

A.A. 2020/21

ELABORAZIONE SEGNALI E IMMAGINI

MARCO CRISTANI

FABS :)

NOTA

Questi appunti/sbobinatura/versione “discorsiva” delle slides sono per mia utilità personale, quindi pur avendole revisionate potrebbero essere ancora presenti typos, commenti/aggiunte personali (che anzi, lascio di proposito) e nel caso peggiore qualche inesattezza!

Comunque spero siano utili! 🌸 ✨

**Questa sbobina fa parte della mia collezione di sbobinature,
che è disponibile (e modificabile!) insieme ad altre in questa repo:**

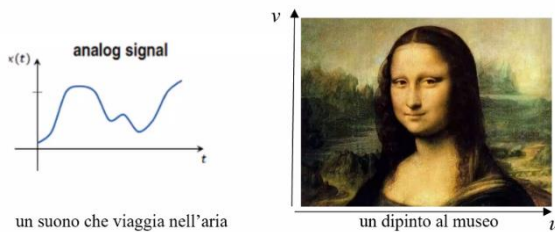
<https://github.com/fabfabretti/sbobinamento-seriale-uniVR>

Table of Contents

NOTA	1
Introduzione	4
» Campionamento	4
» Digitalizzazione	5
» tempo frequenza	6
» Filtraggio	7
Elaborazione delle immagini.....	8
» Qualità di un'immagine	8
» Istogramma	9
» Analisi nel dominio spaziale.....	10
» Rumore nelle immagini	15
» Tipologie di filtraggio	16
» Dominio frequenziale	18
» Rumore nel dominio delle frequenze	22
» Filtri nelle frequenze.....	23
» Filtri passabasso	23
» Filtri PASSA ALTO	24
» Filtri PASSA banda e ferma banda	25

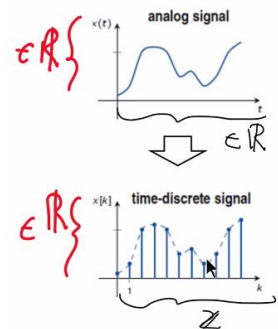
Introduzione

CAMPIONAMENTO



Raccolgo segnali dall'ambiente. Generalmente sono **analogici**; trasportano valori continui, quindi hanno entrambi gli assi continui (=reali).

Il campionamento trasforma un segnale da continuo a discreto.



E' un requisito necessario per fare altre applicazioni come

- **Controllo automatico**
- **Visualizzazione**
- **Memorizzazione**
- **Trasmissione**

Il campionamento deve assicurarsi di **mantenere le proprietà informative** che c'erano nel segnale discreto. Questo non avviene (corruzione, artefatti) in caso di **aliasing**. Se accade o meno è descritto dal **teorema di Nyquist**: se il segnale ha una frequenza massima devo campionare almeno al doppio di questa frequenza.

Il segnale diventa **discreto sull'asse X**. L'asse delle y rimane nell'asse dei reali.

Esempi di segnale

Ci sono pochi segnali discreti nel dominio e continui nel reale, perché normalmente anche il codominio è discreto (e sono detti segnali digitali)

- Un tipo di segnale davvero discreto è il **PAM** (Pulse Amplitude Modulation), che è il segnale che si usa per modulare la luce operata, ad esempio, da un led: vogliamo che il segnale prodotto sia continuo perché altrimenti vedremmo gli scatti man mano che aumentiamo la luminosità.
- **Temperatura misurata**: peschiamo un solo segnale (quindi discreto, faccio n misurazioni)

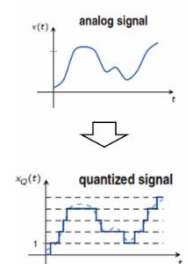
RUMORI PRODOTTI DALL'ALIASING



- > **Immagini**: si creano i **pattern di Moiré**
- > **Suono**: le frequenze più alte (almeno nell'esempio) si **smorzano** e il suono diventa terrificante.
- > **Video**: un video è un segnale nel tempo in cui ad ogni istante ho un'immagine. Quindi, se le immagini non sono catturate a frequenza sufficiente eliminiamo alcune frequenze (quelle alte) e che qui si traducono negli oggetti in movimento

rapido: ad occhio nudo li vediamo bene, mentre in video creano strani effetti per cui gli oggetti si **deformano** o si muovono a scatti.

E' quello che succede sul codominio del segnale; ho **valori discreti sull'asse delle Y**.



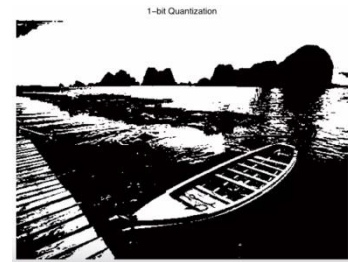
Anche qui ci possono essere problemi relativi a come faccio questa operazione di quantizzazione, ma a differenza della discretizzazione il segnale viene **sempre degradato**. La perdita di info va in logaritmo.

Esempi di segnale

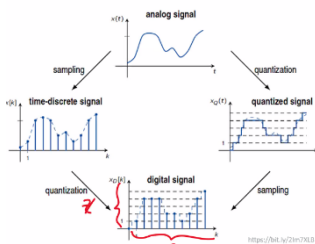
Anche questi sono molto pochi, e di solito si chiamano **segnali di pulsazione**. Sono anche i segnali in uscita da filtri clockless, o commutatori capacitivi.

RUMORI PRODOTTI DA UNA QUANTIZZAZIONE NON ADEGUATA:

- **Audio:** aumenta il noise, lo spettro tende ad appiattirsi.
- **Immagine:** associo ad ogni colore un valore; se ho meno valori avrò meno colori rappresentabili e dovrò “riassumerli” col quanto più vicino.



DIGITALIZZAZIONE



È la composizione di campionatura e quantizzazione. Il segnale risultante è **discreto in dominio e codominio**.

Esempi di segnale

La maggior parte dei segnali con cui interagiamo sono digitali.

La digitalizzazione è essenziale per la trasmissione.

Se ho un segnale continuo (es: strillo :D)

- » Se non faccio niente oltre a urlare, il segnale non arriva altrove.
- » Lo amplifico e basta: chi amplifica non lo sente bene, e viene amplificato in maniera sbagliata (tipo telefono senza fili). L'energia necessaria viene data, ma **l'errore viene mantenuto** e amplificato.
- » Mando un segnale digitale: quando ricevo, non solo sento il segnale ma ho la possibilità di scriverlo su penna, **certificare che il segnale sia corretto** e pooooi rimandarlo.

CODIFICA: Fratello della quantizzazione; quando traduco il segnale non solo scelgo quanti (es. elettrici) ma **simboli informativi** (es. i bit), trasformando un segnale direttamente in un valore numerico. Passo da un livello fisico a un **livello logico** ove ottengo un **numero** che posso utilizzare per la trasmissione o altre applicazioni.

La codifica è fatta in modo tale che il numero di bit sia il **minore possibile**, cosicché al momento della trasmissione sia più leggero possibile.

DECODIFICA: Trasforma il valore in un'altra grandezza e avviene col meccanismo inverso (numero-> colore)

COMPRESSIONE: Serve a ridurre i byte necessari alla memorizzazione del segnale.

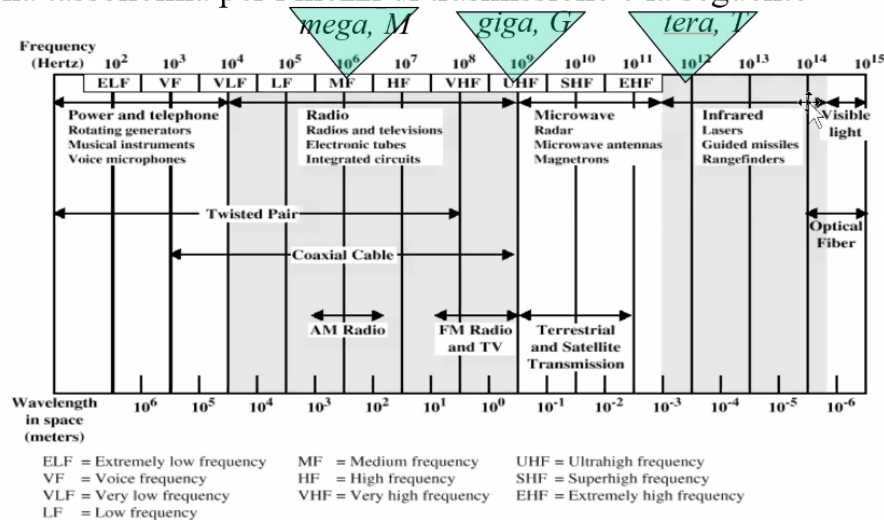
TRASMISSIONE:

- **Guidata:** c'è un mezzo (es. doppino telefonico, cavo elettrico) che determina un cammino fisso
- **In spazio libero:** vuoto, aria, acqua (es. sonar, onda elettromagnetica, wifi)

Il mezzo di trasmissione ha

- **Ampiezza di banda o banda passante (Hz):** freq massima e minima permessa dal mezzo
es. telefono a corona: passava solamente la banda di frequenze relative alla voce. Era un limite per la comunicazione.
- **Velocità di trasmissione o band rate (bit rate):** quante variazioni al secondo riesco a trasmettere.
- **Attenuazione(dB):** quanto il mio mezzo di trasmissione attenua il segnale mano che lo attraversa.
- **Rumore di trasmissione:** segnali aleatori che provengono dall'interno di mezzo di trasmissione (es. rumore termico).

Una tassonomia per i mezzi di trasmissione è la seguente



Una volta che ho il mio segnale in digitale faccio questa analisi. In pratica spiego il segnale nel dominio delle frequenze; gratuitamente vedo il segnale in un altro dominio che mi permette di vederlo meglio e fare più operazioni.

Una volta che ho fatto l'analisi in tf e conosco il segnale da due punti di vista posso filtrarlo.
Il filtraggio serve a:

- MIGLIORAMENTO E RINFORZO



- RESTAURO -> enhancement specifico per segnali video e audio dove cerco di riportare un segnale degradato alla forma originale.
- SEPARAZIONE -> da un segnale ne creo due!
per esempio riesco a rimuovere il noise delle altre persone da una conversazione
- ESTRAZIONE DI CARATTERISTICHE -> da un segnale ne estraggo altri che catturano particolari caratteristiche.
 - > Edge
 - > Regioni
 - > ...

20/11/2020

Elaborazione delle immagini



$f(t)$ $F(\mu)$

$f(t)$ $F(\mu)$

L'elaborazione delle immagini consiste nel **prendere in input un'immagine (=un segnale) e restituirne un'altra (=un altro segnale) in output.**

Per semplificare la comprensione, come esempi prenderemo **immagini a livelli di grigio** (=valori da 0 a 255) che permettono di fare elaborazioni potenti osservando le operazioni dal punto di vista metodologico. Si possono applicare ai colori con modifiche semplici.

Introduciamo (di nuovo) la differenza fra rinforzo e restauro.

RINFORZO

E' l'operazione principale; vuole elaborare un'immagine in modo che il risultato sià **più adatto al compito** in esame.

→ **La definizione è problem oriented**: in base alla situazione sono interessato a un obiettivo diverso!

RESTAURO

Ho un **modello di una immagine a cui devo mirare** (ad esempio immagine molto vecchia con delle crepe; vorrò mirare all'immagine come era all'inizio).

QUALITÀ DI UN'IMMAGINE

La qualità di un'immagine è una **combinazione pesata di più caratteristiche**, dove ciascuna ha un peso a seconda della sensibilità dell'utente; la "versione migliore" è soggettiva, e ci sono delle macro ma non sono assolute.

In genere, qualità significa tutto e niente:



Più verosimile ma ha pochi dettagli



Per nulla verosimile ma mi mostra molti più dettagli.

E' più semplice definire quanto NON è di qualità: un'immagine è NON di qualità **quando non riesce ad essere interpretata facilmente da un operatore umano** – come, ad esempio, un'immagine molto soggetta a rumore, tanto da non riuscire ad essere interpretata (= non si vede niente).



Esempio: riconosco i pattern ma non riesco a fare delle interpretazioni elaborate – a posteriori mi rendo conto che c'è un cane, ma non è facilmente interpretabile.

ISTOGRAMMA

E' importante in quanto si usa anche a livello professionale per fare rinforzo sulle immagini. L'idea è **che i pixel di un'immagine sono una popolazione statistica su cui posso calcolare quantità statistiche** (media, varianza...)

L'istogramma è una **rappresentazione statistica dei pixel visti come popolazione**.

Può essere visto come una **funzione continua o discreta H**, dove per ogni livello di grigio riportiamo il numero di pixel di quel colore.

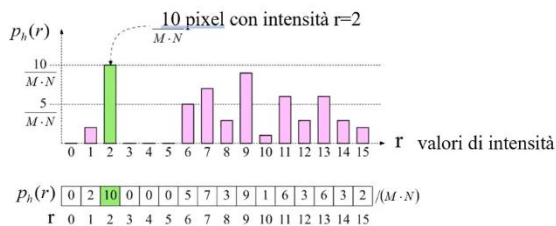


Per un'immagine $I[M, N]$ si ha che $H(r) =$
numero di pixel di valore r
con $0 \leq r \leq L - 1$ e dove $\sum_{r=0}^{L-1} H(r) = M \cdot N$

Intuitivamente, l'istogramma è una snapshot che in un attimo, dà l'idea del contenuto cromatico dell'immagine.

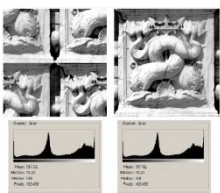
ISTOGRAMMA COME DISTRIBUZIONE DI PROBABILITÀ

Per certe applicazioni risulta comodo vedere l'istogramma come una **distribuzione di probabilità**:



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

Per arrivare a questa divisione mi basta **dividere ciascuno dei bucket per l'area**.



Attenzione: alcune operazioni potrebbero portarmi a non vedere alcun cambiamento nell'istogramma!

Disponendo i pixel in modo diverso senza modificarli l'istogramma non cambia.

Le operazioni che faremo operano sempre sui livelli di grigio, quindi per nostra fortuna vedremo sempre i cambiamenti sull'istogramma.

! Per ambiti come il machine learning, l'istogramma non è certamente sufficiente.

Quello che è davvero catturato bene dall'istogramma è il contrasto, ovvero il rapporto o la differenza fra i valori più bassi e più alti.

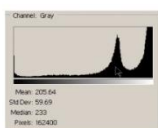


Immagine chiara o sovraesposta:
istogramma con **valori più alti a destra**, e una coda che si **concentra sui valori più chiari**.

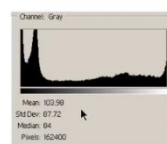


Immagine scura o sottoesposta:
istogramma con **valori più alti a sinistra**

NB:

Chiaro/scuro: caratteristica **dell'immagine**

Sovraesposizione/sottoesposizione: caratteristica del **metodo di acquisizione**

ANALISI NEL DOMINIO SPAZIALE

Nel dominio spaziale possiamo esprimere l'elaborazione delle immagini come

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

dove f è l'immagine di ingresso, g l'immagine in uscita e T un operatore su f definito in un intorno di x, y .

Si identificano tre tipi principali di elaborazioni, dipendenti dall'intorno che si prende in considerazione:

- » **Puntuale:** l'intorno coincide con il **pixel stesso**
- » **Locale:** l'intorno rappresenta una **regione (per esempio quadrata) attorno al pixel** di locazione
- » **Globale:** l'intorno rappresenta **l'intera immagine**.

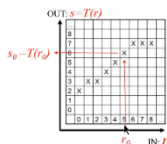
OPERAZIONI PUNTUALI

Si dice **operatore puntuale** un operatore che preso in input il valore di un pixel dà il valore cambiato (ed esso dipende esclusivamente da quello stesso valore).

Un **operatore puntuale** può essere rappresentato da una **funzione** che preso in input un valore r lo modifica in un valore $s=T(r)$ con s, r appartenenti allo stesso campo di definizione

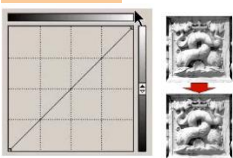
$$T: [0, L-1] \subset \mathbb{R} \rightarrow [0, L-1] \subset \mathbb{R}$$

Poiché un operatore puntuale dipende solo dal valore del pixel posso descriverlo completamente con la seguente tabella, detta **Look Up Table**:



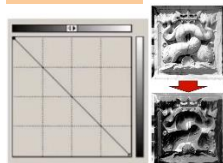
r	0	1	2	3	4	5	6	...
s	$T(0)$	$T(1)$	$T(2)$	$T(3)$	$T(4)$	$T(5)$	$T(6)$...

IDENTITÀ



Chiaramente **non fa nulla** :)

NEGATIVO



$$s = L - 1 - r$$

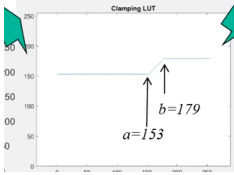
max del dominio

Prende il livello massimo meno uno e vi sottrae il valore del pixel.

E' usata spesso in ambito di **segnali biomedici**, in modo tale che se ho la maggior parte di valori di grigio scuri essi diventano chiari – esempio: le tac.

Funziona perché a livello umano si percepisce meglio il nero sul bianco che il bianco sul nero, quindi risulta più chiara.

CLAMPING O CLIPPING



Fun fact: se noto che ci sono picchi sull'ultimo valore minimo/massimo, è facile che ci sia stata un'elaborazione. Dead giveaway xD

Limita le intensità ad un range definito.

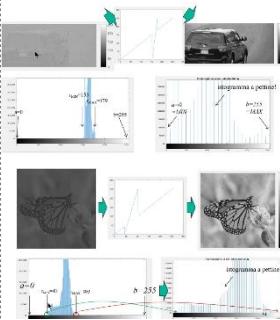
Serve perché se ho un'immagine da riportare a un intervallo di quantizzazione (es. posso diminuire i bit della codifica se mi serve). Questa è un'operazione che posso fare per risolvere il problema del rinforzo, perché riduce o elimina il rumore impulsivo.

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

All'immagine ho aggiunto pixel neri e bianchi di rumore. Il clamping mi risolve! Sostituisco i disturbi con qualcosa di **percettivamente meno visibile**.

STRETCHING O SHRINKING

(Il nome cambia in base alla parametrizzazione)



Permette di stirare o comprimere l'intensità da un range [rmin,rmax] ad un range in [a,b].

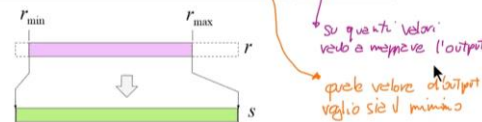
E' la più duttile fra quelle lineari, e sono io a definire tutti e quattro i parametri.

- > R_{max}, R_{min} : il più grande e piccolo valore del range che voglio trattare
- > a, b : sono il massimo e minimo stretchati.

A fine operazione farò un rounding.

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

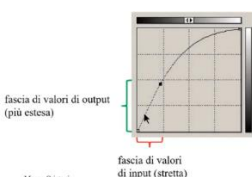
dove r_{min}/r_{max} sono il più grande/piccolo l'dg del range che voglio trattare, b e a sono il massimo e il minimo stretchati. L'operazione e' seguita da rounding nel caso di dominio di valori nei naturali (come in 0-255)



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

!! Lo stretching non risolve il problema del rumore impulsivo

TRASFORMAZIONE LOGARITMICA



$$s = c \log(1 + r)$$

$$\text{con } r \in [0, L-1] \wedge c = \frac{L-1}{\log(L)}$$

C è la **costante di normalizzazione** che mi assicura la mappatura in $[0, L-1]$.

L'aggiunta di 1 permette di evitare il calcolo di quantità fra 0 e 1, che genererebbero valori < 0 (e il calcolo di $\log(0)$ va a $-\infty$ che non ci piace)

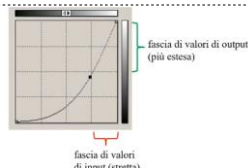
Viene usata quando voglio **mappare fasce strette di valori nell'immagine originale in fasce più ampie**, aumentandone così il range del contrasto e rendendo l'interpretazione umana più informativa. Espande tanto i valori in un certo range: sacrifici il range chiaro per "ampliare" il range scuro!!

TRASFORMAZIONE LOGARITMICA INVERSA

$$s = (e^r)^c - 1$$

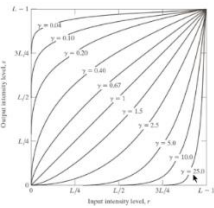
$$r \in [0, L-1]$$

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



fa l'opposto obv

TRASFORMAZIONE DI POTENZA



$S = cr^\gamma$ Potresti modulare le ultime due come casi particolari di questa.

- > Per gamma < 1 è analoga alla logaritmica
- > Per gamma > 1 è diversa

Il termine di normalizzazione c è complicato da definire quindi definiamo due versioni della trasformazione: normalizzata e non normalizzata.

Si preferisce fare calcolare s senza normalizzazione, ovvero senza c , ottenendo valori che stanno fuori dal range 0-255; poi li normalizzo stretchando linearmente all'interno del range giusto.

$$\tilde{s} = r^\gamma$$

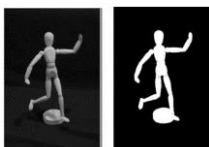
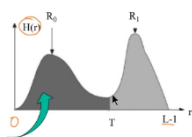
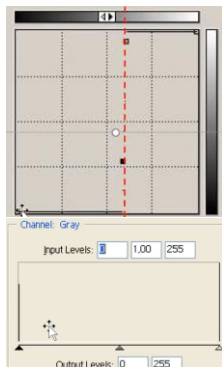
$$\downarrow$$

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

$\tilde{s}_{max}, \tilde{s}_{min}$ sono il massimo e minimo livello di grigio di \tilde{s}
MAX e MIN sono i valori massimi rappresentabili alla fine

Le operazioni di potenza servono anche a fare i monitor → gamma correction di Giakets: su un monitor CRT si può applicare una correzione pre-processando l'input con la corrispondente funzione inversa.

BINARIZZAZIONE



Produce un'immagine che ha solo due livelli: bianco e nero.

Si ottiene scegliendo una certa soglia T , e mettendo a nero tutto ciò che è inferiore e a bianco tutto ciò che è superiore.

Di solito è usata per fare la segmentazione di un oggetto che si trova su uno sfondo semplice e non texturizzato. E' molto usata in ambito biomedicale e di videosorveglianza.

La difficoltà risiede nel trovare la soglia T ragionevole! Quello che faccio è calcolare una funzione costo per ciascun T possibile e prendere il minore.

METODO DI OTSU

Assume che ci siano due regioni ben distinte da scegliere.

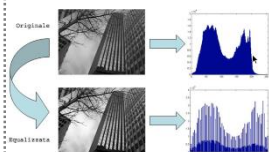
Istogramma visto come distribuzione di probabilità di avere un pixel nella regione che sta fra 0 e T .	$p_{0 \rightarrow T}(r) = \frac{H(r)}{\sum_{r=0}^T H(r)}$	E' la controparte probabilistica dell'istogramma ma sul sottoistogramma < T
Varianza	$\sigma_{0 \rightarrow T}^2 = \sum_{r=0}^T p_{0 \rightarrow T}(r) (r - \mu_{0 \rightarrow T})^2$	Usiamo la forma biased, e la calcoliamo con i pesi.
Media	$\mu_{0 \rightarrow T} = \sum_{r=0}^T r \cdot p_{0 \rightarrow T}(r)$	Pi non è uniforme ma pesata (dall'istogramma), quindi la media diventa media pesata.

Con la binarizzazione voglio minimizzare le formule appena viste: voglio minimizzare una quantità composta da due fattori indicizzati su 0 e 1, prendendo in considerazione gli oggetti visti nelle slides precedenti.

$$\sigma_w^2(T) = W_B(T) \sigma_B^2(T) + W_F(T) \sigma_F^2(T)$$

$$W_B(T) = \sum_{r=0}^{T-1} p(r), \quad W_F(T) = \sum_{r=T}^{L-1} p(r), \quad p(r) = \frac{1}{M \cdot N} \sum_{r=0}^{T-1} H(r)$$

EQUALIZZAZIONE



Viene venduta come un'operazione che "migliora" la qualità dell'immagine, quantomeno dal punto di vista esclusivamente percettivo. → Operazione di **rinforzo** che migliora la qualità percettiva.

1. Migliorare qualità percettiva
2. Approssimare l'istogramma con uno uniforme

L'obiettivo è quello di rendere massimamente simile l'istogramma a un istogramma uniforme: deduciamo che l'istogramma uniforme è quello

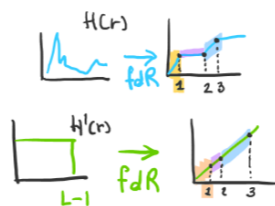
percettivamente migliore, poiché ogni intensità di grigio ha un egual valore, quindi il cervello ha la massima gamma di valori di grigio percettibili; quindi il cervello può elaborare un dataset più informativo. Se fosse concentrato in una piccola area, tutto il contenuto dell'immagine mi darebbe meno simboli per codificare la mia immagine.

L'equalizzazione rende la funzione di ripartizione dell'istogramma più simile alla funzione di ripartizione dell'istogramma uniforme.

Vorremmo arrivare a ciò, ma così distruggerei il contenuto informativo dell'immagine!

Risolvere con 2 trasformazioni:

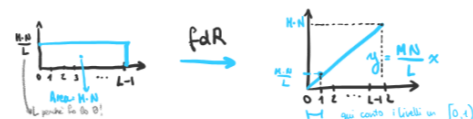
▷ **FUNZIONE DI RIPARTIZIONE:** sommo cumulativamente $H(r)$



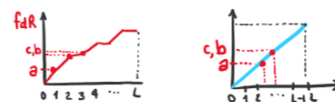
L'idea è di ricollocare alcuni punti di sx su dx o seconda delle fdr.

- **CASO ①:** se le due fdr hanno andamento simile non fa nulla.
- **CASO ②:** è meno INCLINATO di quello lineare, dunque lo mappa PIÙ INDIETRO
- **CASO ③:** è PIÙ INCLINATO di quello lineare, dunque lo mappa PIÙ AVANTI.

▷ **HAPPING** Assumiamo che $AS = M \cdot N$ pixel



Con l'equalizzazione mappa le quote sulla retta



- **a:** cerca dove è la a sulle fdr "normale"
- **b, c:** ho poco spazio! In 06 ho 2 livelli di grigio che mappano su una quota simile. Nel mapping, mappano alla stessa quota che mi dava lo stesso CDF di grigio

→ Rindatterò 06 in modo che 2,3 si comportino come nel grafico equalizzato, ovvero mappino sullo stesso livello di grigio!

HAPPING DA COMPORTAMENTO

$$x = \frac{L}{MN} \cdot y_r$$

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k P_r(r_k) = \frac{L-1}{MN} \sum_{j=0}^k H(r_j) = \frac{\sum_{j=0}^k H(r_j)}{\frac{MN}{L-1}}$$

ALGORITMO DI EQUALIZZAZIONE

1. Calcolo delle **L somme cumulative** dei valori dell'istogramma visto come distribuzione
2. Moltiplicazione dei valori per il **valore massimo di grigio -1**
3. Normalizzazione dei valori calcolati al passo 1 **dividendo per il numero totale di pixel MN**
4. **Arrotondamento**
5. **Applicazione del mapping T ottenuto.**

Per lo stesso principio, se io ho un istogramma che ritengo essere ancora più bello nessuno mi vieta di usare quell'istogramma al posto di quello normale!

Significa che quando voglio fare una sostituzione di grigio in una certa posizione x,y , il **numero che produco e sostituisco dipende da un calcolo che prende in considerazione il valore di grigio precedente e anche il valore di altri pixel in una zona vicina** che posso parametrizzare – per esempio, una cella quadrata 3×3 intorno al pixel originale, o dei pixel sparsi intorno a quello in x,y .

Sono utilizzate in operazioni più potenti, che permettono di estrarre più informazioni rispetto alle puntuali (dato che analizzano più pixels).

Per esempio, con le operazioni locali riesco a **estrarre gli edge**. Di solito, queste operazioni si realizzano con una convoluzione con una **matrice** detta **maschera** o **kernel**.

Gli intornoi sono di grandezza $k \times k$ (con k dispari), e possono essere sia densi (=tutti i pixel del vicinato) che sparsi. I pixel al di fuori dell'intorno non sono presi in considerazione.

Pseudocodice:

Input:	<ul style="list-style-type: none"> » Immagine f definita con un suo valore di pixel generico $f(n, m)$ » Intorno di valori di pixel definito come $[f(n, m)]$ ovvero $f(n - u, m - u)$ con $(u, v) \in \left[-\frac{K-1}{2} \dots \frac{K-1}{2}\right] \times \left[-\frac{K-1}{2} \dots \frac{K-1}{2}\right]$ con K dispari
Code:	<pre>for n=1..N for m = 1..M g(n,m) = T([f(n,m)])</pre>
Output:	» Nuova immagine $g(n, m)$

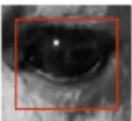

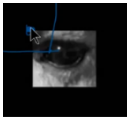
Tipologie di filtraggio:

Filtraggio lineare T può essere visto come combinazione lineare dei valori di pixel nel vicinato, come la convoluzione: è complicata ma sono solo shift e moltiplicazioni!	Filtraggio non lineare Non è una combinazione lineare pesata: ci sono operazioni non lineari, quindi non riesco a srotolarla e metterla su una matrice. Ad esempio, il mediano: <ul style="list-style-type: none"> - Max dei pixel - Mediano: devo prendere i vicini, riordinarli e prendere quello in mezzo -> non riesco a esprimerlo in combinazione lineare
--	--

Problema dei bordi

Se voglio prendere un pixel che si trova sul bordo mi manca un pezzo!

Ci sono tre alternative:

Crop	Extend	0 padding
 <p>Limito la zona perdendo dei valori di pixel dell'immagine di partenza in quanto non possono essere filtrati</p>	 <p>Prendo i valori sulla frontiera e li replico.</p>	 <p>Metto una cornice nera attorno all'immagine. Ma viene fuori una schifezza sfumata col nero.</p>

RUMORE NELLE IMMAGINI

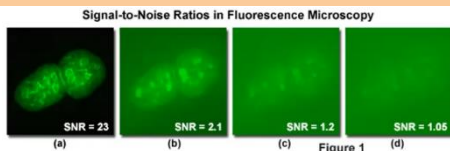
Il rumore nelle immagini è la **degradazione/corruzione del contenuto informativo**.

E' tipicamente modellato come additivo o moltiplicativo e casuale o causale. Noi vedremo additivi e casuali.

I rumori additivi e casuali significano che possiamo vedere l'immagine affetta da rumore come una funzione dove alla funzione pulita ho aggiunto una funzione rumore, dove la funzione rumore è frutto di un processo aleatorio ed è indipendentemente da dove è collocato il processo.

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

SIGNAL TO NOISE RATIO – SNR



La quantità di rumore può essere stimata attraverso la misura di signal to noise ration (SNR), di cui esistono numerose versioni.

- » **Mean square:** fai la somma di tutti i pixel quadrati e fai la stessa cosa sottraendo l'immagine di partenza

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

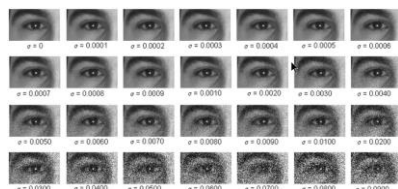
Se ho un'immagine molto simile a quella di partenza, sotto avrò qualcosa di vicino a zero: quindi più il valore molto alto significa immagine pulita.

- » **Deviazione standard:**

Anche qui, il rumore aumenta la varianza del segnale e il segnale pulito è quello co SNR grande. Questo motiva anche l'utilizzo di immagini ad alto contrasto: σ_s risulta maggiore.

$$SNR_{\sigma} = \frac{\sigma_s}{\sigma_n}$$

RUMORE GAUSSIANO ADDITIVO BIANCO



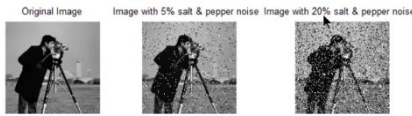
E' un rumore aleatorio, dunque descritto da una probabilità di avere rumore in una certa posizione:

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

Ovvero, segue una distribuzione gaussiana con $\mu = 0$ e una particolare σ^2 tale per cui più la varianza è alta, più i numeri prodotti saranno distanti da 0, e dunque più sarà rumorosa l'immagine finale. σ^2 , di fatto è la forza della distribuzione. Il numero può essere sia positivo che negativo

RUMORE IMPULSIVO

- Selezione D% pixel in maniera uniforme nell'immagine
- Per ognuno di essi assegna il valore min o max con probabilità uniforme $p=0.5$



Può sostituire 0 o 255 a ciascun pixel con distribuzione uniforme; inoltre ho una percentuale di pixel sulla quale posso fare questa modifica (es. il 50% dei pixel; su ciascuno di quel 50% tiro una moneta per decidere se lo sporco o meno).

TIPOLOGIE DI FILTRAGGIO

FITRI DI SMOOTHING

Aumentano il SNR, ossia tolgono il rumore.

FILTRO MEDIA

- Serve a rimuovere **rumore gaussiano**.
- È un filtraggio **T lineare** (=combinazioze lineare di coefficienti e valori dei pixel)
- Si attua come **convoluzione dell'immagine con la maschera media**, la quale ha le seguenti caratteristiche:

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

- Dimensioni **KxK** con **K dispari**
 - Coefficienti tutti uguali e pari a **$1/K^2$** , che è il numero di valori presente nella media
 - La **somma** dei valori del filtro deve essere **1**.
- Questo significa che la combinazione lineare è **convessa**, ovvero che non sto sommando/sottraendo valore di grigio ma lo sto solo ridistribuendo (a meno di padding).

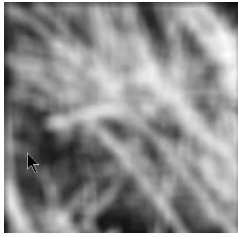
Dal fatto che il filtro è convesso deriva che dopo il filtraggio di media ottengo come output un'immagine interpretabile "simile" a quella di partenza.. ovvero, l'output non è una mappa ma un'immagine. Questi output visivamente/percettivamente non saranno più chiari o più scuri.

Funziona perché dato un intorno di applicazione $[n,m]$, esso calcola la media dei valori compresi e lo sostituisce. Dato che il rumore gaussiano ha media = 0, in generale dovrebbe riuscire a compensarlo. Matematicamente:

$$\begin{aligned}
 \underset{\substack{\text{img filtrata} \\ \downarrow}}{g(n,m)} &= T \left[\underset{\substack{\text{img con noise} \\ \uparrow}}{\tilde{f}(n,m)} \right] = \underset{\substack{\text{Def di img} \\ \text{affetti da noise}}}{\underset{\substack{\text{ASPETTATIVA} \left(\begin{smallmatrix} \text{in statistica} \\ \text{E' la media} \end{smallmatrix} \right)}}{E[\tilde{f}(n,m)]}} \\
 &= E[(f(n,m) + \varepsilon(n,m))] \\
 &= E[f(n,m)] + E[\varepsilon(n,m)] \\
 &= E[f(n,m)] = \underset{\substack{\mu = 0 \text{ per def} \\ \mu(f(n,m))}}{\mu(f(n,m))}
 \end{aligned}$$

L'inghippo è che il valore atteso viene **approssimato** dalla media! Non è la media perfetta! Funzionerebbe se avessimo infiniti termini su cui fare questa operazione. *QUESTA DOMANDA È UNA TIPICA DOMANDINA DI TEORIA*

E se io prendessi un insieme di valori più grandi? Non dovrei avere stimatori migliori? Ebbene sì, ma più grande è il supporto del filtro, più mi avvicino ad avere $\epsilon = 0$, ma sto mediando l'immagine con i suoi dintorni -> sarà sempre più sfocata.



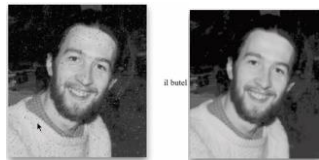
Problema: artefatti

Vedo degli artefatti quadratinosi! Questo succede perché il filtro di media può essere visto come un **filtro ideale taglia frequenza** (=box), che dove esegue una certa operazione in frequenza nello spazio (**sinc**) sto moltiplicando l'immagine per una sinc, i cui ripple si ripercuotono nell'immagine. Si capirà meglio con il filtraggio in frequenza.

FILTRO MEDIANA

- Serve per il **rumore impulsivo**.
- È un filtraggio **NON lineare**, che si realizza attraverso un algoritmo: riordino i valori e prendo quello centrale.

È utile poiché la media è una statistica che soffre di outlier: una parte molto piccola ma molto diversa rischia di sballare il risultato. Questo non accade con la median, che è una **stima robusta della media**.



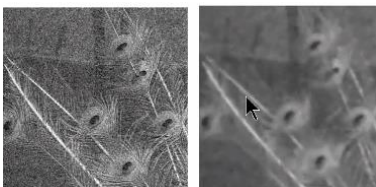
FILTRO GAUSSIANO

- È una **media pesata** dove i pesi vicino al centro hanno un peso maggiore.
- Anche questa è una **combinazione lineare convessa**. Devo **parametrizzarlo sulla deviazione standard; vedendolo in elevazione voglio approssimare la campana gaussiana**

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

$\frac{1}{273}$

Il valore centrale è più alto perché così evito di “tirare dentro” fortemente cose troppo lontane (e quindi di mischiare insieme l'informazione). Il valore finale sarà più simile al valore precedente al filtraggio.



Agisce in maniera meno pesante della media, ovvero esegue uno smoothing ma non abbatte tutti i dettagli come la media.

Risulta molto efficiente in quanto la maschera è separabile ovvero posso applicarla prima tutta sulle righe e poi tutta sulle colonne.

Definizione appropriata dei parametri σ e W

σ = deviazione standard

W = dimensione della maschera 1D

Si fissi σ e si trovi la dimensione della maschera W sapendo che W deve essere tale da contenere un'elevata percentuale di probabilità (= area della densità gaussiana, vedasi figura). In particolare, la statistica dice che con $W=5\sigma$ copro il 98.76% dell'area della densità Gaussiana



In pratica, immaginando che la dimensione W sia il mio “limite massimo” visualizzabile voglio che tutta la campana sia entro la maschera (ma ovviamente non troppo stretta).

Per cui, posso dire che selezionando $W = 3$ o $W = 5$, voglio determinare il valore della varianza in modo da avere valori prossimi allo zero.

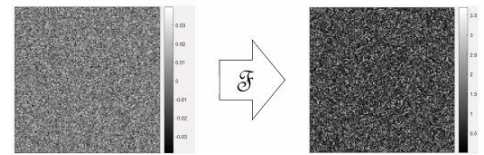
Se non faccio attenzione, rischio di avere una varianza troppo grande che verrà tagliata fuori. In questo modo renderò il mio filtraggio sempre più simile alla media, lol.

DOMINIO FREQUENZIALE

FILTRAGGI DI SHARPENING O DI DERIVATA

Servono ad evidenziare i dettagli, o come post processing dopo filtraggio di smoothing.

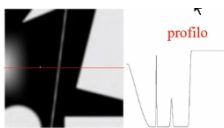
I dettagli sono ad alta frequenza, come il rumore, dunque possono anche incrementare il rumore.



Ne abbiamo due categorie:

- **Basic highpass spatial filtering**
→ filtro con Laplaciano Filtraggio lineare con il laplaciano. La maschera H è caratterizzata da coefficienti positivi vicino al centro e negativi nella periferia esterna.
- **High boost filtering** = mano a mano che taglio a frequenza più alte esso diventa più acceso

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



Un **edge** è una **discontinuità ad alta frequenza**: i filtri di sharpening esaltano queste discontinuità.

Estraiamo il profilo sulla riga: i contrasti sono chiaramente in corrispondenza dei picchi!

In pratica:

DERIVATA PRIMA

Derivata asimmetrica (=il filtro non è simmetrico)

Derivata asimmetrica

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

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

Filtro differenziale
asimmetrico

$$\partial_x = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

più piccolo intervallo possibile, ovvero $h=1$.

Per fare questa operazione in modo veloce uso una minimaschera di convoluzione $\begin{bmatrix} -1 & 1 \end{bmatrix}$ (al successivo togli il precedente).

Analizziamo l'immagine in x,y (coordinate del pixel). Questo è nell'ambito continuo: in ambito digitale, non possiamo spostarci di un $h \rightarrow 0$ (perché al minimo mi sposto di 1 pixel). Quello che facciamo è tradurre la formula di conseguenza: facciamo quest'operazione nel

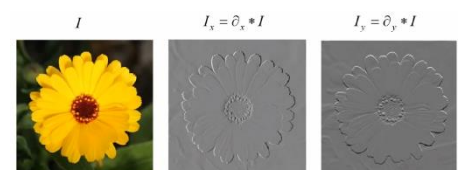
Derivata simmetrica

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

Fa la stessa operazione ma su un intorno di 3 pixel.

Il risultato di queste operazioni non è più un'immagine ma una mappa.

In base a cosa le discontinuità sono bianche/nere? Viene dal fatto che io sottraggo il primo valore di grigio al secondo: nella prima img noi passiamo da scuro a chiaro, quindi ottengo uno schiarimento.



DERIVATA SECONDA

Sarebbe il filtro Laplaciano. È isotropico, ovvero invariante a rotazione.

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

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

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



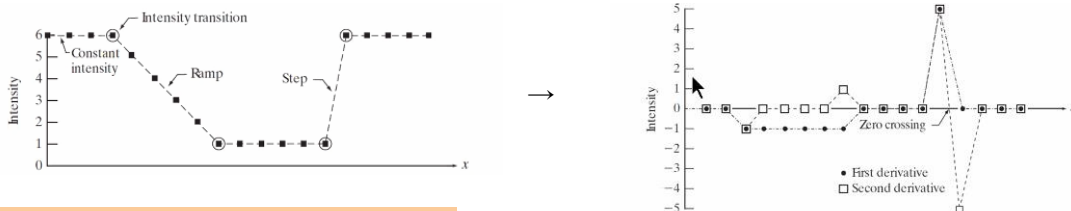
- Nulla in regioni con intensità costante o variazione costante (rampe)
- Non nulla in presenza di discontinuità: **ottengo una mapa di edge!**
- Di fatto misura la **differenza di coefficiente angolare**.

Prende il pixel prima e il pixel dopo sulla stessa linea e fa l'operazione. Poi fa lo stesso sulle x. È esprimibile come convoluzione.

È molto usato perché di solito non mi interessa sapere *quanto* sia la variazione (derivata prima), ma *dove* sia.

CONFRONTO

Entrambe mi servono per analizzare le discontinuità;



BASIC HIGHPASS SPATIAL FILTERING

Fa il filtraggio lineare con il laplaciano, ovvero una maschera caratterizzata da coefficienti positivi al centro e negativi in periferia.

- **Somma coefficienti a 0: quando il filtro passa su regioni con livelli di grigio stabili o quasi, l'output è 0 o quasi**
- L'uscita è alta quanto il valore centrale differisce dalla periferia
- L'output p una mappa e mostra i bordi

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

Matlab mette a disposizione una funzione apposita che prende l'immagine originale e le somma il risultato della mappa di convoluzione amplificato (posso decidere quanto).

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

HIGH BOOST FILTERING

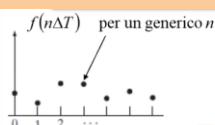
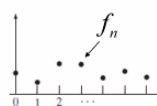
Stessa identica cosa, ma anziché amplificare la mappa di edge amplifica la mappa originale.



DFT 1D

Ripasso: DFT 1D

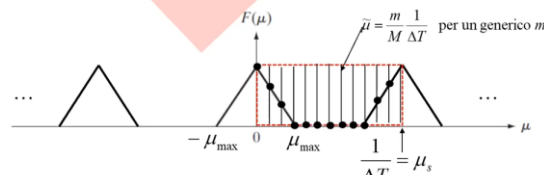
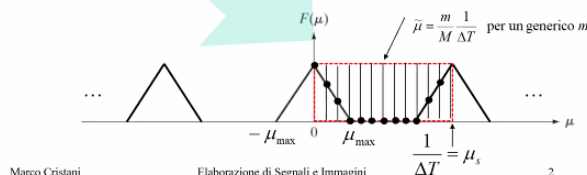
- Prima di passare al filtraggio in frequenza, un richiamo alla DFT



TRASFORMATA DI FOURIER DISCRETA INVERSA

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

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



Marco Cristani

Elaborazione di Segnali e Immagini

2

DFT 2D

Quello che faccio, di fatto, è un semplice cambio di variabile: là dove parlavo di tempo passo allo spazio.

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

Corrisponde a prendere una riga dall'immagine e chiamarla f(x). Passo al 2D aggiungendo y (spazio) e v (colonne) per l'analisi in verticale.

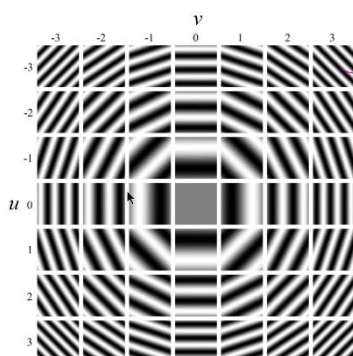
$$F_{(u,v)} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi \left(\frac{u}{M} x + \frac{v}{N} y \right)}$$

$$= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) B^{(u,v)}(x,y)$$

+ inversa:

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

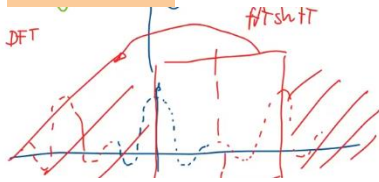
Più u,v sono alti, più definiscono frequenze alte e più definiscono variazioni brusche: possiamo vedere B come un'immagine che sta nei complessi, e che può a sua volta essere visualizzata come un piccolo kernel che è il tipo di contrasto "rilevato" da quella coppia di u,v.



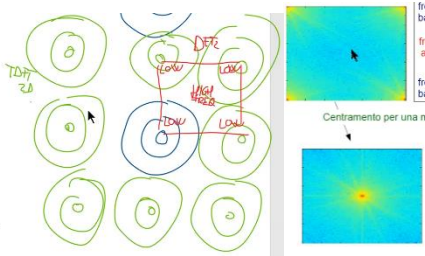
Immaginiamo il variare della tessellazione in funzione di u e v. Avremo delle immaginette complesse B che mostrano cosa stiamo chiedendo alla nostra trasformata (=quanto è presente questo pattern nell'immagine?)

Per ciascuna di queste immagini avremo una risposta in ampiezza tanto più alta quanto è presente quella componente, e otteniamo dunque una nuova immagine.

FFT SHIFT 2D

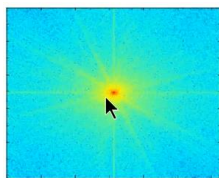


In ambito 1D si arrivava, alla fine, ad avere un asse cartesiano con il risultato ripetuto all'infinito, e in cui noi consideriamo solo un periodo (riordinandolo con fftshift).

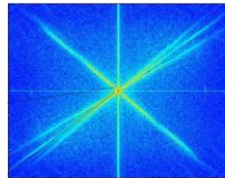


In ambito 2D, analogamente, avrò una funzione bidimensionale con una mappetta 2D ripetuta all'infinito, e con fftshift2 riordino un solo "periodo".

TRASFORMAZIONE LOGARITMICA DELLA FFT2D



Prima di log



Dopo di log

Per costruzione della fft2, il punto al **centro sarà sempre molto elevato** rispetto al resto dell'immagine, in quanto è la somma di tutti i valori di pixel a disposizione.

Quello che si fa in pratica è **applicare la trasformata logaritmica**, che come abbiamo visto nelle operazioni puntuali dà più spazio ai valori più bassi e schiaccia quelli più alti

PROPRIETÀ

Derivano direttamente dalle proprietà della fft 1D: applicarle a due variabili è come applicarla a una sola due volte.

TRASLAZIONE

$$g(x, y) = f(x - x_0, y - y_0)$$

$$G(u, v) = F(u, v) e^{-j2\pi \left(\frac{u}{M} x_0 + \frac{v}{N} y_0 \right)}$$

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

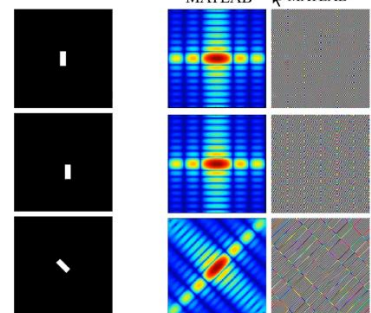
$$G(u, v) = f(u - u_0, v - v_0)$$

ROTAZIONE

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

- la TdF di una immagine a cui è stata applicata una rotazione θ porterà ad una immagine di trasformata ruotata di un angolo θ

MODULO $|F(u, v)|$
FASE $\angle F(u, v)$
abs di MATLAB
angle di MATLAB



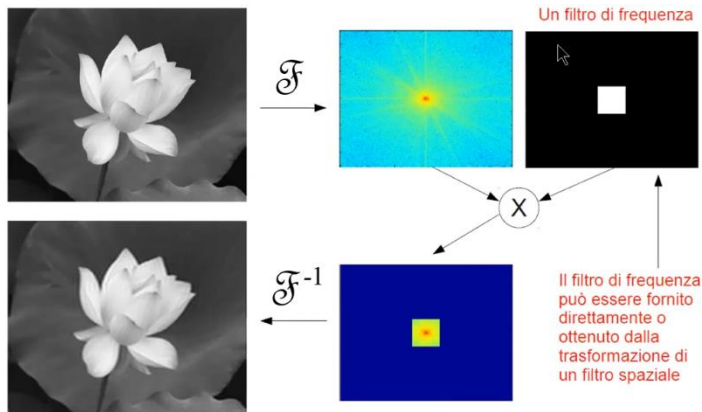
- ! Quando **traslo**, l'ampiezza rimane identica e cambia solo la fase (=mi dice *dove* sta una componente frequenziale)
- ! Quando **ruoto**, ruota anche lo spettro.

Non approfondiamo il segnale di fase, ma in ambito di trasformata inversa, il segnale di fase aiuta a capire dove mettere le varie componenti frequenziali.

PROPRIETÀ DI CONVOLUZIONE

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

Così come era valida in ambito 1D rimane valida in ambito 2D. È fondamentale, poiché volendo fare un filtraggio fra un'immagine e un filtro posso semplicemente moltiplicare gli spettri.



Questo mi permette di progettare filtri che in frequenza si comportano in maniera molto determinata, cosa che non avrei potuto fare rimanendo nel dominio del tempo/spazio. Ho quindi due pipeline possibili:

Pipeline 1

- Progetto filtro in frequenza H
- Calcolo la TDF dell'immagine F
- Moltiplico $F \cdot H$
- Antitrasformo il prodotto

Pipeline 2

- Ho un filtro dato nello spazio
- Eseguo operazione di zero-padding
- Trasformo H in frequenza
- Esegui pipeline 1

RUMORE NEL DOMINIO DELLE FREQUENZE

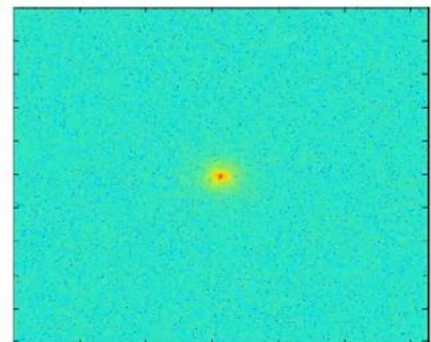
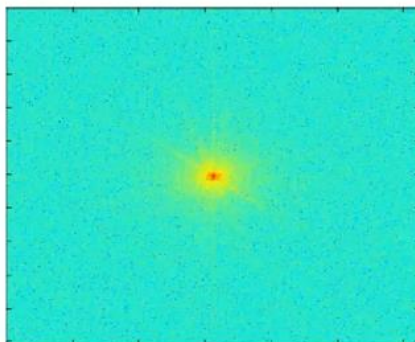
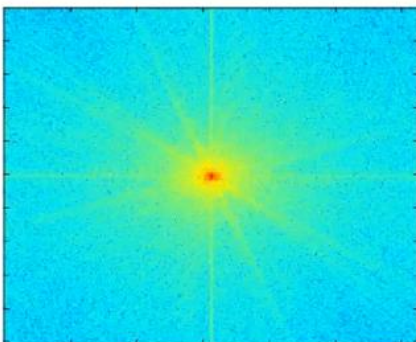
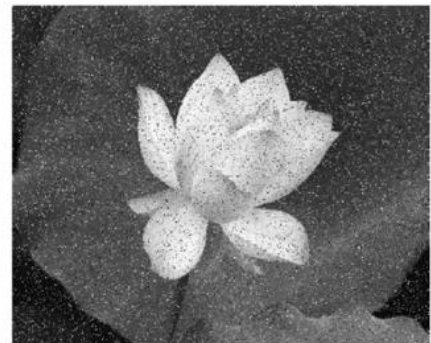
IMMAGINE ORIGINALE



RUMORE GAUSSIANO



RUMORE SALE E PEPE

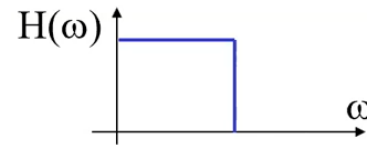


In generale la presenza di rumore porta ad un aumento significativo delle alte frequenze, togliendo dettaglio e creando dei puntini: la fft cerca di contare le istanze di rumore, i picchi appaiono con frequenze randomiche (aka sia a bassa che ad alta frequenza) e "riempie" il grafico.

FILTRI NELLE FREQUENZE

- L'informazione a basse frequenze corrisponde a **variazioni lente**: gradazioni di colore su un muro, il cielo. → Mantenate da un filtro passabasso
- Le informazioni sulle alte frequenze corrispondono a parti dove abbiano **variazioni rapide**, come spigoli angoli e rumore.
→ Mantenate da un filtro passaalto

In generale, i filtri passa alto possono essere derivati dai filtri passaalto e viceversa.



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

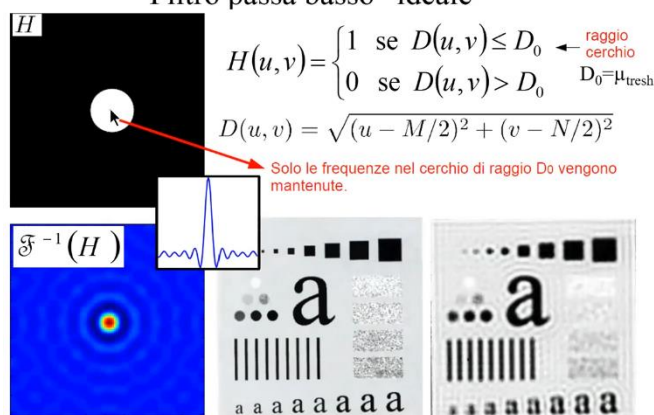
FILTRI PASSABASSO

- La **frequenza di taglio** definisce cosa tolgo e cosa mantengo.
- Un filtro passabasso, di solito, **toglie il rumore**.

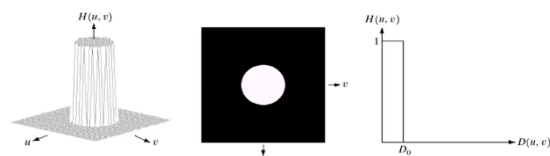
IDEALE	BUTTERWORTH	GAUSSIANO
<ul style="list-style-type: none"> - Transizione di cut-off brusca - Forte fenomeno di ringing 	<ul style="list-style-type: none"> - Ripidità di cut-off varia in funzione dell'ordine - La frequenza di cut-off è selezionata indipendentemente dall'ordine del filtro - Ringings solo su ordini alti 	<ul style="list-style-type: none"> - Transizione di cut-off dolce - Sigma determina la frequenza di cut-off

FILTRO IDEALE

Filtro passa basso "ideale"



Si chiama ideale in quanto è impossibile da ottenere analogicamente: non riesco ad avere un sistema che mi generi un cutoff netto, poiché non riesco a generare la discontinuità. Si usa per modellare i sistemi perché è comodo.



Visualizziamoli nello spazio: ottengo il cerchietto in foto in quanto posso immaginare di "guardarlo da sopra" mentre

ruota.

Ringings

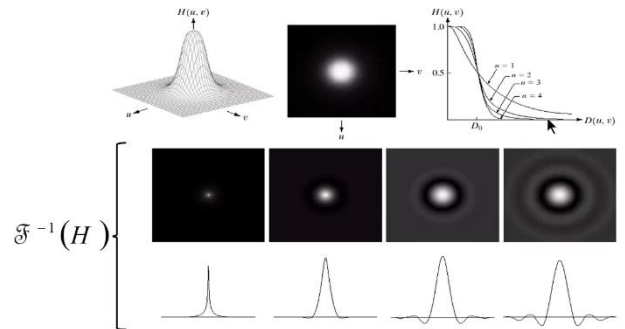
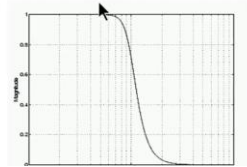
Usando sistemi digitali (e non analogici) potrei comunque usarli, ma non lo faccio perché generano **ringing**, detto anche **effetto di Gibbs**. Esso è dovuto al fatto che filtrare con un PB ideale in frequenza corrisponde a convolvere con una sinc nello spazio. Più alto la frequenza di taglio, più il ringing diventa acceso.

Quindi, il motivo per cui smoothing gaussiano > smoothing di media è che lo smoothing gaussiano corrisponde a un filtro il cui *profilo* è una gaussiana, che a differenza del sinc non mi crea il problema!

FILTRO DI BUTTERWORTH

Ha un profilo simile alla gaussiana e permette di avere una frequenza di taglio comunque abbastanza definita senza cadere nel ringing se l'ordine è < 20 .

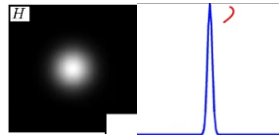
$$H(u, v) = \frac{1}{1 + (D(u, v) / D_0)^{2n}}$$



FILTRO GAUSSIANO

È quello meno parametrizzabile, in quanto il suo profilo è quello dettato dalla varianza. La trasformata di Fourier di una gaussiana è una gaussiana, quindi non ho problemi di ringing.

$$H(u, v) = e^{-\frac{D^2(u, v)}{2D_0^2}}$$



(D sarebbe sigma ed è = alla f di taglio)

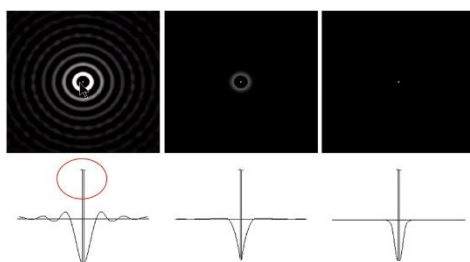
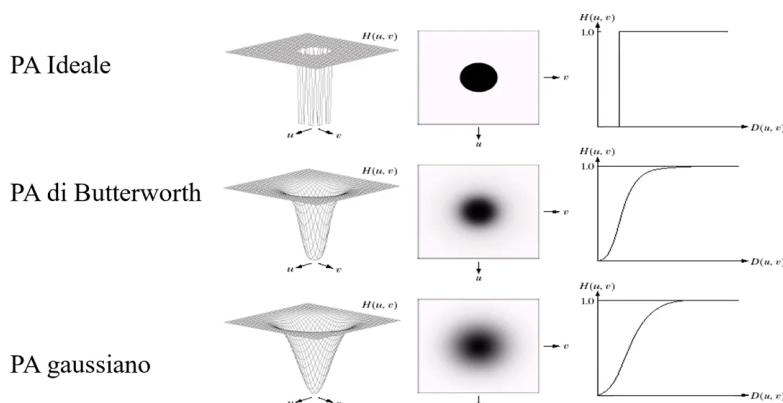
Quando $D(u, v) = \sigma$ l'intensità di taglio è 0.607, ossia il filtraggio crea un attenuamento di quella frequenza del 60.7%

Quindi, riassumendo:

FILTRI PASSA ALTO

Blocca le frequenze basse; come già visto, basta fare $1 - H_{PB}$.

In pratica non ottengo più un'immagine, ma evidenzio tutti i pattern che generano le alte frequenze (= i bordi! :D)



Anche nei filtri passabasso, dove mi aspetto di avere valori bassi al centro, noto che ho un outlier in (0,0)! Questo è dovuto al fatto che la trasformata in 0,0 vale sempre 1.

FILTRI PASSA BANDA E FERMA BANDA

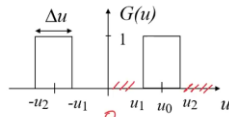
Sono condizioni di passabasso e passa alto che fermano determinate frequenze. Anche qui, per far diventare un passabanda → fermabanda o viceversa mi basta fare

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

FILTRO PASSA-BANDA IDEALE 1D

- Mantenimento inalterato delle componenti in frequenza compresi tra u_1 e u_2 , $u_1 < u_2$
- La funzione di trasferimento desiderata è:

$$G(u) = \begin{cases} 1, & u_1 < |u| < u_2 \\ 0, & \text{altrove} \end{cases}$$

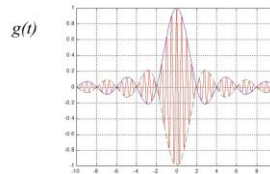


$$\Rightarrow G(u) = \Pi\left(\frac{u}{\Delta u}\right) * [\delta(u - u_0) + \delta(u + u_0)]$$

dove $u_0 = (u_1 + u_2)/2$ e $\Delta u = u_2 - u_1$

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

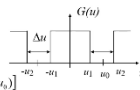
- Poiché $\Delta u < u_0$, allora $g(t)$ è un coseno modulato dal sinc
- Se u_0 è costante e $\Delta u \rightarrow 0$, allora si ottiene un coseno



Per generare un filtro passabanda o fermabanda mi basta convolvere una box per un coseno (che in frequenza diventa due impulsi :))

FILTRO FERMA-BANDA IDEALE 1D

$$G(u) = \begin{cases} 0, & u_1 < |u| < u_2 \\ 1, & \text{altrove} \end{cases}$$



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

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

È semplicemente 1-la funzione precedente

FILTRO PASSA-BANDA 2D

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

IDEALE

$$H(u, v) = \begin{cases} 0 & \text{se } |D(u, v) - D_0| \leq W/2 \\ 1 & \text{altrimenti} \end{cases}$$

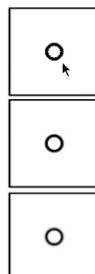
BUTTERWORTH

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

GAUSSIANO

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

dove $W = \Delta u$, $D_0 = u_0$, vedi pagina prec.



Come prima, devo immaginarmi di ruotare il mio filtro in 2D: genero un anello più o meno netto a seconda di quale sto usando.