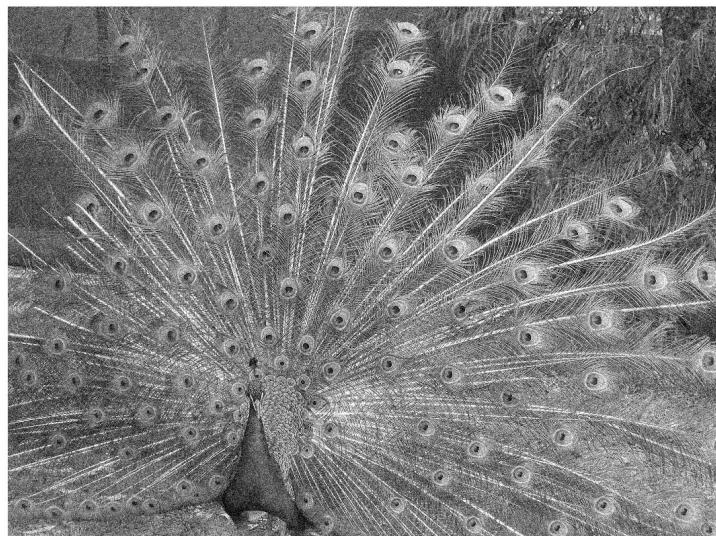


Figura 76: Immagine con rumore gaussiano.

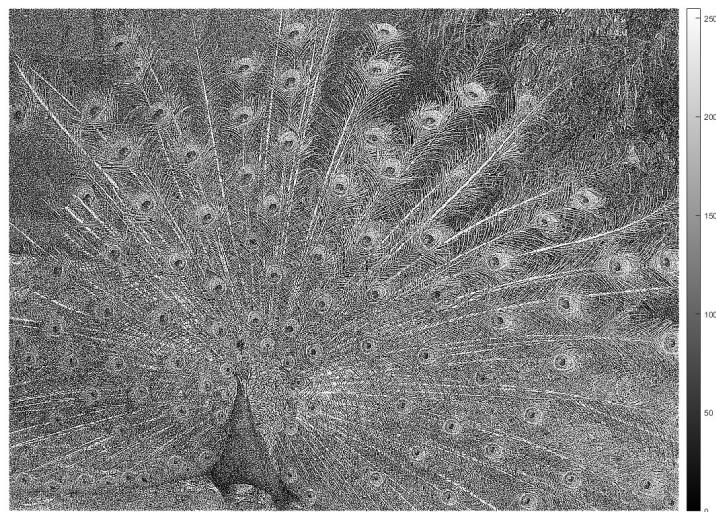


Figura 77: Immagine con sharpened.

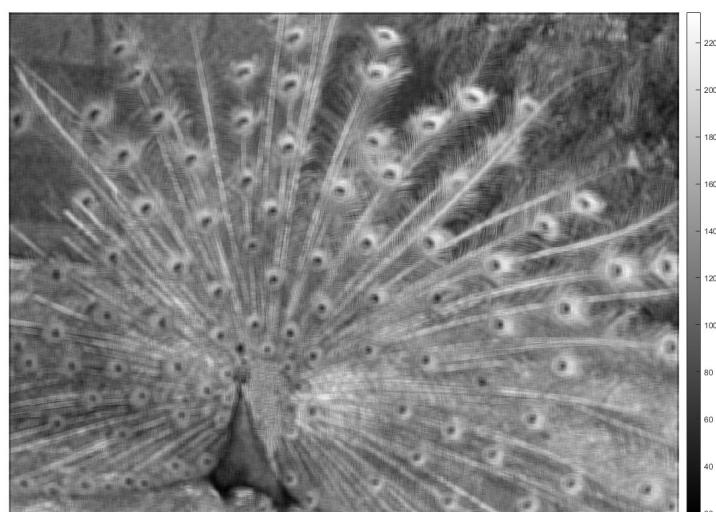


Figura 78: Immagine con filtro di media smoothing: sharpened.

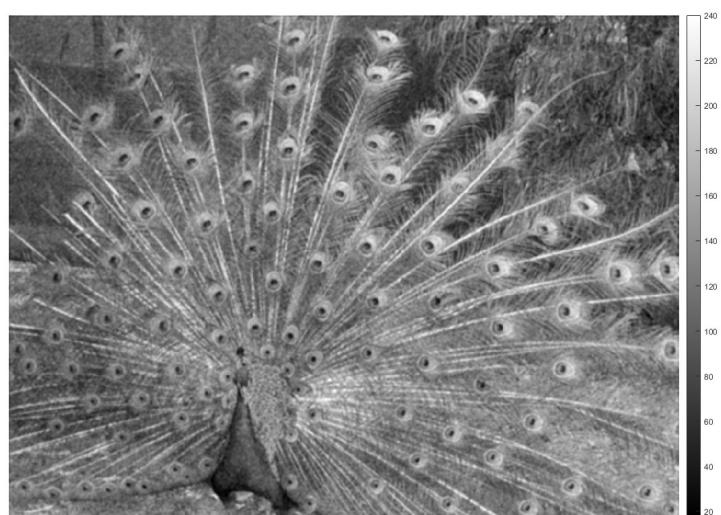


Figura 79: Immagine con gaussiano smoothing: sharpened.

6.7.10 Esercizio 3

Caso reale: riuscire a migliorare il più possibile la qualità della scrittura nel file “hand.tif”, utilizzando gli strumenti visti fino ad ora. Suggerimenti: ragionare sull’istogramma e utilizzare sia operatori puntuali che operatori locali.

Soluzione

Il codice risultante:

```
1 %% Esercizio 3. Caso di studio reale
2 %
3 % - Riuscire a migliorare il piu' possibile la qualita' della
4 %   scrittura nel file hand.png, utilizzando gli strumenti
5 %   visti fino ad ora;
6 % - Suggerimento: ragionare sull'istogramma, e utilizzare
7 %   sia operatori puntuali (lezione scorsa) che operatori
8 %   locali (questa lezione).
9
10 clear all; close all; clc
11
12 img = imread('hand.png');
13
14 figure(1);
15 imshow(img); title('Originale')
16
17
18 figure(2)
19 set(gcf,'name','Analisi scrittura a mano: istogrammi');
20 subplot(221); imhist(img); title('Originale')
21
22
23 % stretching
24 Cmin = 90;
25 Cmax = 240;
26 Cs = imadjust(img,[Cmin, Cmax]./255,[0,1]);
27 figure;
28 figure(3), imshow(Cs); title('Stretched')
29 figure(2), subplot(222); imhist(Cs); title('Stretched')
30
31 % sharpening
32 H = fspecial('laplacian',1);
33 Csl = imfilter(Cs,H);
34 cost = -1;
35 Cffinal = double(Cs) + cost*double(Csl);
36 figure;
37 figure(4), imshow(uint8(Cffinal)); title('Sharpened')
38 figure(2), subplot(223); imhist(uint8(Cffinal)); title('Sharpened')
```

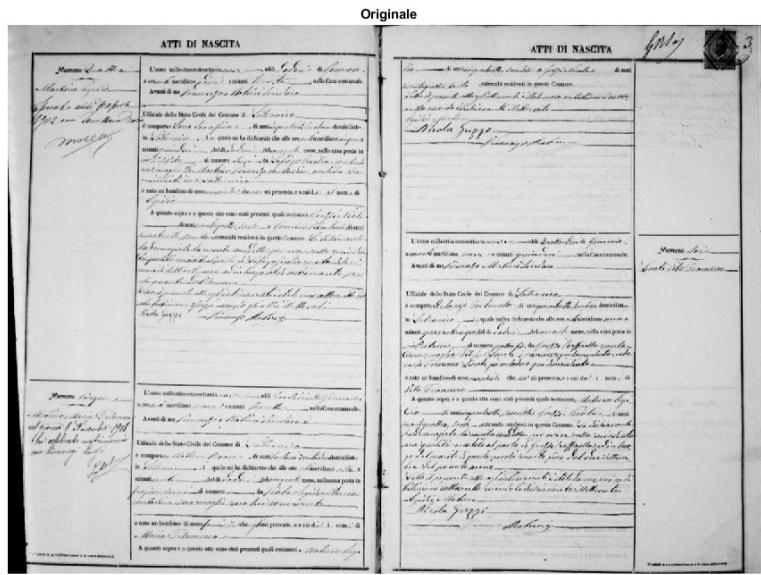


Figura 80: Immagine originale.

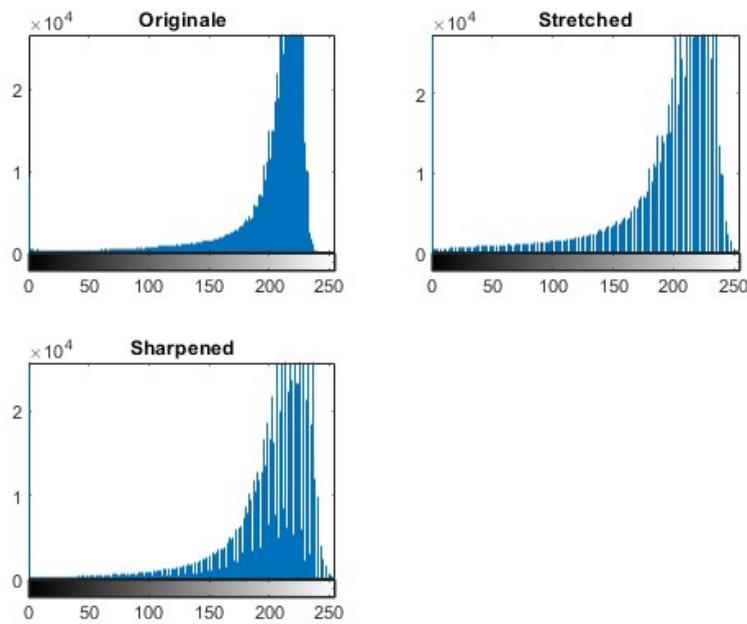


Figura 81: Analisi della scrittura a mano: istogrammi.

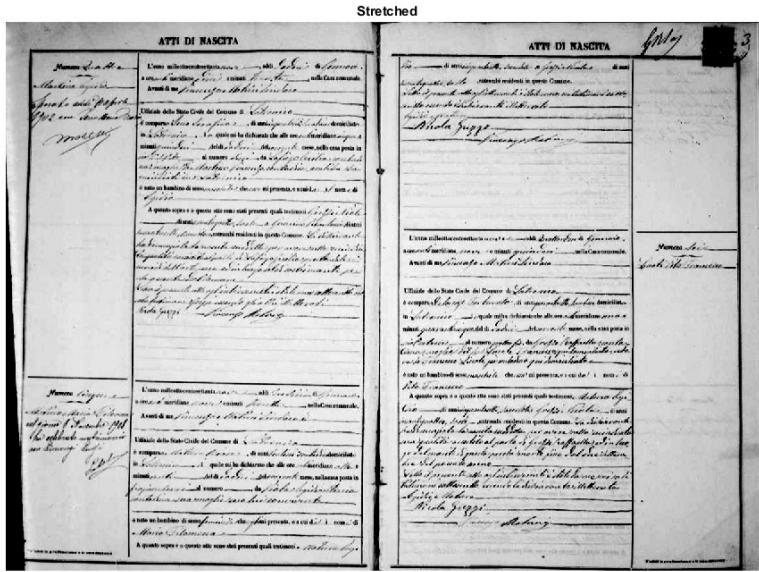


Figura 82: Immagine dopo lo stretching.

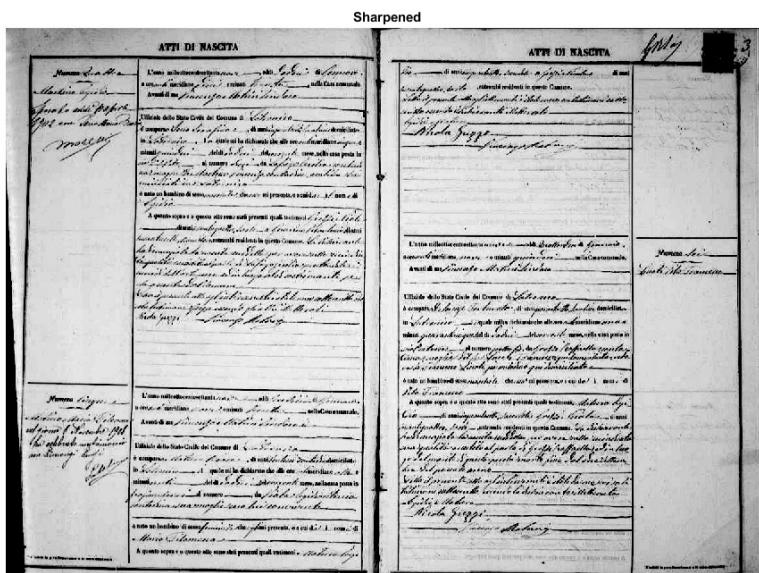


Figura 83: Immagine dopo lo sharpened.

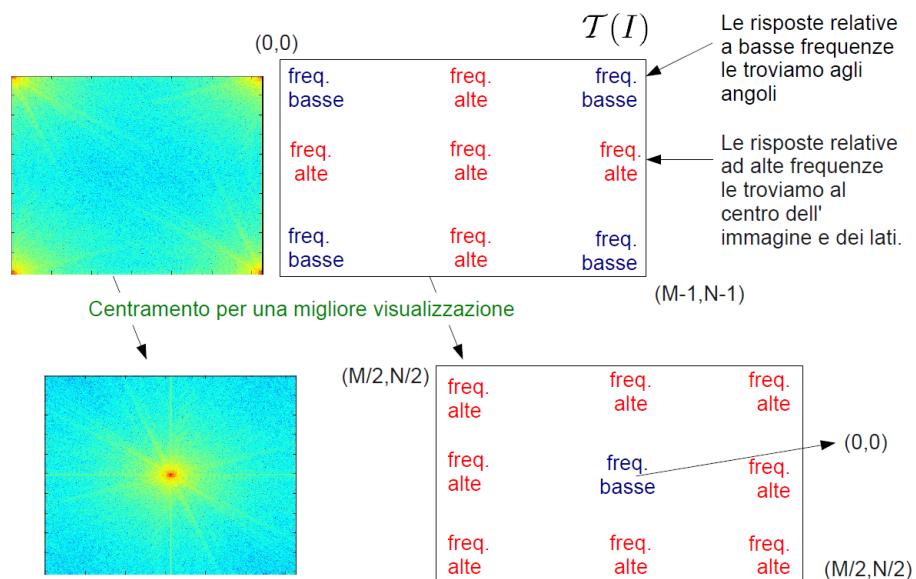
6.8 TdF 2D e trasformata di Hough per rette

6.8.1 Trasformata di Fourier 2D

Ricordando cosa è stato spiegato nel paragrafo delle trasformata di Fourier 1D (par. 6.5), il passaggio da 1D a 2D è semplice. Nella **trasformata di Fourier 2D** l'**analisi** si sviluppa sia in orizzontale che in verticale.

6.8.2 Applicazione della TdF 2D in MATLAB

La **trasformata di Fourier 2D** in MATLAB viene calcolata con il comando: `fft2(image)`. La sua **interpretazione** è spiegata nella seguente immagine:



Come si vede dalla figura, **per migliorare la visualizzazione**, viene effettuato uno shift dello spettro: `fftshift(F)`.

Quindi, in MATLAB, l'analisi di Fourier per un'immagine avviene seguendo i seguenti passaggi:

1. Calcolare la trasformata di Fourier 2D;
2. Visualizzare lo spettro delle ampiezze e lo spettro di fase;
3. Ricostruire l'immagine a partire dalle frequenze.

Un **esempio** in MATLAB:

```
1 %% Esempio -
2 % Trasformata di Fourier discreta 2D in immagini reali ----
3 % In questo esempio vedremo come calcolare la trasformata di
4 % Fourier per un'immagine, visualizzare lo spettro delle ampiezze
5 % e lo spettro di fase e come ricostruire l'immagine a partire
6 % dalle frequenze
7
8 clear all
9 close all
10 clc
11
12 imdata = imread('imageA1.png');
13 imdata = rgb2gray(imdata);
14 figure(1); imshow(imdata), title('Original Image');
15
16 % Calcolo la trasformata di Fourier dell'immagine
17 F = fft2(imdata);
18
19 % FFT dell'immagine: spettro di ampiezza
20 S = abs(F);
21 figure(2); imshow(S,[20 100000]),
22 title('Magnitude Spectrum');
23
24 % Centro lo spettro delle ampiezze per aiutare la visualizzazione
25 Fsh = fftshift(F);
26 figure(3); imshow(abs(Fsh),[20 100000]),
27 title('Centered Magnitude Spectrum');
28
29 % Applico il logaritmo
30 S2 = log10(1+abs(Fsh));
31 figure(4); imshow(S2,[]),
32 title('Log transformed Magnitude Spectrum');
33
34 % FFT dell'immagine: spettro di fase
35 figure(5); imshow(angle(fftshift(F)),[-pi pi]),
36 title('Phase Spectrum');
37
38 % Ricostruzione dell'immagine a partire dalle frequenze
39 F = ifftshift(Fsh);
40 f = ifft2(F);
41 f = real(f);
42 figure(6); imshow(f,[]),
43 title('Reconstructed Image');
```

6.8.3 La trasformata di Hough per rette

La **trasformata di Hough per rette** viene **utilizzata per identificare le rette in un'immagine**. L'idea generale è la seguente: data l'equazione di una retta $y = mx + q$, se si considera lo spazio dei parametri $[m, q]$, una retta è rappresentata da un punto. Quindi, per un punto $[x_i, y_i]$ passano infinite rette, tutte quelle per cui vale $y_i = mx_i + q$. Invece, nello spazio dei parametri queste rette sono punti disposti lungo una retta: $q = -x_i m + y_i$. Nel caso in cui si abbiano solo tre punti allineati, che si trovano lungo una retta, si hanno tre rette nello spazio dei parametri. Inoltre, la retta che contiene tutti e tre i punti si può determinare trovando l'intersezione tra le rette nello spazio $[mq]$.

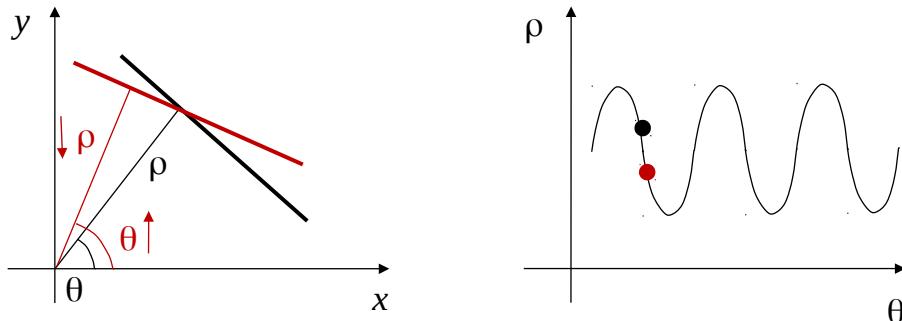
Nella pratica, la trasformata di Hough:

- Quantizza lo spazio dei parametri, ovvero lo divide in celle;
- Ogni punto di *edge* “vota” per un insieme di rette, ovvero vota per un insieme di celle;
- Dopo che tutti i punti hanno votato, è possibile estrarre i massimi. Le rette con più voti vincono, ovvero sono le rette che “spiegano” più punti.

Questo è il motivo per cui viene utilizzata la forma polare per la parametrizzazione. Ovvvero, è più comoda dato che:

- σ è naturalmente delimitato;
- ρ si può delimitare specificando una regione circolare attorno all'origine (sensato).

$$x \cos \theta + y \sin \theta - \rho = 0 \quad (x, y) \rightarrow (\rho, \theta)$$



6.8.4 Dettagli sulla trasformata di Hough e algoritmo

Come **input** l'algoritmo riceve un immagine I in formato binario dove 1 indica *edge* e 0 no *edge*. La dimensione è $M \times N$.

Lo spazio dei parametri è:

$$\begin{aligned}\rho &\in [-\sqrt{M^2 + N^2}, \sqrt{M^2 + N^2}] \\ \theta &\in [-90, 90]\end{aligned}$$

1. Viene eseguita la discretizzazione (ρ, θ) in (ρ_d, θ_d) usando un passo accettabile per il problema da risolvere (precisione a discapito della computazione, o viceversa);
2. Inizializzazione a zero della matrice risultante A di dimensione $R \times T$;
3. Per ogni pixel $I(x, y) = 1$ e per $h = 1, \dots, T$:
 - (a) Viene calcolato $\rho = x \cos(\theta_d(h)) + y \sin(\theta_d(h))$;
 - (b) Viene trovato l'indice k tale che $\rho d(k)$ è l'elemento più vicino a ρ ;
 - (c) Viene incrementato $A(k, h)$ di una unità.
4. Vengono trovati tutti i massimi locali (k_ρ, h_ρ) tale che $A(k_\rho, h_\rho) > \tau$ soglia.

6.8.5 Esercizio 1

- Caricare l'immagine `imageA1.png`
- Visualizzare (con il logaritmo) lo spettro delle ampiezze centrato;
- Estrarre una sola porzione delle frequenze e ricostruire l'immagine a partire da queste frequenze selezionate;
- Osservare cosa succede se cambiano il contenuto in frequenza scelto. In particolare: cosa succede se escluso le basse frequenze? E se escludo le alte?
- Suggerimento per selezionare le frequenze:
 - Creare una matrice, della stessa dimensione della `fft2`, contenente tutti zeri;
 - Copiare nella posizione della porzione considerata le frequenze corrispondenti dalla matrice della `fft2`.

Soluzione

Il codice risultante:

```
1 %% Esercizio 1
2 % - Caricare l'immagine 'imageA1.png'
3 % - Visualizzare lo spettro delle ampiezze centrato
4 % (e visualizzato con il logaritmo),
5 % - Estrarre una sola porzione delle frequenze e ricostruire l'
6 % immagine a
7 % partire da queste frequenze selezionate.
8 % - Osservare cosa succede se cambiamo il contenuto in frequenza
9 % scelto.
10 %
11 % Suggerimento per selezionare le frequenze:
12 % - creare una matrice, della stessa dimensione della fft2,
13 % contenente tutti
14 % zeri;
15 % - copiare nella posizione della porzione considerata le frequenze
16 % corrispondenti dalla matrice della fft2
17 %
18 % Porzioni da considerare (sulla trasformata shiftata):
19 % 1) righe: 260-360, colonne: 410-520
20 % 2) righe: 150-460, colonne: 270-670
21 % 3) righe: 105-310, colonne: 175-460
22 %
23 % Nota: provare anche ad usare getrect per selezionare la porzione:
24 % (La funzione getrect serve per estrarre un rettangolo da un'
25 % immagine,
26 % si veda l'help della funzione)
27
28 clear all
29 close all
30 clc
31
32 imdata = imread('imageA1.png');
33 imdata = rgb2gray(imdata);
34
35 % Calcolo FFT2 e visualizzo lo spettro di ampiezza centrato (con
36 % logaritmo)
37 F = fft2(imdata);
38 Fsh = fftshift(F);
39 figure(1); imshow(log10(1+abs(Fsh)),[]);
40
41 % Selezione la porzione di frequenze.
42
43 % Selezione manuale (decommentare e mettere i valori giusti)
44 % 1) righe: 260-360, colonne: 410-520
45 % 2) righe: 150-460, colonne: 270-670
46 % 3) righe: 105-310, colonne: 175-460
47 % 4) ...
48 rangeX = 260:360;
49 rangeY = 410:520;
50
51 % Selezione con getrect
52 % rect = getrect;
53 % x = int32(rect(2));
```

```

53 % X = int32(rect(2)+rect(4));
54 % rangeX = x:X;
55 % y = int32(rect(1));
56 % Y = int32(rect(1)+rect(3));
57 % rangeY = y:Y;
58
59 % creo un'immagine delle stesse dimensioni della trasformata:
60 croppedSection = zeros(size(Fsh));
61 croppedSection(rangeX,rangeY) = Fsh(rangeX,rangeY);
62
63 figure (2), imshow(log10(1+abs(croppedSection)),[])
64 axis image off
65 colormap gray
66 title('Section')
67
68 % Ricostruisco immagine a partire dalla sola selezione di frequenze
       fatta
69 Test = ifft2(ifftshift(croppedSection));
70
71 % Visualizzo le due immagini
72 figure(3), imshowpair(imdata, real(Test), 'montage')

```

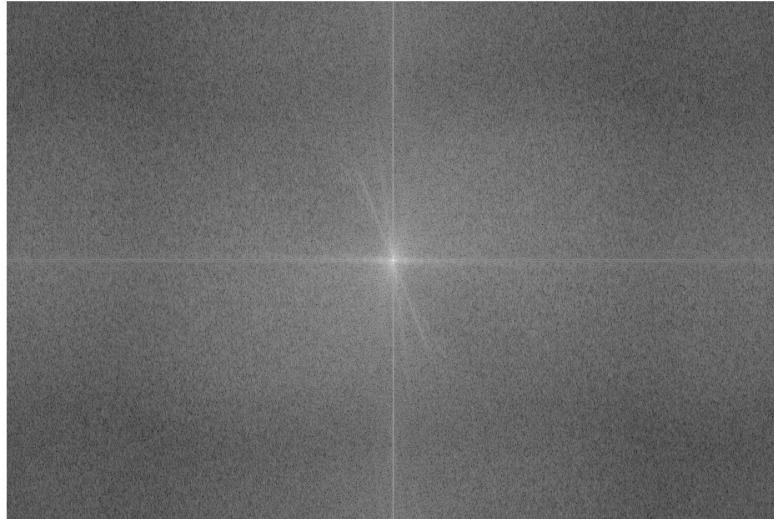


Figura 84: Spettro di ampiezza centrato (con logaritmo).

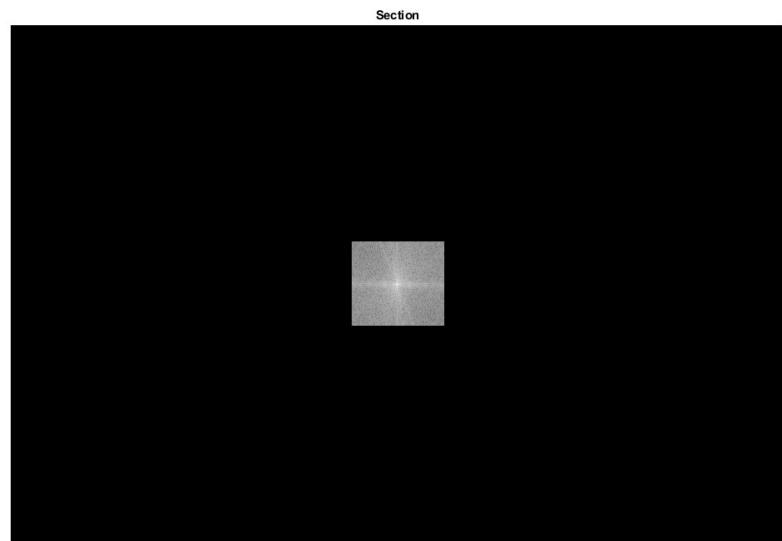


Figura 85: Immagine delle stesse dimensioni della trasformata.



Figura 86: Immagini a confronto dopo le operazioni.

6.8.6 Esercizio 2

Utilizzare la trasformata di Hough per estrarre le rette delle seguenti 3 immagini (una sintetica, due reali): `line.jpg`, `building.jpg`, `Crane.png`. Utilizzare le funzioni MATLAB già implementate per la trasformata (`hough`, `houghpeaks`, `houghlines`). Nota importante: l'input è un'immagine binaria che contiene gli edge, da calcolare sull'immagine originale (per estrarre gli edge utilizzare il Canny edge detector `BW = edge(I, 'canny')`). Visualizzare le rette estrarre sovrapposte all'immagine di partenza e valutare come cambiano i risultati al variare dei parametri della trasformata (numero di picchi, soglia, livello di quantizzazione di ρ e di θ , etc etc – si veda l'help delle funzioni).

Soluzione

Il codice risultante:

```
1 %% Esercizio 2 - Trasformata di Hough
2 %
3 % - Utilizzare la trasformata di Hough per estrarre le rette dalle
4 %   seguenti 3
5 %   immagini (una sintetica, due reali):
6 %   - "line.jpg"
7 %   - "building.jpg"
8 %   - "Crane.png"
9 % - Utilizzare le funzioni Matlab già implementate per la
10 %   trasformata.
11 % Nota importante: l'input è un'immagine binaria che contiene
12 %   gli edge, da
13 %   calcolare sull'immagine originale.
14 % Suggerimento: per estrarre gli edge utilizzare il Canny edge
15 %   detector
16 % BW = edge(I,'canny');
17 % - Visualizzare le rette estrate sovrapposte all'immagine di
18 %   partenza
19 % - Valutare come cambiano i risultati al variare dei parametri
20 %   della trasformata
21 % (numero di picchi, soglia, livello di quantizzazione di rho e
22 %   di theta,
23 %   etc etc -- si veda l'help delle funzioni).
24 %
25 %
26 % TRASFORMATA DI HOUGH in MATLAB
27 % Sono tre le funzioni di riferimento:
28 %
29 % [H,theta,rho] = hough(BW) -> esegue la trasformata di Hough e
30 %   ritorna:
31 %   "rho" = distanza dall'origine fino alla linea lungo una
32 %   retta perpendicolare alla linea stessa
33 %   "theta" = l'angolo tra l'asse x e la retta perpendicolare
34 %   "H" = la matrice di Hough le cui righe e colonne corrispondono ai
35 %   valori
36 % di rho e theta
37 % Per settare una diversa quantizzazione dello spazio utilizzare la
38 % variante:
39 % [H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89);
40 %
41 % peaks = houghpeaks(H,numpeaks) -> identifica i picchi nella
42 %   matrice H
43 % derivante dalla funzione hough. Numpeaks specifica il numero
44 %   massimo di
```

```

34 % picchi da considerare. L'output, "peaks", ritorna una matrice
35 % contenente
36 %
37 % lines = houghlines(BW,theta,rho,peaks) -> estrae le linee nell'
38 % immagine
39 % binaria a partire dalle coordinate dei picchi contenute in peaks
40 % e dai
41 % valori di rho/theta. L'output, "lines", e' una struttura
42 % contenente
43 % informazioni sulle linee estratte.
44 %
45 clear all
46 close all
47 clc
48
49 % Immagine 1
50 I = imread('line.png');
51
52 % Decommentare qui sotto per provare le altre immagini:
53 % Immagine 2
54 % RGB = imread('Crane.png');
55 % I = rgb2gray(RGB);
56 %
57 % Immagine 3
58 % RGB = imread('building.jpeg');
59 % I = rgb2gray(RGB);
60
61
62 % Passo 1. Estraggo gli edge
63 image = edge(I,'canny');
64
65 [M,N] = size(image);
66 figure(1), subplot(121), imshow(I), title('Immagine di Partenza'),
67 axis square
68 subplot(122), imagesc(image), title('Edge - Canny'),axis square
69
70 % 1. Eseguo la trasformata di Hough e la visualizzo
71 [H_M,theta_M,rho_M] = hough(image);
72 figure(2), imagesc(H_M), colormap jet, title('Trasformata di Hough',
73 )
74
75 % 2. Identifico i picchi nella trasformata di Hough
76 peaks = houghpeaks(H_M,5);
77 display(peaks)
78
79 % 3. Estraggo e visualizzo le linee sovrapposte all'immagine di
80 % partenza
81 lines = houghlines(image,theta_M,rho_M,peaks);
82 figure(3), imagesc(I), colormap gray, hold on
83 for k = 1:length(lines)
84     xy = [lines(k).point1; lines(k).point2];
85     plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
86 end

```

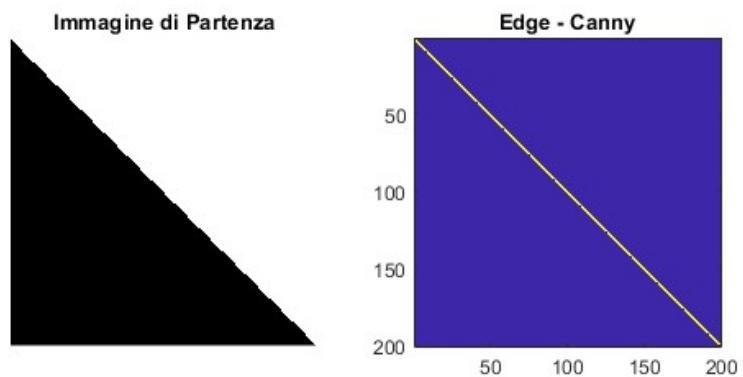


Figura 87: Confronto immagine originale e immagine creata con l'estrazione degli edge (Canny).

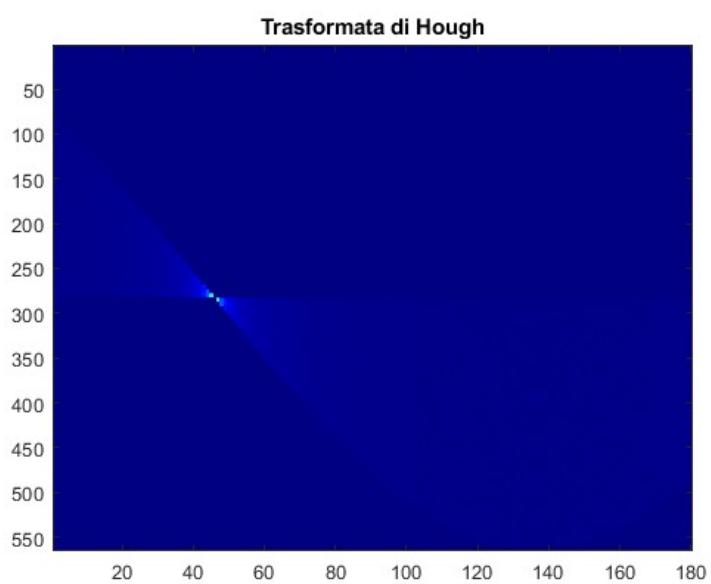


Figura 88: Trasformata di Hough.

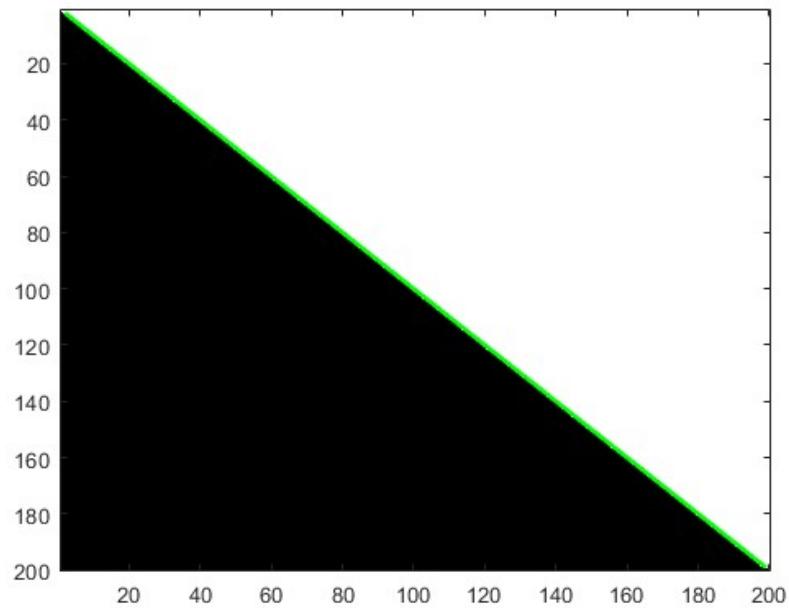


Figura 89: Estrazione delle linee sovrapposte all'immagine di partenza.