

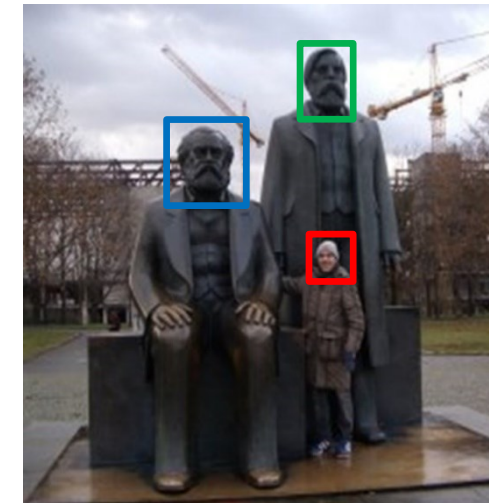
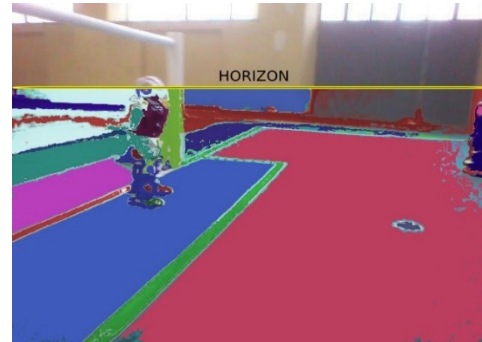
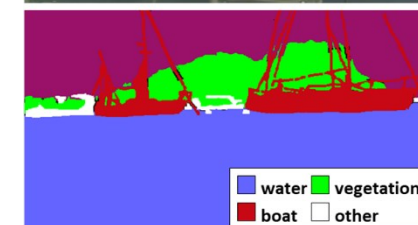
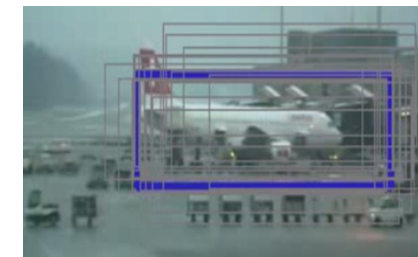


**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi

Processamento delle immagini



Marzo 2019

Immagine Digitale

- Una immagine digitale è una matrice di pixel
- Il termine pixel deriva da *picture element*
- Il pixel contiene l'informazione relativa alla rappresentazione della realtà che è stata catturata tramite uno scanner, una macchina fotografica o un frame grabber (per i video)



Processamento delle immagini

Per poter elaborare il contenuto di una immagine, avremo bisogno di caricarla in memoria per poter accedere ai suoi elementi e modificarli.

Una volta terminate le modifiche, potremmo voler salvare l'immagine modificata su disco.

Per poter processare le immagini utilizzeremo delle librerie esterne.

La libreria NumPy

NumPy è una libreria per il calcolo scientifico in Python.

La utilizzeremo principalmente per la gestione degli array N-dimensionali e per la definizione di nuovi tipi di dato



<http://www.numpy.org/>

NumPy viene rilasciata sotto la [BSD license](#)

Array in NumPy

```
import numpy as np  
a = np.array([1, 2, 3])  
print(a)
```

```
[1 2 3]
```

NumPy è inclusa in Google Colab

Un array in NumPy è una griglia di valori, tutti dello stesso tipo.

Gli array sono indicizzati.

Array in NumPy

Il *rank* è la dimensione dell'array (ndim)

La *shape* è una tupla di interi che fornisce la lunghezza dell'array lungo ogni dimensione

Array in NumPy



```
import numpy as np

a = np.array([1, 2, 3])
print(a.ndim) #rank
print(a.shape)

b = np.array([[1,2,3],[4,5,6]])
print(b.ndim)
print(b.shape)
```



```
1
(3,)
2
(2, 3)
```

Funzioni per creare array

creazione di un array con i tutti valori a zero

```
[10] import numpy as np  
      a = np.zeros((1,3))  
      print(a)
```

↳ `[[0. 0. 0.]]`

creazione di un array con tutti i valori a uno

```
[12] b = np.ones((3,1))  
      print(b)
```

↳ `[[1.]
 [1.]
 [1.]]`

Funzioni per creare array

creare un array di valori costanti

```
[5] import numpy as np  
  
    c = np.full((2,3), 6.2)  
    print(c)
```

```
↳ [[6.2 6.2 6.2]  
    [6.2 6.2 6.2]]
```

creare una matrice identità

```
▶ d = np.eye(3)  
  print(d)
```

```
↳ [[1. 0. 0.]  
    [0. 1. 0.]  
    [0. 0. 1.]]
```

Funzioni per creare array

creare una matrice di numeri (pseudo)random



```
import numpy as np
```

```
r_1 = np.random.random((5,))  
print(r_1)
```

```
r_2 = np.random.random((2,3))  
print(r_2)
```

```
↳ [0.37389921 0.79844257 0.35715868 0.54446747 0.51375202]  
   [[0.56753889 0.77342533 0.85682621]  
    [0.33778405 0.41373532 0.2560605 ]]
```

Slicing



```
import numpy as np

a = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a)
s1 = a[1,:]
print(s1)
s2 = a[1:2,:]
print(s2)
s3 = a[2,0:2]
print(s3)
```



```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[4 5 6]
[[4 5 6]]
[7 8]
```

Indexing



```
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

print(a[0])
print(a[0,1])
print(a[[0,1]])
print(a[[0,1,2]])
print(a[[0,1,2],[0]])
print(a[[0,1,2],[0,1,1]])
```



```
[1 2]
2
[[1 2]
 [3 4]]
[[1 2]
 [3 4]
 [5 6]]
[1 3 5]
[1 4 6]
```

Boolean indexing



```
import numpy as np

b = np.array([[1,2], [3, 4], [5, 6]])

print(b)

print(b > 3)

print(b[b > 3])
```

```
[[1 2]
 [3 4]
 [5 6]]
[[False False]
 [False  True]
 [ True  True]]
[4 5 6]
```

Tipi di dato



```
import numpy as np

a = np.array([22, 33, 44])
print(a)
print(a.dtype)

b = np.array([22.3, 44.5])
print(b)
print(b.dtype)

c = np.array([22, 33, 44], dtype=np.float64)
print(c)
print(c.dtype)
```



```
[22 33 44]
int64
[22.3 44.5]
float64
[22. 33. 44.]
float64
```

Tipi di dato

Numpy type	C type	Description
<code>np.int8</code>	<code>int8_t</code>	Byte (-128 to 127)
<code>np.int16</code>	<code>int16_t</code>	Integer (-32768 to 32767)
<code>np.int32</code>	<code>int32_t</code>	Integer (-2147483648 to 2147483647)
<code>np.int64</code>	<code>int64_t</code>	Integer (-9223372036854775808 to 9223372036854775807)
<code>np.uint8</code>	<code>uint8_t</code>	Unsigned integer (0 to 255)
<code>np.uint16</code>	<code>uint16_t</code>	Unsigned integer (0 to 65535)
<code>np.uint32</code>	<code>uint32_t</code>	Unsigned integer (0 to 4294967295)
<code>np.uint64</code>	<code>uint64_t</code>	Unsigned integer (0 to 18446744073709551615)
<code>np.intp</code>	<code>intptr_t</code>	Integer used for indexing, typically the same as <code>ssize_t</code>
<code>np.uintp</code>	<code>uintptr_t</code>	Integer large enough to hold a pointer
<code>np.float32</code>	<code>float</code>	Note that this matches the precision of the builtin python <i>float</i> .
<code>np.float64 / np.float_</code>	<code>double</code>	
<code>np.complex64</code>	<code>float complex</code>	Complex number, represented by two 32-bit floats (real and imaginary components)
<code>np.complex128 / np.complex_</code>	<code>double complex</code>	Note that this matches the precision of the builtin python <i>complex</i> .

Operazioni con gli array



```
import numpy as np

a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a + b)
print(np.add(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])

print(c + d)
print(np.add(c, d))
```



```
[ 6  8 10 12]
[ 6  8 10 12]
[[ 6  8]
 [10 12]]
[[ 6  8]
 [10 12]]
```


ValueError



```
import numpy as np

a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a + b)
print(np.add(a, b))

c = np.array([[1,2,4], [3,4,4]])
d = np.array([[5,6], [7,8]])

print(c + d)
print(np.add(c, d))
|
```



```
[ 6  8 10 12]
[ 6  8 10 12]
```

ValueError

Traceback (most recent call last)

[<ipython-input-9-25beb093c60b>](#) in <module>()

10 d = np.array([[5,6], [7,8]])

11

---> 12 print(c + d)

13 print(np.add(c, d))

14

ValueError: operands could not be broadcast together with shapes (2,3) (2,2)

SEARCH STACK OVERFLOW

Sottrazione



```
import numpy as np
```

```
a = np.array([1,2,3,4])
```

```
b = np.array([5,6,7,8])
```

```
print(a - b)
```

```
print(np.subtract(a, b))
```

```
c = np.array([[1,2], [3,4]])
```

```
d = np.array([[5,6], [7,8]])
```

```
print(c - d)
```

```
print(np.subtract(c, d))
```



```
[-4 -4 -4 -4]
```

```
[-4 -4 -4 -4]
```

```
[[ -4  -4]
```

```
[-4 -4]]
```

```
[[ -4  -4]
```

```
[-4 -4]]
```

Divisione



```
import numpy as np

a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a / b)
print(np.divide(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])

print(c / d)
print(np.divide(c, d))
```



```
[0.2      0.33333333 0.42857143 0.5      ]
[0.2      0.33333333 0.42857143 0.5      ]
[[0.2      0.33333333]
 [0.42857143 0.5      ]]
[[0.2      0.33333333]
 [0.42857143 0.5      ]]
```

Moltiplicazione elemento per elemento



```
import numpy as np

a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a * b)
print(np.multiply(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])

print(c * d)
print(np.multiply(c, d))
```



```
[ 5 12 21 32]
[ 5 12 21 32]
[[ 5 12]
 [21 32]]
[[ 5 12]
 [21 32]]
```

Prodotto scalare



```
import numpy as np

a = np.array([1,2,3,4])
b = np.array([5,6,7,8])
print(a.dot(b))
print(np.dot(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])
print(c.dot(d))
print(np.dot(c, d))

e = np.array([[1,2], [3,4], [5,6]])
print(e.dot(d))
print(np.dot(e, d))
```



```
70
70
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]
 [67 78]]
[[19 22]
 [43 50]
 [67 78]]
```

Trasposta



```
import numpy as np

a = np.array([[1,2], [3,4]])
print(a)

print(a.T)
```



```
[[1 2]
 [3 4]]
[[1 3]
 [2 4]]
```

Broadcasting

1	2	3
4	5	6

A

1	2	3
4	5	6
7	8	9

X

1	2	3
---	---	---

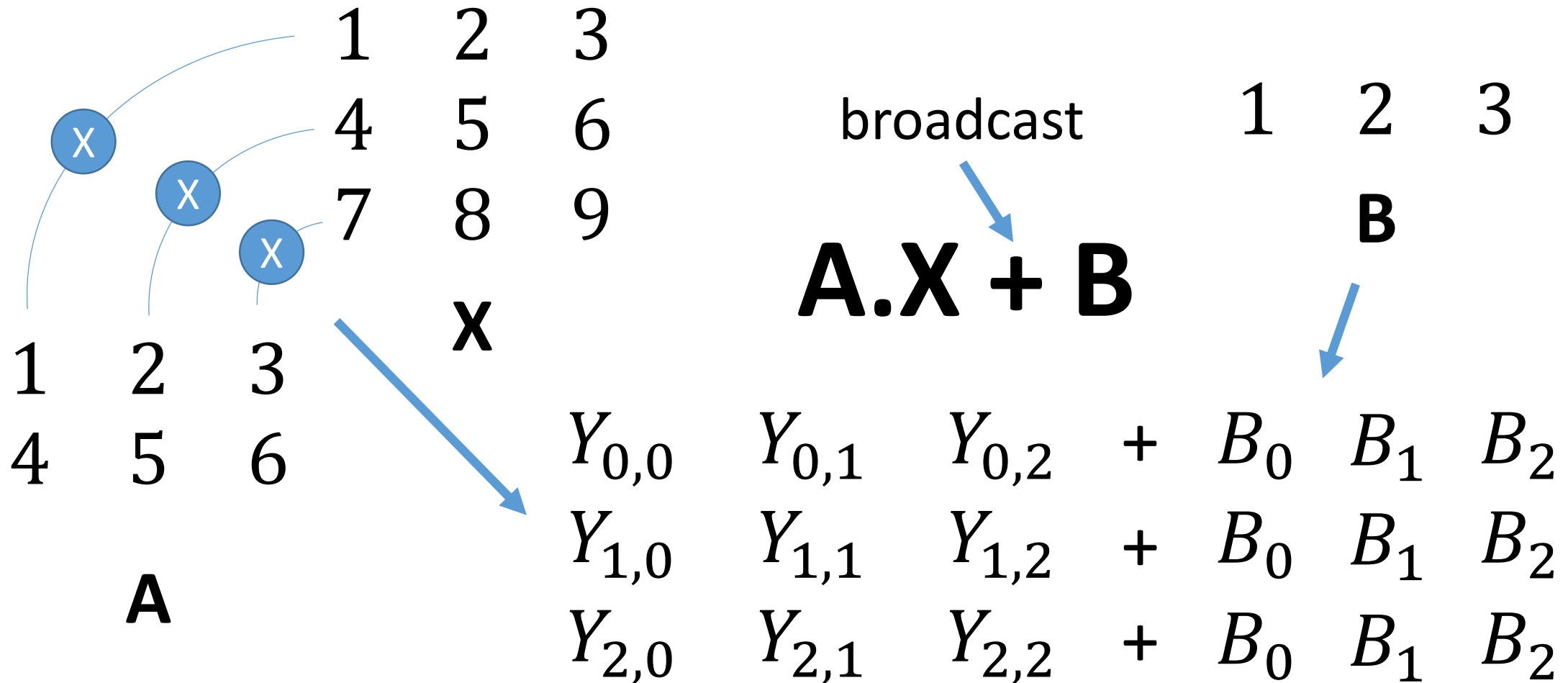
B

A.X + B



broadcast

Broadcasting



Broadcasting



```
A = np.array([[1,2,3],[4,5,6]])  
X = np.array([[1,2,3], [4,5,6], [7,8,9]])  
B = np.array([1,2,3])  
print(A.dot(X))  
print(A.dot(X) + B)
```



```
[[30 36 42]  
 [66 81 96]]  
[[31 38 45]  
 [67 83 99]]
```

Matplotlib

Matplotlib è una libreria Python per il plotting in 2D

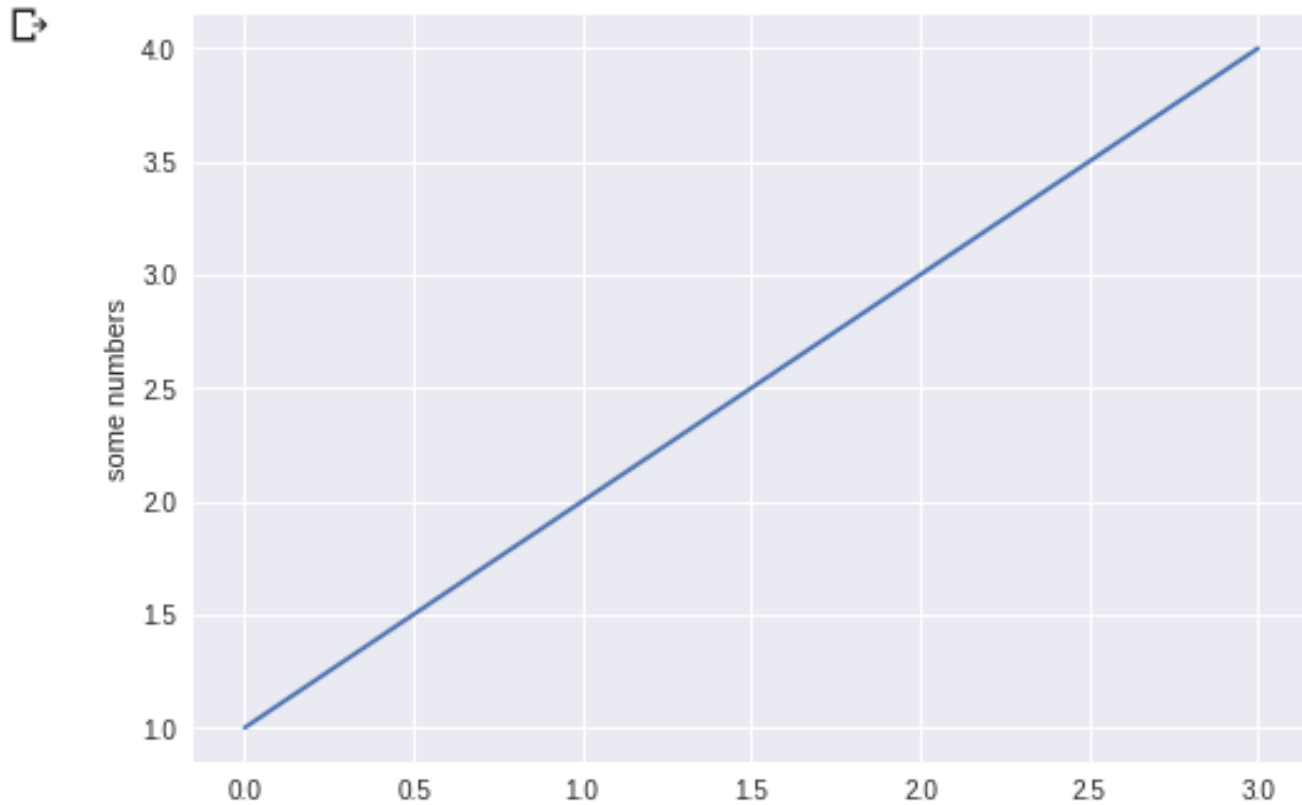


<https://matplotlib.org/>

Con matplotlib è possibile generare grafici, istogrammi, spettri, diagrammi a barre, grafici di dispersione e altro ancora usando una interfaccia tipo MATLAB

plot

```
import matplotlib.pyplot as plt  
  
plt.plot([1,2,3,4])  
plt.ylabel('some numbers')  
plt.show()
```



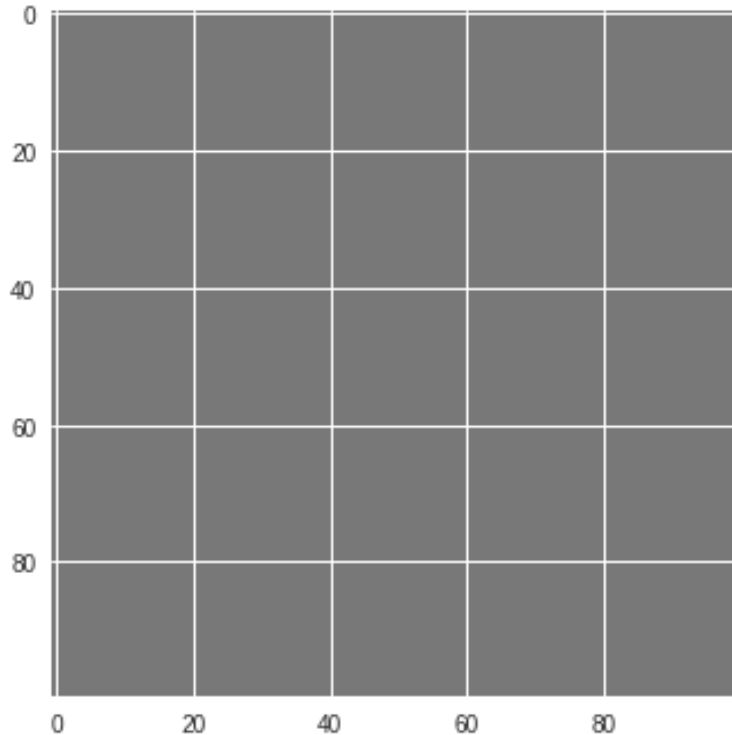
https://matplotlib.org/users/pyplot_tutorial.html

Immagine con matplotlib

```
import numpy as np
import matplotlib.pyplot as plt


img = np.ones([100,100,3], dtype=np.uint8)*120
plt.imshow(img)
```

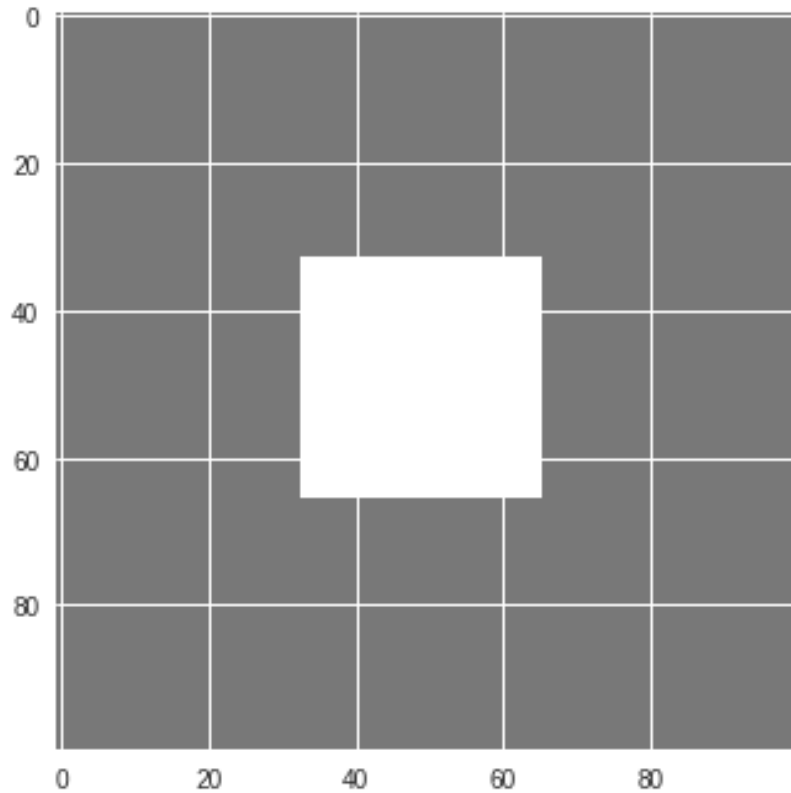
<matplotlib.image.AxesImage at 0x7f480341ccf8>



Modifica dell'immagine

```
img[33:66,33:66,:] = 255  
plt.imshow(img)
```

 <matplotlib.image.AxesImage at 0x7f4803376550>

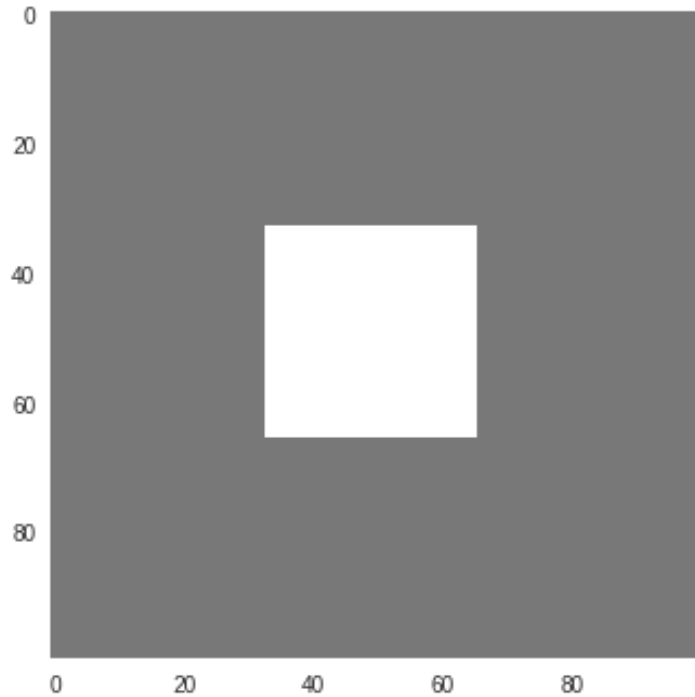


grid

```
import numpy as np
import matplotlib.pyplot as plt

img = np.ones([100,100,3], dtype=np.uint8)*120
img[33:66,33:66,:] = 255
plt.grid(b=None)
plt.imshow(img)
```

<matplotlib.image.AxesImage at 0x7f6d196e74e0>



axis



```
import numpy as np
import matplotlib.pyplot as plt

img = np.ones([100,100,3], dtype=np.uint8)*120
img[33:66,33:66,:] = 255
plt.axis('off')
plt.imshow(img)
```



<matplotlib.image.AxesImage at 0x7f6d196bb940>



Salvare l'immagine

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

numpy_array = np.ones([100,100,3], dtype=np.uint8)*120
numpy_array[33:66,33:66,:] = 255
plt.axis('off')
plt.imshow(numpy_array)

pil_img = Image.fromarray(numpy_array)
pil_img.save("image.bmp")

!ls
```

image.bmp sample_data



Pillow

Pillow è una libreria open source per aprire, elaborare e salvare immagini derivata dalla Python Imaging Library (PIL)



Homepage: <https://python-pillow.org/>

Source code: <https://github.com/python-pillow/Pillow>

Documentation: <https://pillow.readthedocs.io/>

Aprire l'immagine con Pillow

```
▶ import matplotlib.pyplot as plt  
from PIL import Image  
  
!ls  
  
bmp_img = Image.open("image.bmp")  
plt.axis('off')  
plt.imshow(bmp_img)
```

```
↳ image.bmp image.jpg sample_data  
<matplotlib.image.AxesImage at 0x7fd4f5d2e4e0>
```



Immagini in Jpeg

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

numpy_array = np.ones([100,100,3], dtype=np.uint8)*120
numpy_array[33:66,33:66,:] = 255

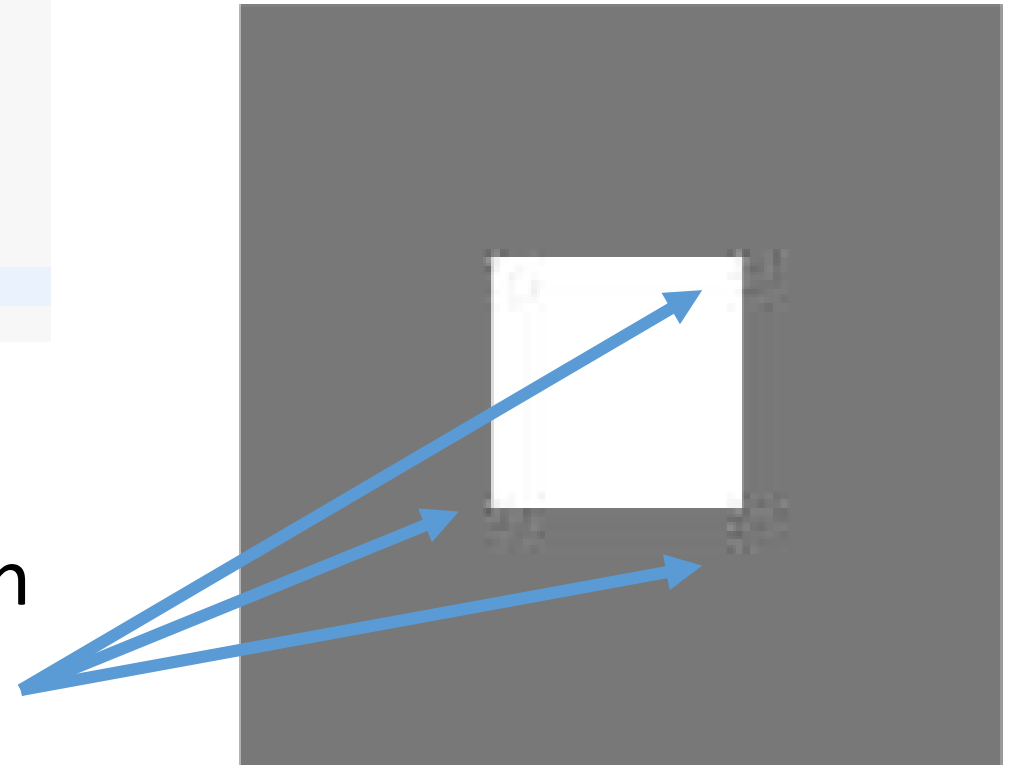
pil_img = Image.fromarray(numpy_array)
pil_img.save("image.jpg")

!ls

jpg_img = Image.open("image.jpg")
plt.axis('off')
plt.imshow(jpg_img)
```

```
image.bmp image.jpg sample_data
<matplotlib.image.AxesImage at 0x7fd4f5cf5f98>
```

compression
artifacts



Immagini: occupazione di memoria

L'occupazione di memoria è data dal prodotto tra la dimensione dell'immagine e la profondità di colore del singolo pixel

Occupazione = (Dimensione) x (Profondità di colore)

Esempio:

Una immagine a colori (RGB) 640x480 occupa in memoria 9830400 bit pari a circa 1.23 MB

Tipi di Compressione

La compressione può essere "lossless" o "lossy" a seconda del numero di bit riservati alla differenza

- Compressione "lossless": **reversibile**
Ad esempio file PNG e file ZIP
- Compressione "lossy": **ricostruzione approssimata**, dove maggiore è il rapporto di compressione, maggiore è l'errore
Ad esempio file JPEG e file MP3

Formato BMP

- Il formato BMP (bitmap) è stato sviluppato da Microsoft per la gestione dei file in Windows.
- Si tratta di un formato piuttosto datato (anni 90) che permette di salvare immagini in grayscale e a colori
- Viene usato di solito per salvare immagini senza compressione (lossless)

<https://docs.microsoft.com/en-us/windows/desktop/gdi/bitmap-storage>

Formato PNG

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

numpy_array = np.ones([100,100,3], dtype=np.uint8)*120
numpy_array[33:66,33:66,:] = 255

pil_img = Image.fromarray(numpy_array)
pil_img.save("image.png")

!ls

png_img = Image.open("image.png")
plt.axis('off')
plt.imshow(png_img)
```

```
image.bmp image.jpg image.png sample_data
<matplotlib.image.AxesImage at 0x7fd4f5c492e8>
```



Il formato PNG
(Portable Network Graphics)
utilizza un algoritmo di
compressione lossless che
permette, a differenza
dell'algoritmo lossy del JPEG, di
preservare dettagli e sfumature
di colore nell'immagine

<http://www.libpng.org/pub/png/>

Ridurre l'occupazione di memoria

L'occhio umano è meno sensibile alle ALTE frequenze spaziali:

- Se l'ampiezza di una componente ad ALTA frequenza cade sotto una certa soglia, l'occhio NON LA RILEVA



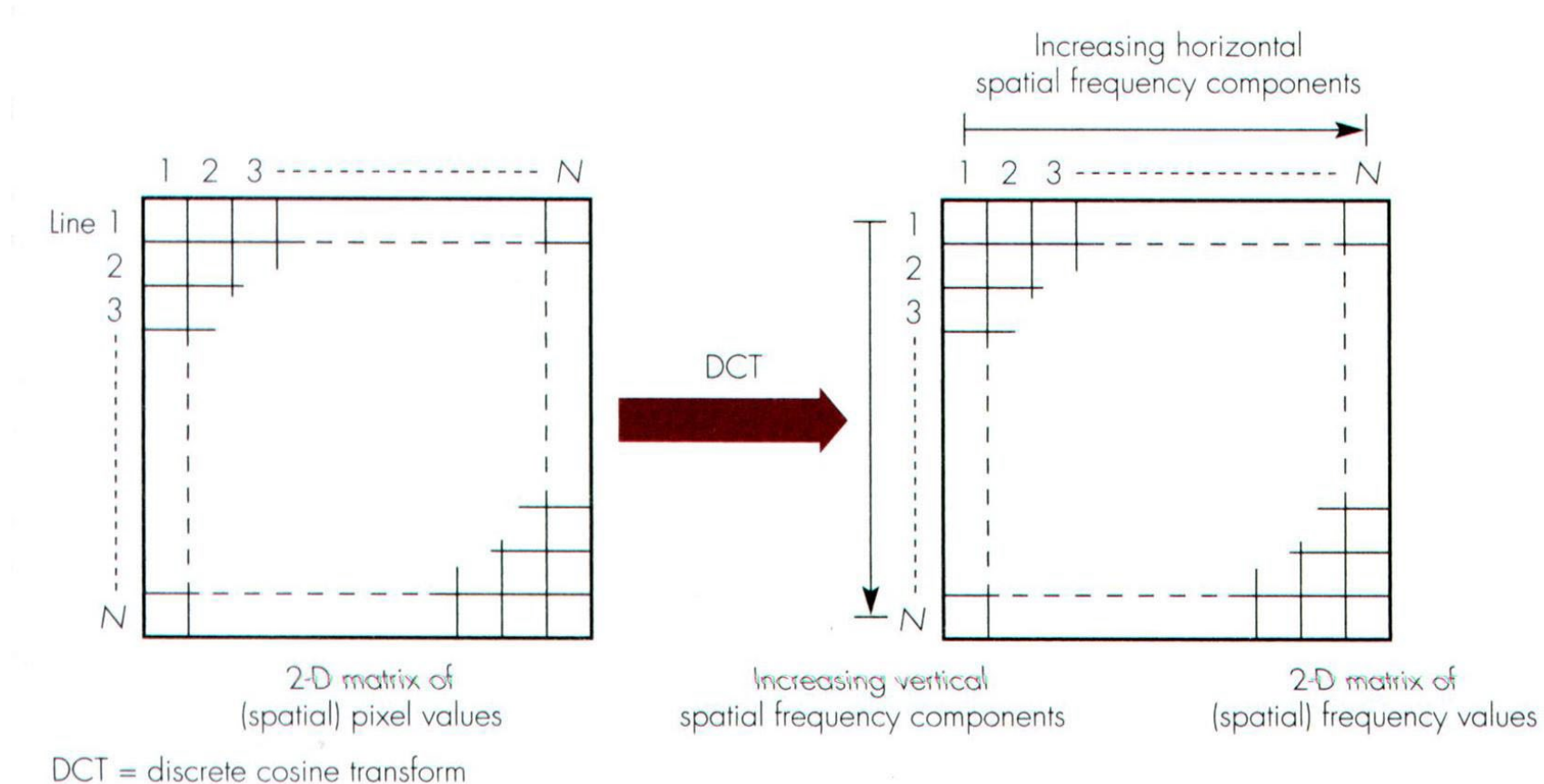
La quantizzazione può essere
meno accurata alle alte frequenze



Possono essere utilizzati meno bit

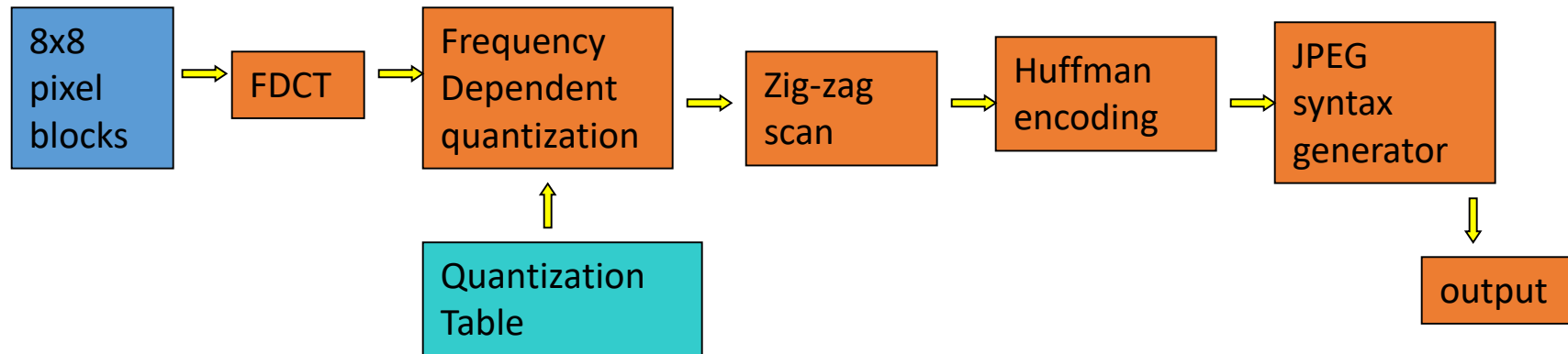
Discrete Cosine Transform

La DCT trasforma una matrice bi-dimensionale di pixel in unamatrice equivalente di "spatial frequency components"



Compressione JPEG

- L'immagine viene divisa in blocchi 8x8
- Si applica la 2D Fourier Discrete Cosine Transform (FDCT)
- I componenti ad alta frequenza spaziale vengono quantizzati più grossolanamente
- I dati risultanti dalla quantizzazione vengono compressi con un meccanismo senza perdita di informazione



Quantizzazione in frequenza

Spatial domain

128	128	128	128	128	128	128	128
118	111	112	117	120	123	123	122
125	121	115	111	119	119	118	117
120	121	113	113	125	124	115	108
120	120	116	119	124	120	115	110
117	113	111	122	120	110	116	119
109	113	111	122	120	110	116	119
111	121	124	118	115	121	117	113

Frequency domain

-80	4	-6	6	2	-2	-2	0
24	-8	8	12	0	0	0	2
10	-4	0	-12	-4	4	4	-2
8	0	-2	-6	10	4	-2	0
18	4	-4	6	-8	-4	0	0
-2	8	6	-4	0	-2	0	0
12	0	6	0	0	0	-2	-2
0	8	0	-4	-2	0	0	0

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantization Matrix to divide by

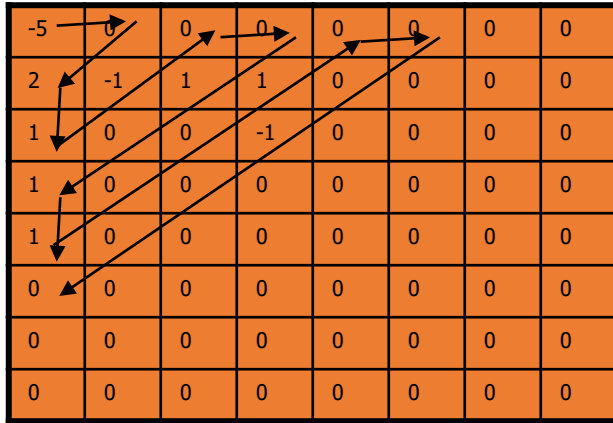
-5	0	0	0	0	0	0	0
2	-1	1	1	0	0	0	0
1	0	0	-1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized spatial frequency values

Immagine tratta da M. Moewe
“Media Compression
Techniques”

Scanning e Huffman Encoding

- Si usa un percorso a zig-zag per scansionare le frequenze spaziali
- Le alte frequenze valgono quasi sempre zero
- La Huffman encoding è usata per immagazzinare con compressione lossless i valori



i valori

0,2,1,-1,0,0,1,0,1,1,0,0,1,0,0,0,-1,0,0,... 0

Possono essere memorizzati come

$(1,2),(0,1),(0,-1),(2,1),(1,1),(0,1),(0,1),(2,1),(3,1),\text{EOB}$

Vari livelli di compressione JPEG



500KB image, minimum compression



40KB image, half compression



11KB image, max compression

Perdita di dettagli



Uncompressed image
(roughness between
pixels still visible)



Half compression,
blurring & halos around
sharp edges



Max compression, 8-
pixel blocks apparent,
large distortion in high-
frequency areas

Jpeg quality

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

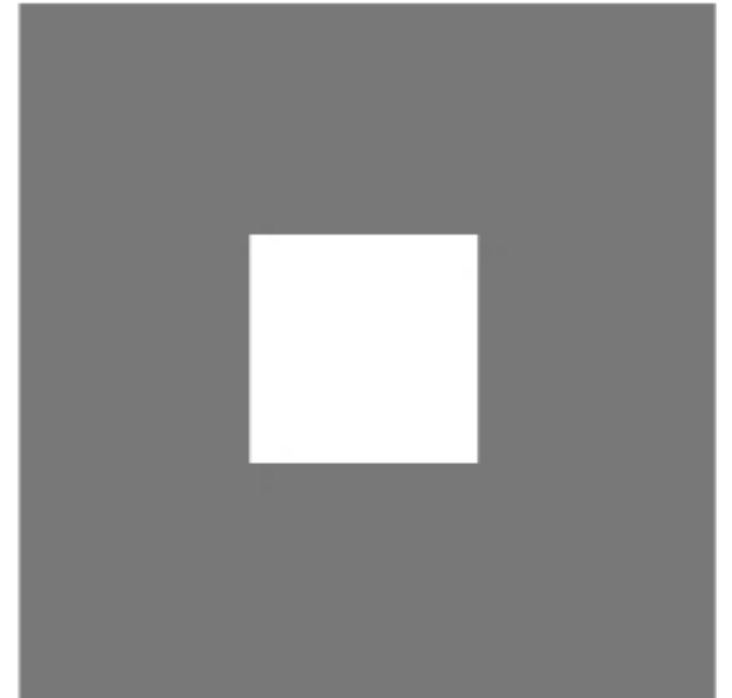
numpy_array = np.ones([100,100,3], dtype=np.uint8)*120
numpy_array[33:66,33:66,:] = 255

pil_img = Image.fromarray(numpy_array)
pil_img.save("image.jpg", format='JPEG', quality=100)

!ls

jpg_img = Image.open("image.jpg")
plt.axis('off')
plt.imshow(jpg_img)
```

image.bmp image.jpg image.png sample_data
<matplotlib.image.AxesImage at 0x7fd4f5c135f8>




Cropping

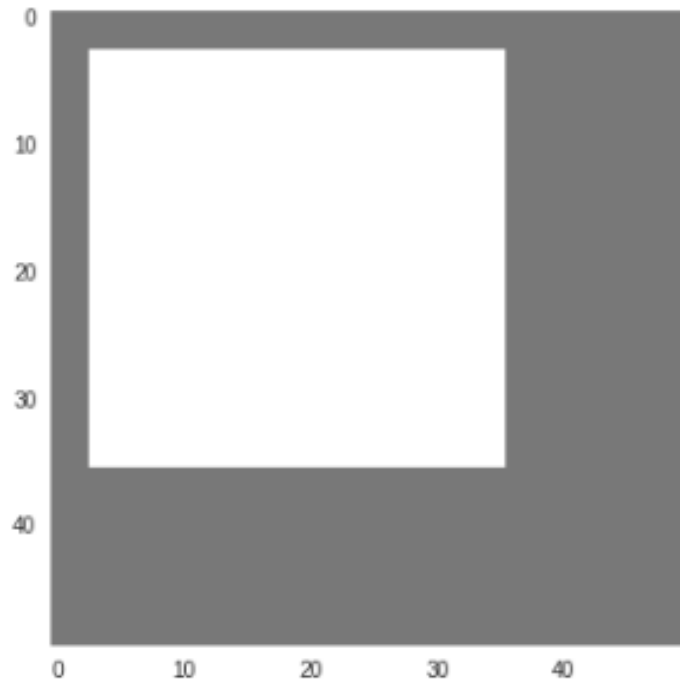
```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("image.png")

roi = (30,30,80,80)
crop_img = img.crop(roi)

plt.grid(b=False)
plt.imshow(crop_img)
```

 <matplotlib.image.AxesImage at 0x7fd4f5ac8780>




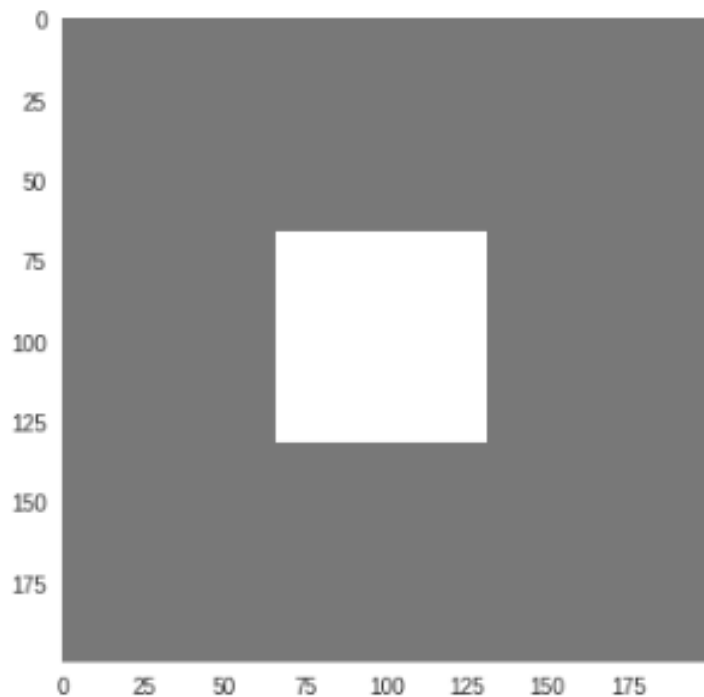
Resize

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("image.png")
resize_img = img.resize((200,200))

plt.grid(b=False)
plt.imshow(resize_img)
```

 <matplotlib.image.AxesImage at 0x7fd4f5a98668>



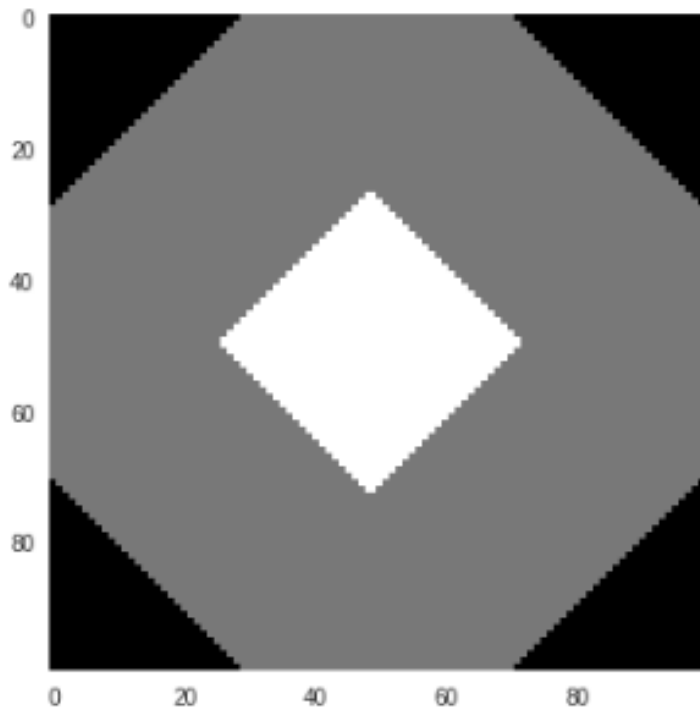
Rotate

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("image.png")
rotate_img = img.rotate(45)

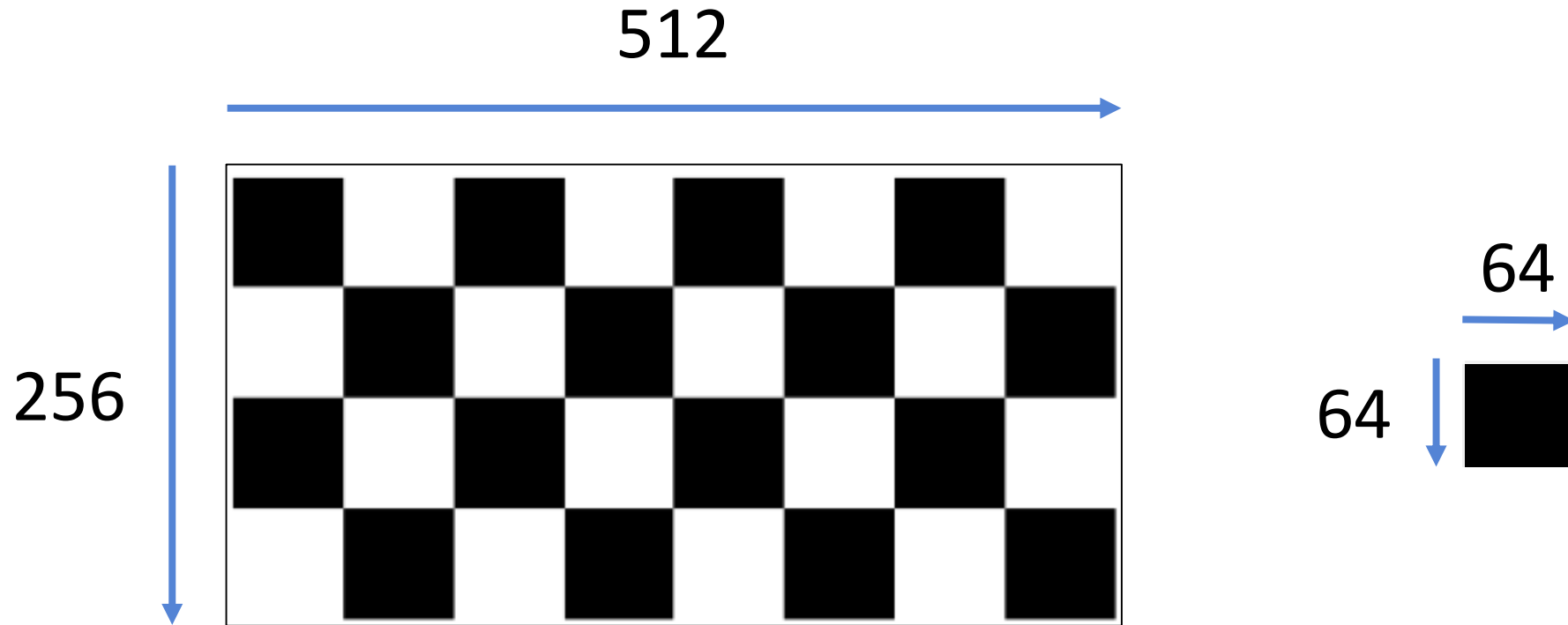
plt.grid(b=False)
plt.imshow(rotate_img)
```

<matplotlib.image.AxesImage at 0x7fd4f5a70f98>



Esercizio

Realizzare una immagine come quella in figura utilizzando le librerie NumPy, Matplotlib e Pillow



Esercizio: soluzione



```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

width = 512
height = 256
l = 64

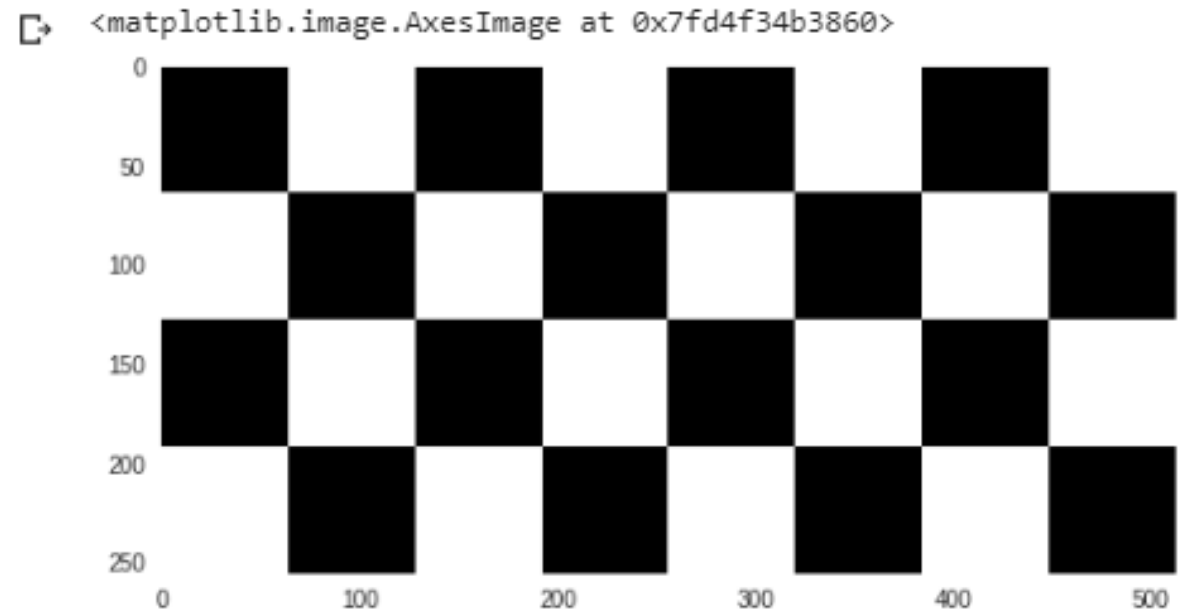
numpy_array = np.ones([height,width,3], dtype=np.uint8)*255

righe = numpy_array.shape[0] // l
colonne = numpy_array.shape[1] // l

for i in range(0, righe*l, l):
    for j in range((i // l) % 2, colonne, 2):
        c = j * l
        numpy_array[i:i+l,c:c+l] = 0

chessboard = Image.fromarray(numpy_array)

plt.grid(b=False)
plt.imshow(chessboard)
```



How to read an image from url

```
from PIL import Image
import matplotlib.pyplot as plt

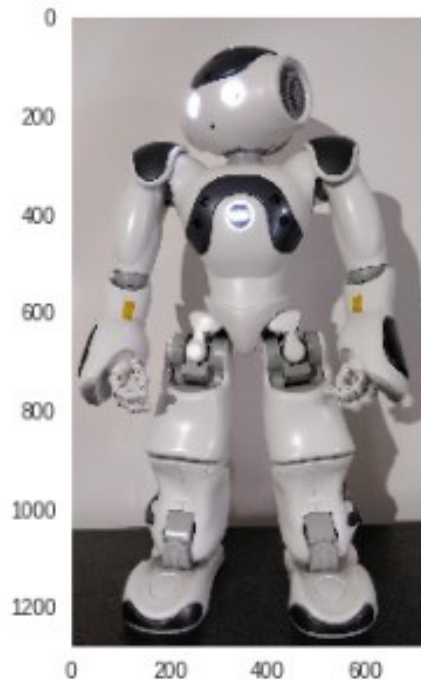
import urllib.request

url = "https://dbloisi.github.io/corsi/images/nao-v6-spqr.jpg"

img = Image.open(urllib.request.urlopen(url))

plt.grid(b=False)
plt.imshow(img)
```

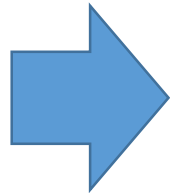
<matplotlib.image.AxesImage at 0x7fd4f5933048>



Esercizio

Utilizzare le librerie NumPy, Matplotlib e Pillow per effettuare il cropping dell'immagine

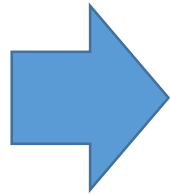
<https://dbloisi.github.io/corsi/images/nao-v6-spqr.jpg>



Esercizio

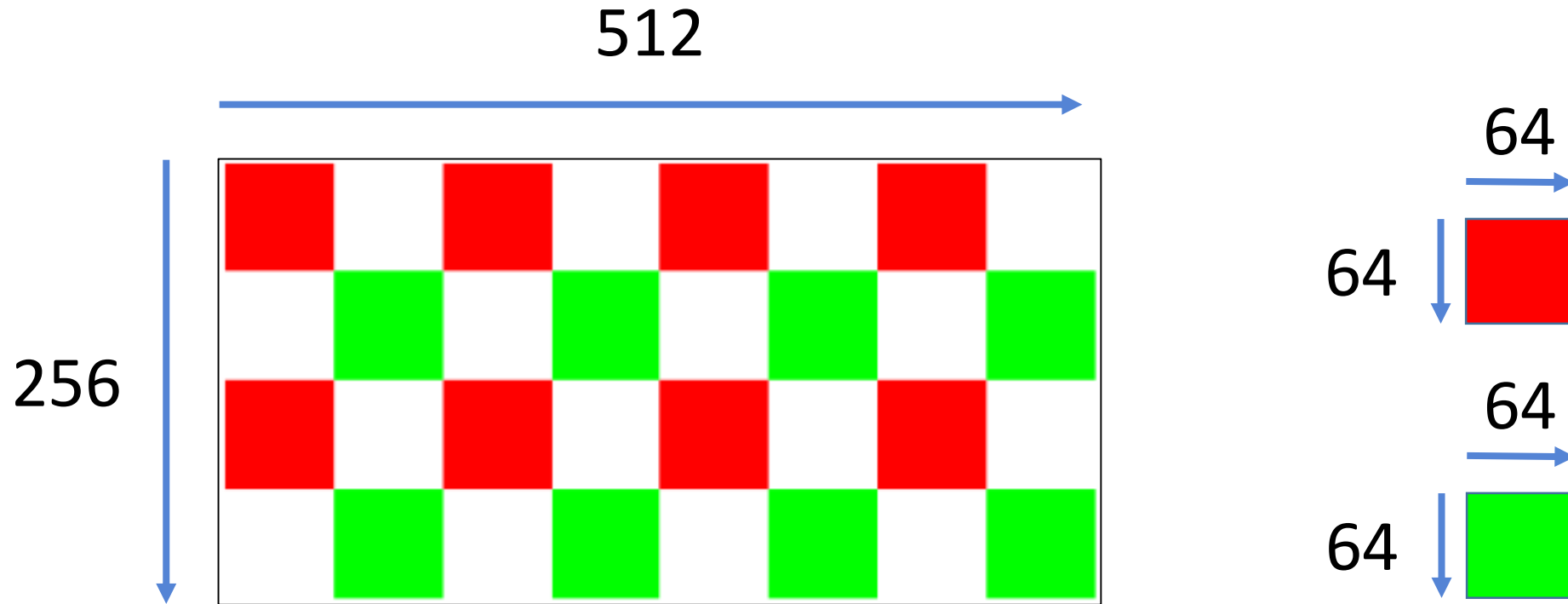
Utilizzare le librerie NumPy, Matplotlib e Pillow per effettuare la rotazione di 30° dell'immagine

<https://dbloisi.github.io/corsi/images/nao-v6-spqr.jpg>



Esercizio

Realizzare una immagine come quella in figura utilizzando le librerie NumPy, Matplotlib e Pillow



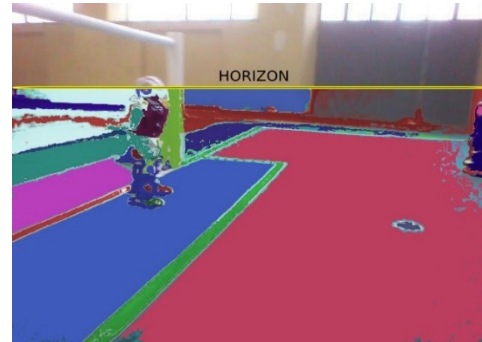
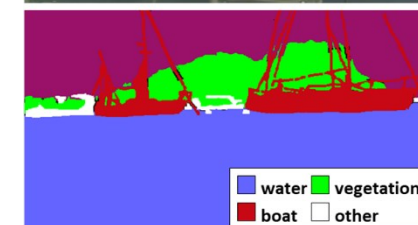
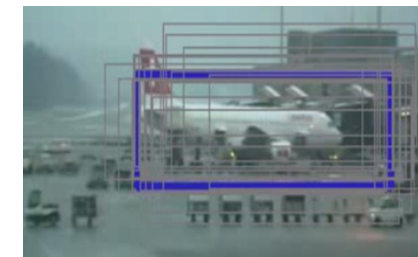


**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi

Processamento delle immagini



Marzo 2019