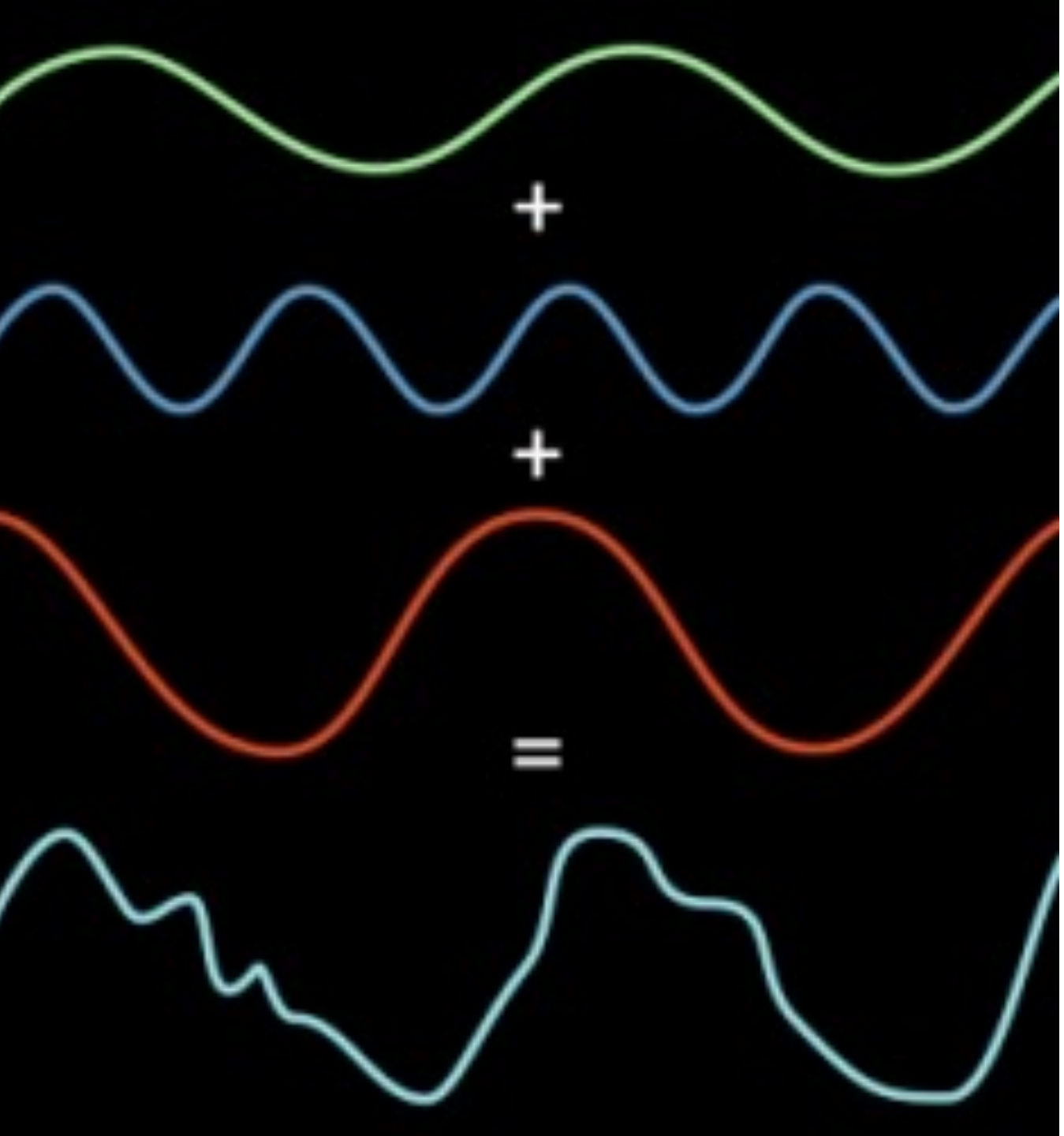
Image Processing:
Fourier Transform

Fondamenti di Visione Artificiale e Biometria

Esempio di utilizzo della Trasformata di Fourier

… « potete incominciare un dialogo con una telefonata, poi ottenerne la Trasformata di Fourier e stralciare tutte le componenti del segnale che hanno frequenze troppo alte o troppo basse per poter essere percepite dall'orecchio umano. Una tale operazione permette di trasmettere più conversazioni con gli stessi canali di comunicazione ed è proprio questa una delle ragioni per cui oggi le nostre bollette del telefono sono relativamente poco care. Lo stesso giochetto non è possibile con il segnale originale, non-trasformato, perché esso non ha la « frequenza » come caratteristica specifica e dunque non sappiamo che cosa potremmo tagliare. »

«Le 17 equazioni che hanno cambiato il mondo»



La Trasformata di Fourier

La Trasformata di Fourier permette di sostituire il segnale originale con il suo spettro, cioè con un elenco di ampiezze e frequenze delle sinusoidi e cosinusoidi.

La Trasformata di Fourier Discreta 2D

La Trasformata di Fourier Discreta (DFT) viene definita come:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2i\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

per $u = 0,1,\dots,M-1 \;\; v = 0,1,\dots N-1$

La sua inversa (DFT $^{-1}$) invece assume la seguente forma:

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

per $x = 0,1,\dots,M-1 \;\; y = 0,1,\dots N-1$

Osservazioni:

1. Il valore della DFT nell'origine, cioè nel punto $(u, v) = (0,0)$, è uguale alla media dei valori di grigio contenuti nell'immagine $f(x, y)$. $\boldsymbol{F(0,0)}$ prende anche il nome di **componente DC.**
2. I valori della trasformata di Fourier sono complessi, ciò significa che hanno una parte reale ed una immaginaria.
3. Possiamo analizzare graficamente la Trasformata computando il suo spettro ($|F(u, v)|$) e visualizzandolo sotto forma di immagine.
4. Lo spettro di Fourier è simmetrico rispetto all'origine.
5. L'algoritmo Fast Fourier Transform (FFT) permette di computare in modo veloce la DFT.

La Trasformata di Fourier in Python (1.1)

Numpy

Funzione np.fft.**fft2**(image, s): **calcola la DFT.**

La funzione richiede come primo argomento un’immagine in scala di grigi. Il secondo è opzionale e riguarda la dimensione dell’array di output. Se è maggiore della dimensione dell'immagine di input, l'immagine di input viene riempita di zeri prima del calcolo della Trasformata. Se è inferiore all'immagine di input, l'immagine di input verrà ritagliata. Se non vengono passati argomenti, la dimensione dell'array di output sarà uguale all'input.

La Trasformata di Fourier in Python (1.2)

Numpy

Funzione np.fft.**fftshift**(f, axes): **trasla la componente DC al centro dello spettro.**

Si fornisce come input alla funzione l'output della precedente. Il secondo valore invece è opzionale e indica gli assi rispetto ai quali bisogna applicare questa operazione di *shift*. Di default è rispetto a tutti gli assi.

La Trasformata di Fourier in Python (1.3)

Numpy

Funzione np.**log**(np.**abs**(v)): **computa lo spettro della DFT. Per una migliore visualizzazione dello spettro si preferisce applicare una trasformazione di tipo logaritmica.**

Si fornisce come input alla funzione l'output della precedente.

La Trasformata di Fourier in Python (1.4)

Numpy

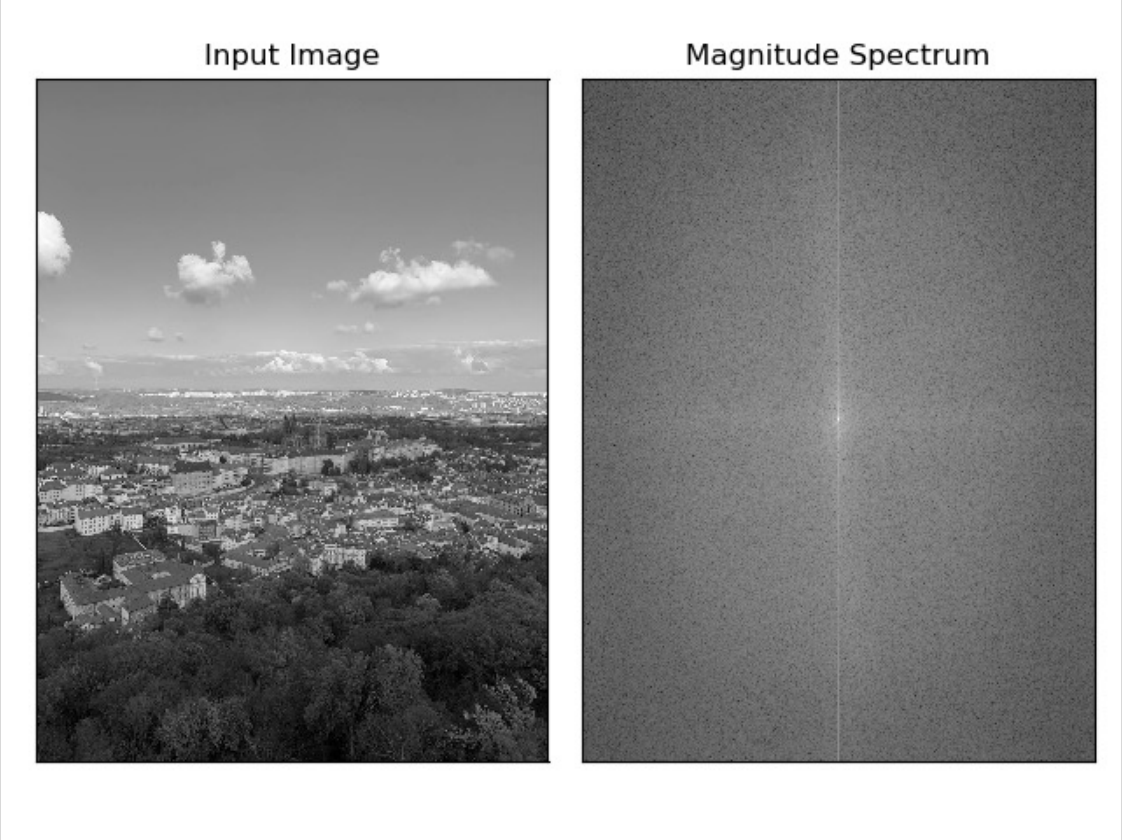
Esempio:

```
import cv2
import numpy as np
from matplotlib import pyplot as pt
img = cv2.imread('immagine.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum=np.log(np.abs(fshift))
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

La Trasformata di Fourier in Python (1.5)

Numpy

Output:



La Trasformata di Fourier in Python (2.1)

Funzione cv2.**dft**(image, flags): **calcola la DFT .**

La funzione richiede come primo argomento l'immagine in input. Il secondo parametro è opzionale e riguarda la rappresentazione dell'array di output. Di default restituisce un array reale della stessa dimensione dell'input. Se flags = cv2.DFT_COMPLEX_OUTPUT si ottiene come output un array complesso.

La Trasformata di Fourier in Python (2.2)

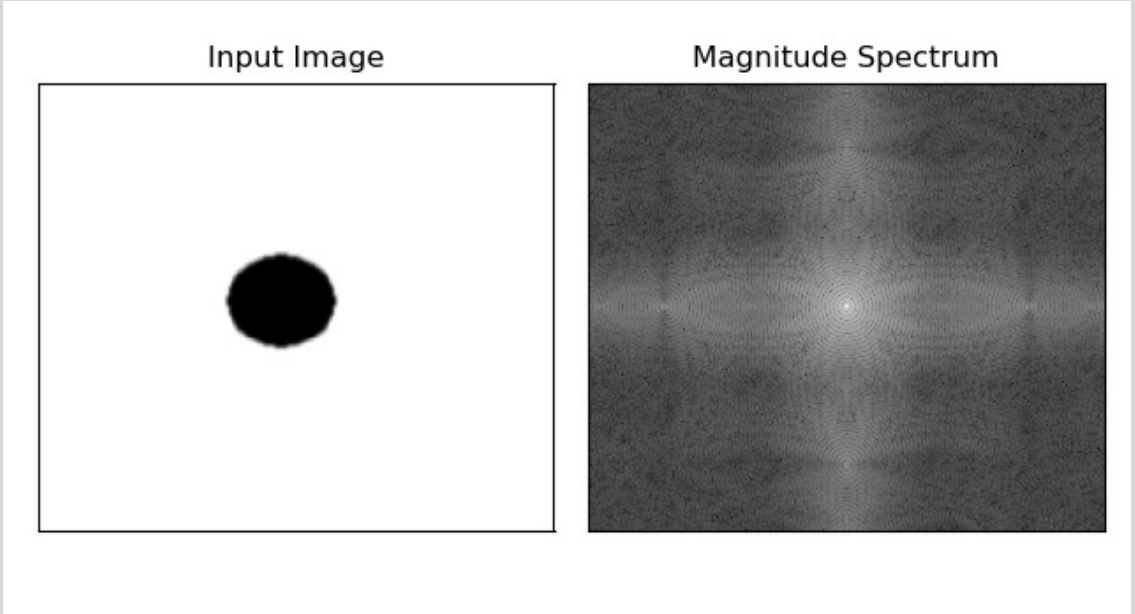
Esempio:

```
import cv2
import numpy as np
from matplotlib import pyplot as pt
img = cv2.imread('immagine.jpg',0)
img_float32= np.float32(img)
dft = cv2.dft(img_float32, flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum=np.log(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :, 1]))
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

La Trasformata di Fourier in Python (2.3)

Numpy

Output:



Teorema di convoluzione

La Trasformata di Fourier del prodotto di convoluzione di due funzioni $f, g \in L^1(R)$ è pari al prodotto delle Trasformate di Fourier delle due funzioni e viceversa:

$$T(f * g) = T(f) \cdot T(g) \qquad T(f \cdot g) = T(f) * T(g)$$

dove T è l'operatore Trasformata di Fourier.

Ciò significa che effettuare la convoluzione di due immagini e poi passare al dominio delle frequenze equivale a passare prima al dominio delle frequenze e poi moltiplicarle e viceversa. Se la convoluzione regolava le operazioni di filtraggio nel dominio spaziale (con forte costo computazionale), passando al dominio in frequenza il filtraggio si riduce ad una moltiplicazione fra matrici.

Filtri nel dominio delle frequenze

I filtri più comuni sono:

1. Filtri passa-basso, anche detti filtri di smoothing;
2. Filtri passa-alto, anche detti filtri di sharpening;
3. Filtri passa-banda.

Filtri passa-basso (1)

Creano un’immagine sfocata, tagliando fuori le alte frequenze (quelle al di sopra di una certa soglia) e lasciando inalterate le basse frequenze. Attenuano il rumore e sfocano i dettagli in modo da conservare le caratteristiche più evidenti.

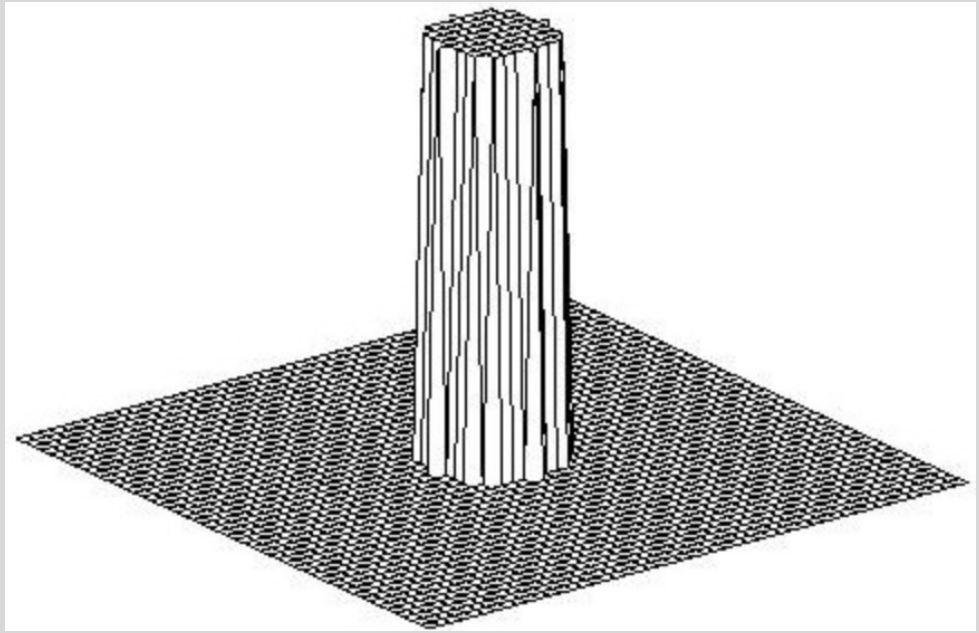
Filtri passa-basso (2)

Esempio: ***Filtri Ideali***

I filtri ideali passa-basso permettono di eliminare le componenti di frequenza che distano dall'origine (centrata) *più* di una certa soglia non negativa, la cosiddetta *frequenza di taglio* (D_0).

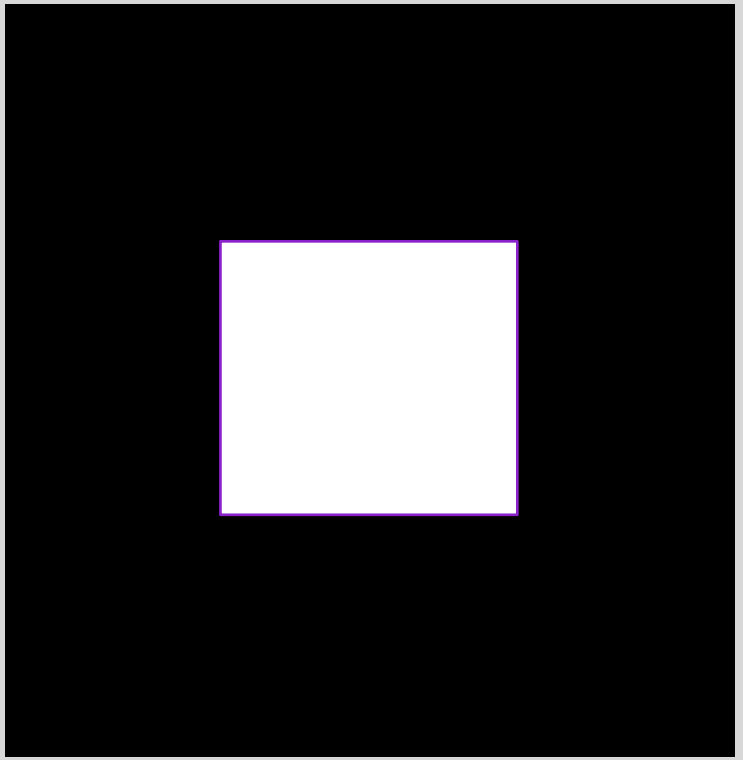
Sia $D(u,v)$ la distanza del generico punto (u,v) dal centro del filtro allora un filtro ideale può essere così rappresentato:

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



Filtri passa-basso (3)

Filtri Ideali passa-basso: tutte le frequenze che si trovano *al di fuori* di un certo range vengono eliminate.



Filtro Ideale passa-basso in Python (1)

- Rappresentiamo l'immagine nel dominio delle frequenze

```
import cv2

import numpy as np

from matplotlib import pyplot as pt

img = cv2.imread('immagine.jpg',0)

img_float32= np.float32(img)

dft = cv2.dft(img_float32, flags = cv2.DFT_COMPLEX_OUTPUT)
```

- Fissiamo un valore di soglia, ad esempio 30

Filtro Ideale passa-basso in Python (2)

- Creiamo una maschera il cui quadrato centrale di lato D_0 avrà in corrispondenza tutti 1. I valori al di fuori di questo quadrato, invece, saranno settati a 0.

```
rows, cols = img.shape
d1,d2, d3= dft_shift.shape
mask = np.zeros((d1, d2, d3), np.uint8)
crow,ccol = rows//2 , cols//2
mask[crow-30:crow+30, ccol-30:ccol+30] = 1
```

- Applichiamo la maschera

```
fshift = dft_shift*mask
```

Dal dominio delle frequenze a quello spaziale in Python

1) Funzione np.fft.**ifftshift**(f, axes): **trasla la componente DC nell'angolo in alto a sinistra.**

È la funzione inversa di **fftshift**. Si fornisce come input alla funzione l'output della precedente. Il secondo valore invece è opzionale e indica gli assi rispetto ai quali bisogna applicare questa operazione di *shift*. Di default è rispetto a tutti gli assi.

2) Funzione cv2.**idft**(x, flags): **calcola l'inversa della DFT.**

La funzione richiede come primo argomento un array complesso. Il secondo parametro è opzionale e riguarda la rappresentazione dell'array di output. Di default restituisce un array reale della stessa dimensione dell'input.

Filtro Ideale passa-basso in Python (3)

- Ritorniamo nel dominio spaziale e visualizziamo l'output

```
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0], img_back[:,:,1]
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Output Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

Filtro Ideale passa-basso in Python (4)

Output:

Filtri passa-alto (1)

Creano un’immagine in cui vengono preservati i bordi e enfatizzati i dettagli. Se si taglia troppo però, si potrebbe degradare la qualità dell'immagine in modo significativo.

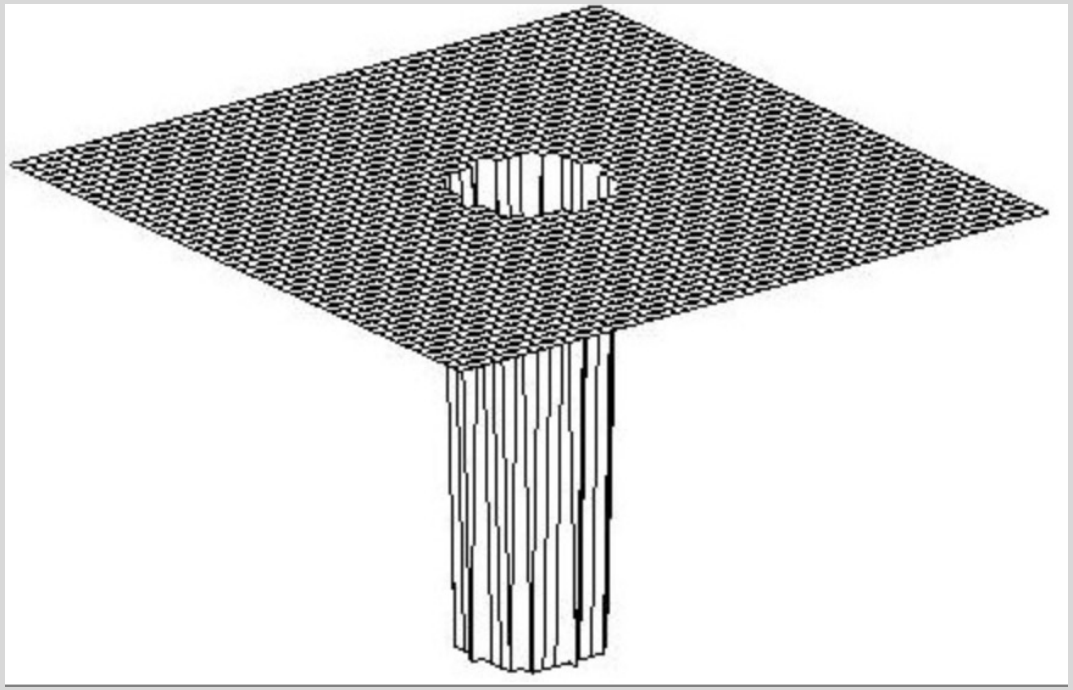
Filtri passa-alto (2)

Esempio: ***Filtri Ideali***

I filtri ideali passa-alto permettono di eliminare le componenti di frequenza che distano dall'origine (centrata) *meno* della *frequenza di taglio* (D_0).

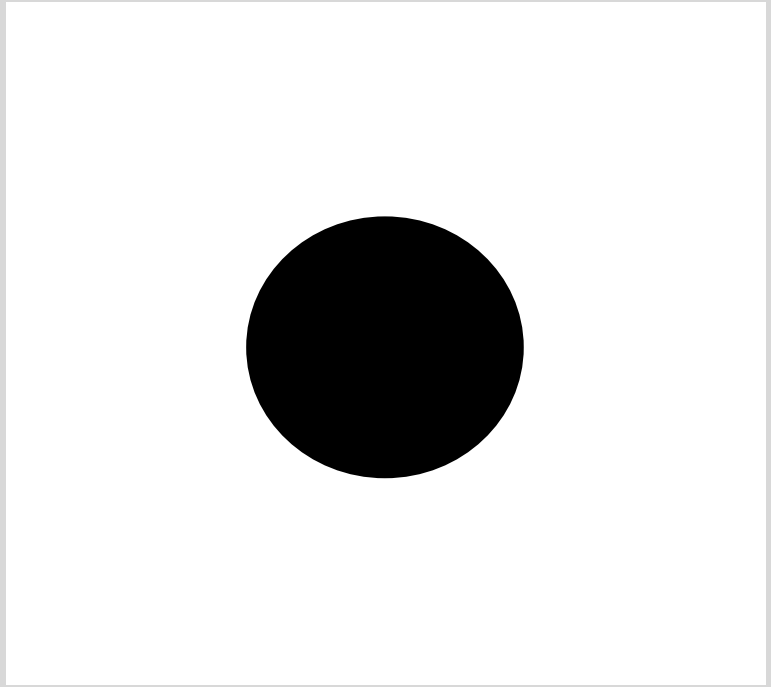
Sia $D(u,v)$ la distanza del generico punto (u,v) dal centro del filtro allora un filtro ideale può essere così rappresentato:

$$H(u,v) = \begin{cases} 0 \;\; se\; D(u,v) \geq D_0 \\ 1\; se\; D(u,v) < D_0 \end{cases}$$



Filtri passa-alto (3)

Filtri Ideali passa-alto: tutte le frequenze che si trovano *all'interno* di un certo range vengono eliminate.



Filtro Ideale passa-alto in Python (1)

- In maniera speculare costruiamo un filtro ideale passa-alto. L'unica differenza è ovviamente nella costruzione della maschera. Stavolta all'interno del quadrato devono esserci tutti 0, mentre al di fuori 1.

```
mask = np.ones((d1, d2, d3), np.uint8)
mask[crow-30:crow+30, ccol-30:ccol+30] = 0
```

Filtro Ideale passa-alto in Python (2)

Output:

Filtri passa-banda (1)

Attenuano le frequenze molto basse e molto alte, mantenendo una banda di frequenze di gamma media. Possono essere utilizzati per migliorare i bordi (sopprimendo le basse frequenze) riducendo allo stesso tempo il rumore (attenuando le alte frequenze).

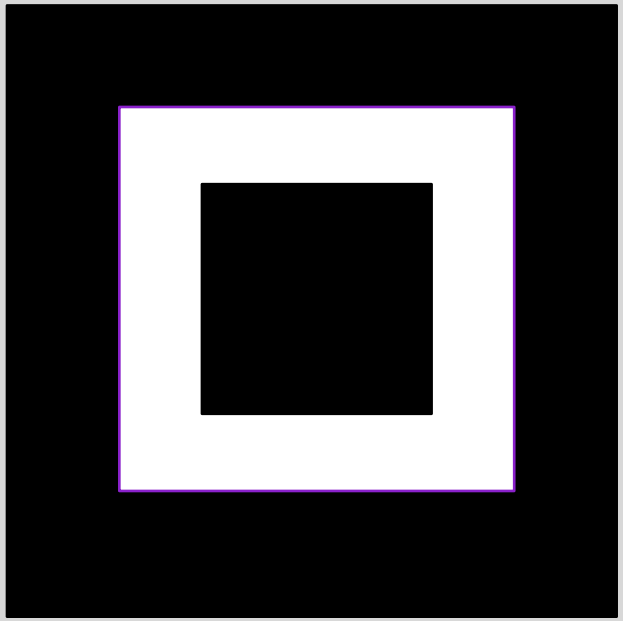
Filtri passa-banda (2)

Esempio: ***Filtri Ideali***

I filtri ideali passa-banda lasciano passare solo le componenti di frequenza che distano dall'origine (centrata) *più* di una certa soglia (D_1) e *meno* di un'altra (D_2).

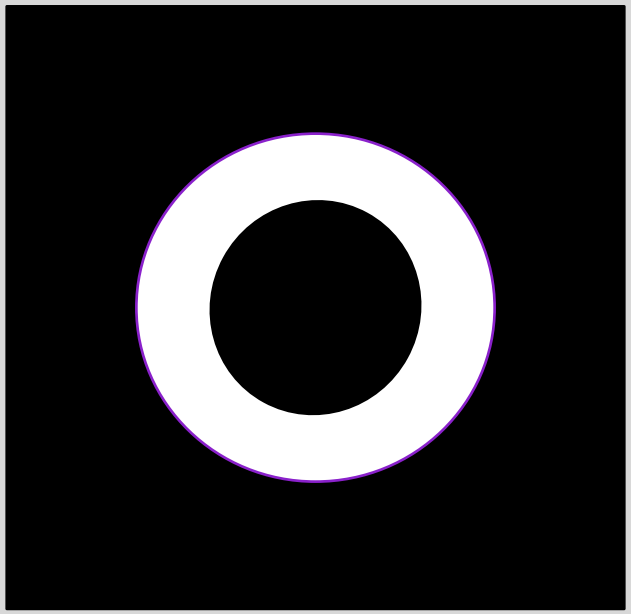
Sia $D(u,v)$ la distanza del generico punto (u,v) dal centro del filtro allora un filtro ideale può essere così rappresentato:

$$H(u,v) = \begin{cases} 1 & se\ D_1 \leq D(u,v) \leq D_2 \\ 0 & altrimenti \end{cases}$$



Filtri passa-banda (3)

Filtri Ideali passa-banda: tutte le frequenze che si trovano *al di fuori* di un certo intervallo vengono eliminate.



Filtro Ideale passa-banda in Python (1)

- Fissiamo le due soglie. In maniera speculare ai casi precedenti costruiamo un filtro ideale passa-banda. L'unica differenza è ovviamente nella costruzione della maschera. Stavolta proviamo a costruire una maschera circolare: all'interno della corona circolare devono esserci tutti 1, mentre al di fuori 0.

```
mask = np.zeros((d1, d2, d3), np.uint8)
r_out = 90
r_in = 5
center = [crow, ccol]
x, y= np.ogrid[:rows, :cols]
mask_area = np.logical_and(((x - center[0]) ** 2 + (y - center[1]) ** 2 >= r_in ** 2), ((x - center[0]) ** 2 + (y - center[1]) ** 2 <= r_out ** 2))
mask[mask_area] = 1
```

Filtro Ideale passa-banda in Python (2)

Output:

Distribuzione delle frequenze in immagini

Curiosità

Alcuni ricercatori dell'MIT (USA – Torralba/Oliva) hanno pubblicato nel 2003 un articolo in cui si analizzano delle statistiche nel dominio di Fourier rispetto alla distribuzione dell'energia per varie classi (e categorie) di immagini naturali.