UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

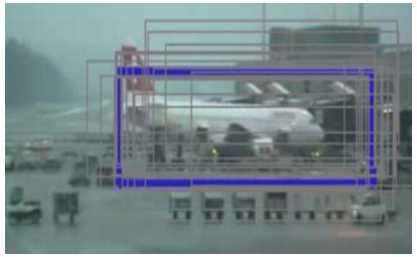


Corso di Visione e Percezione

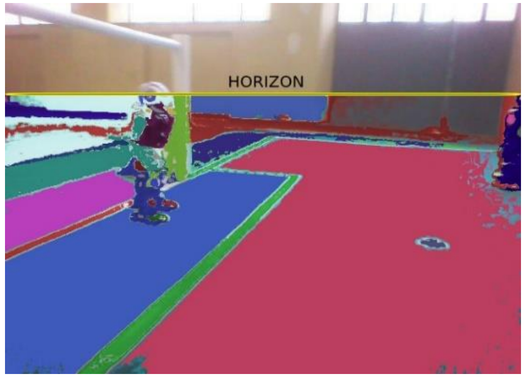
Docente

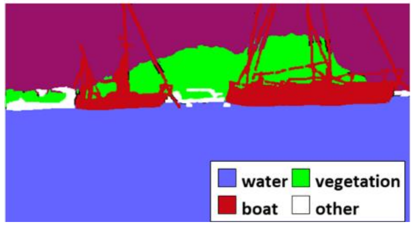
Domenico D. Bloisi

Trasformazioni





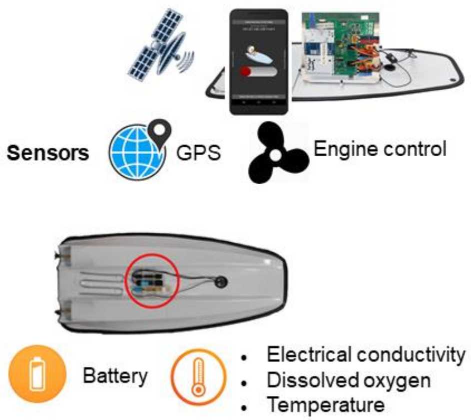


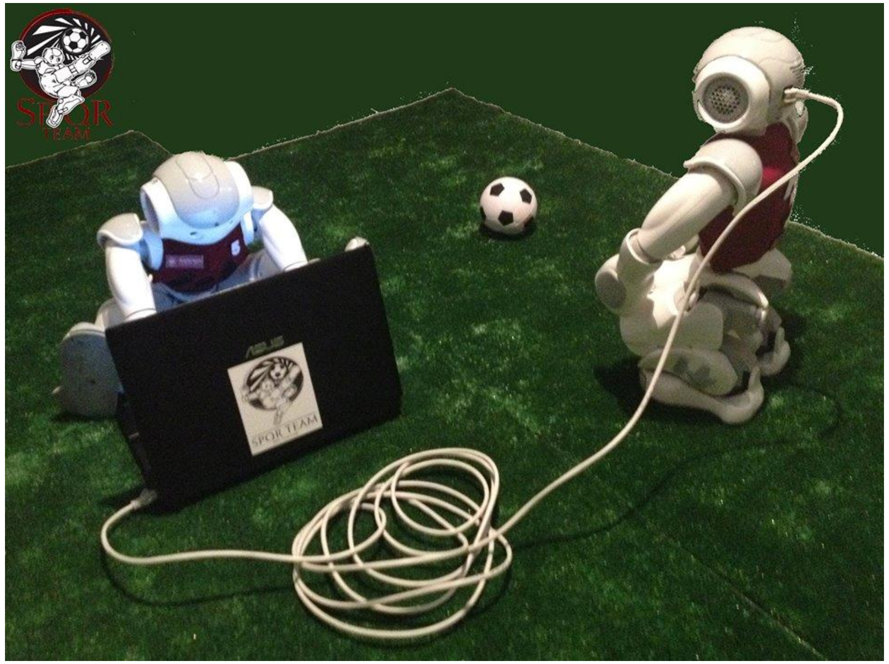



Domenico Daniele Bloisi

- Professore Associato
  Dipartimento di Matematica, Informatica ed Economia
  Università degli studi della Basilicata
  http://web.unibas.it/bloisi

- SPQR Robot Soccer Team
  Dipartimento di Informatica, Automatica e Gestionale Università degli studi di Roma “La Sapienza”
  http://spqr.diag.uniroma1.it





UNIBAS Wolves https://sites.google.com/unibas.it/wolves

- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.

- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome.

Informazioni sul corso

- Home page del corso: https://web.unibas.it/bloisi/corsi/visione-e-percezione.html

- Docente: Domenico Daniele Bloisi

- Periodo: II semestre marzo 2022 – giugno 2022

  - Martedì dalle 15:00 alle 17:00 (Aula Copernico)
  - Mercoledì dalle 8:30 alle 10:30 (Aula Copernico)

Ricevimento

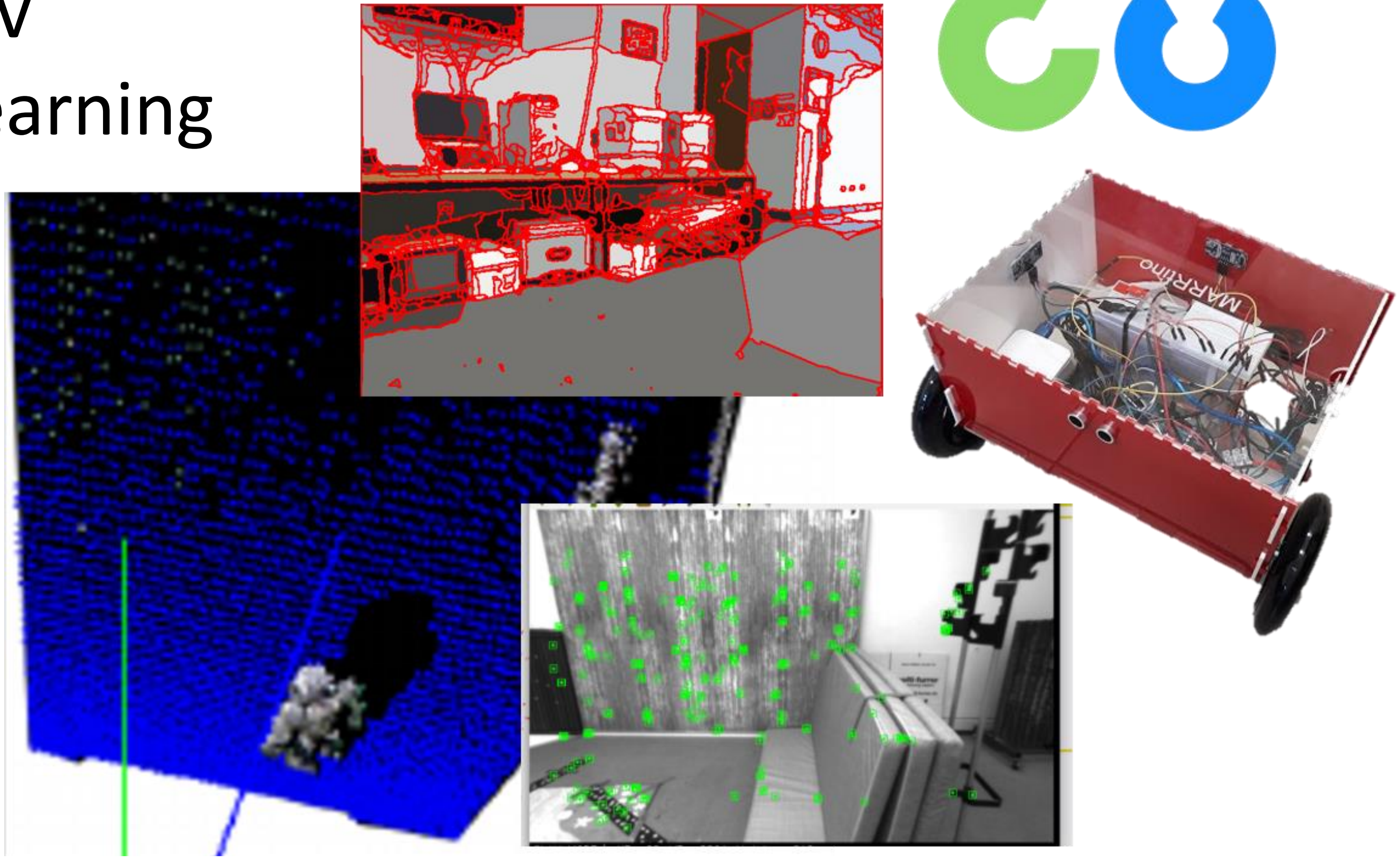
- Durante il periodo delle lezioni:
  Mercoledì dalle 11:00 alle 12:30 → Edificio 3D, II piano, stanza 15
  Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni
- Al di fuori del periodo delle lezioni:
  da concordare con il docente tramite email

Per prenotare un appuntamento inviare una email a

domenico.bloisi@unibas.it

Programma – Visione e Percezione

- Introduzione al linguaggio Python
- Elaborazione delle immagini con Python
- Percezione 2D – OpenCV
- Introduzione al Deep Learning
- ROS
- Il paradigma publisher and subscriber
- Simulatori
- Percezione 3D - PCL



Grayscale image

http://dept.me.umn.edu/courses/me5286/vision/VisionNotes/2018/ME5286-ComputerVision-Lecture3.pdf

Immagini a colori

- Simplified object extraction and identification
- Human vision: ~10 million of distinguishable colors
- Digital RGB representation: 256 x 256 x 256 colors per pixel → 16 million possible colors

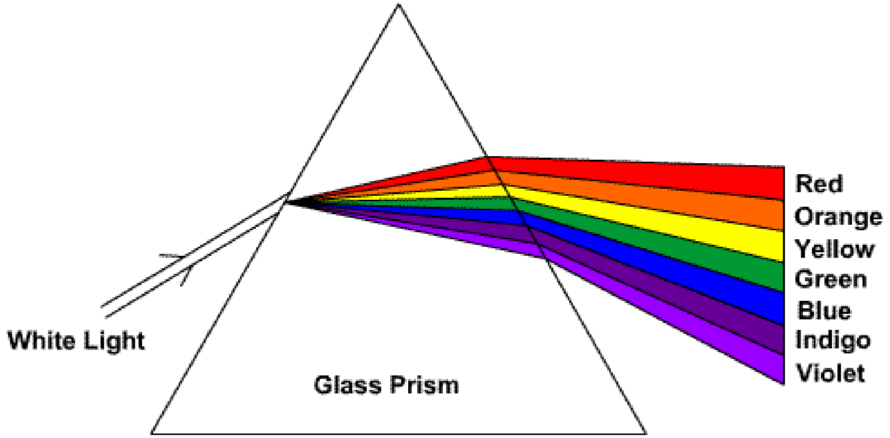
http://dept.me.umn.edu/courses/me5286/vision/VisionNotes/2018/ME5286-ComputerVision-Lecture3.pdf

Color image

http://dept.me.umn.edu/courses/me5286/vision/VisionNotes/2018/ME5286-ComputerVision-Lecture3.pdf

Color spectrum

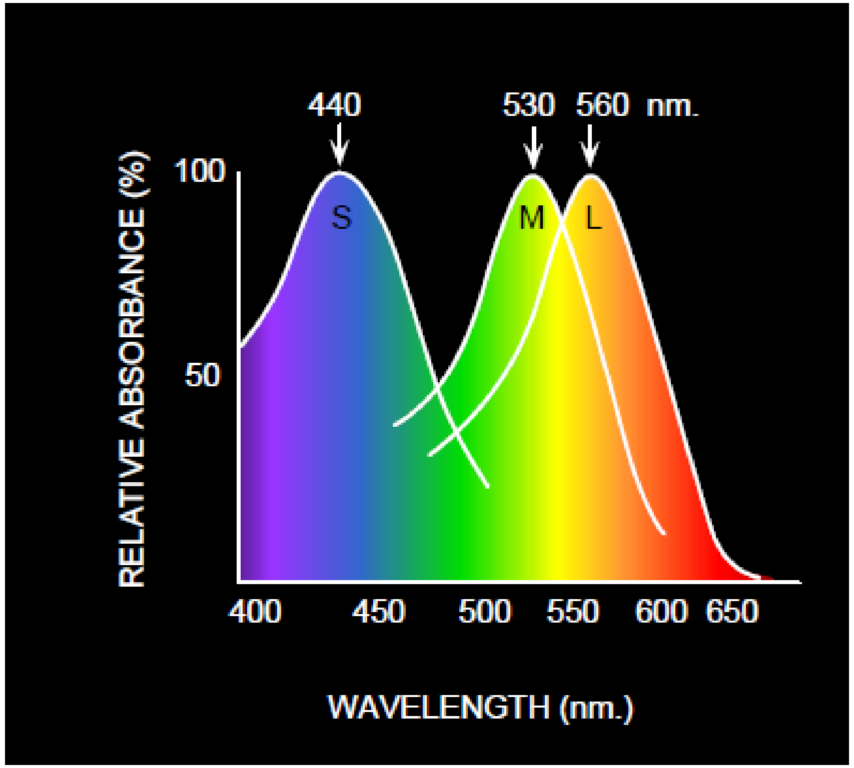
White light with a prism (1666, Newton)



https://www.biography.com/news/how-isaac-newton-changed-our-world

Percezione dei colori

- Approximately 6 million cones in the human eye
- Three different types of cones
- Each cone has a special pigment making it sensitive to specific ranges of wavelengths:
  - Short (S) corresponds to blue
  - Medium (M) corresponds to green
  - Long (L) corresponds to red



http://dept.me.umn.edu/courses/me5286/vision/VisionNotes/2018/ME5286-ComputerVision-Lecture3.pdf

Colori primari

- Color representation is based on the theory of T. Young (1802) which states that any color can be produced by mixing three primary colors C_1, C_2, C_3:

$$C = aC_1 + bC_2 + cC_3$$

- It is therefore possible to characterize a psycho-visual color by specifying the amounts of three primary colors: red, green, and blue, mixed together

- This leads to the standard RGB space used in television, computer monitors, LED screens, etc

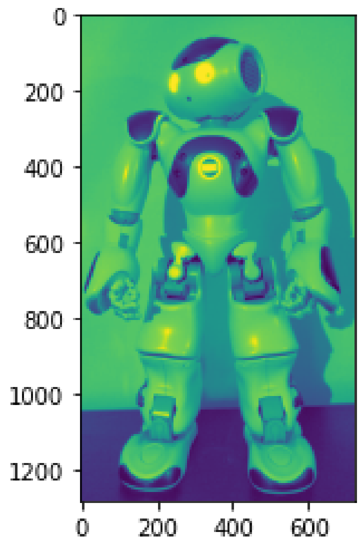
http://dept.me.umn.edu/courses/me5286/vision/VisionNotes/2018/ME5286-ComputerVision-Lecture3.pdf

Esercizio 1

Aprire l’immagine a colori
https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg
e trasformarla in grayscale

Esercizio 1 - soluzione

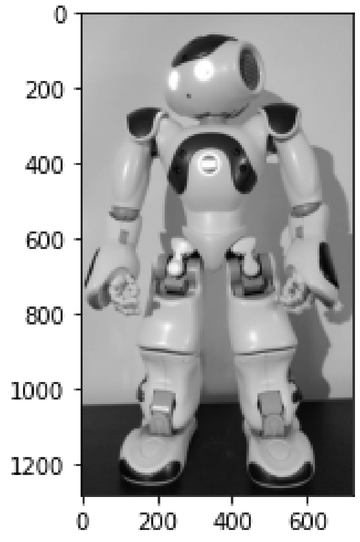
```
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"

img = Image.open(urlopen(url))

gray_img = img.convert("L")

_ = plt.imshow(gray_img)
```



Questa visualizzazione non sembra corretta!

Esercizio 1 - soluzione

```
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"

img = Image.open(urlopen(url))

gray_img = img.convert("L")

_ = plt.imshow(gray_img, cmap="gray")
```



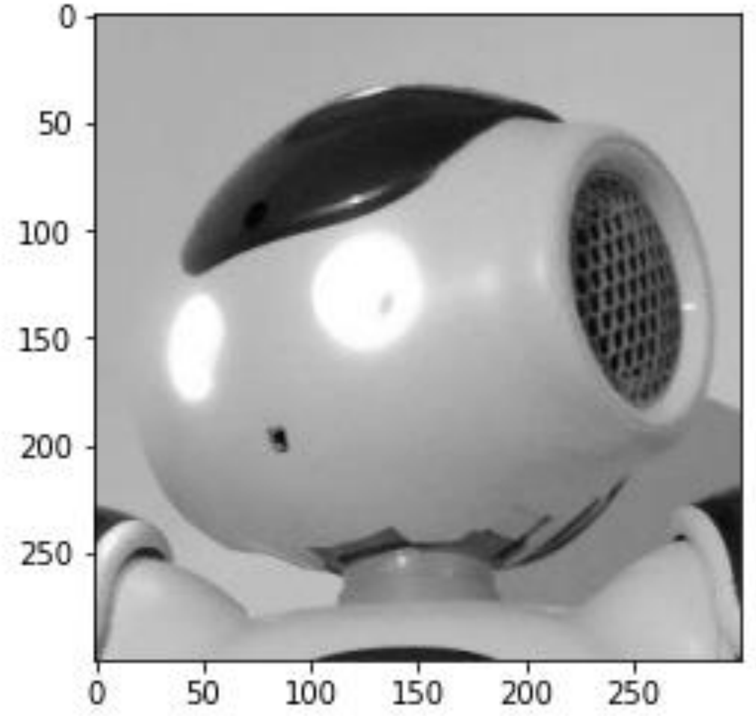
Questa è la visualizzazione corretta!

Esercizio 2

Costruire una immagine 300x300 contenente solo la testa del robot a partire dalla versione grayscale dell'immagine https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg ottenuta nell'esercizio precedente

Esercizio 2 - soluzione

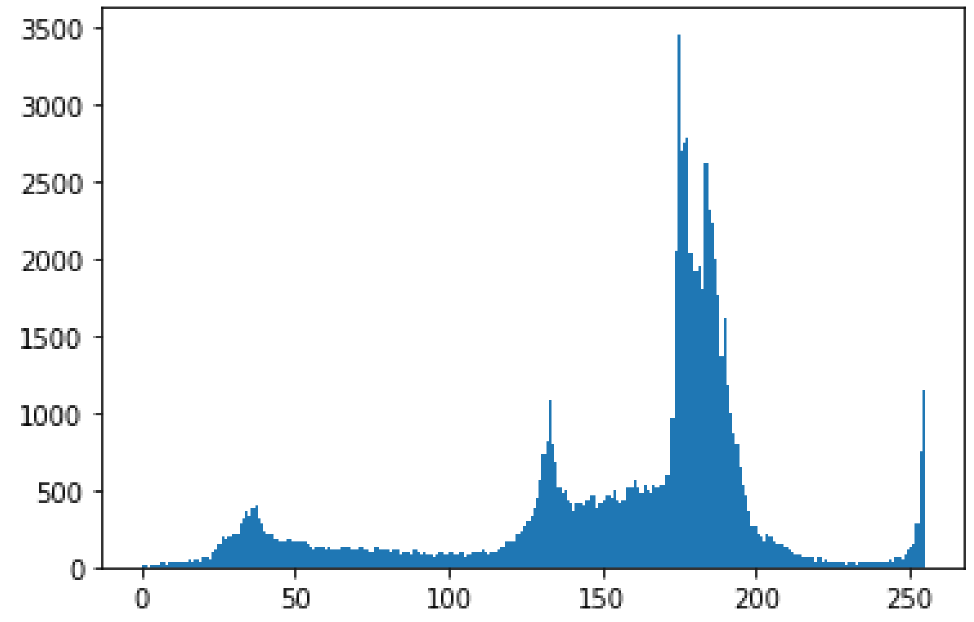
```
ROI = (200,25,500,325) #left, upper, right, and lower pixel coordinate

face = gray_img.crop(ROI)

_ = plt.imshow(face, cmap="gray")
```



Istogramma di una immagine

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

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"
gray_img = Image.open(urlopen(url)).convert("L")
face = np.array(gray_img.crop((200,25,500,325)))

plt.hist(face.flatten(), 256)
plt.show()
```

`ndarray.flatten` returns a copy of the array collapsed into one dimension.



Scipy library

Scipy library (https://www.scipy.org/scipylib) è una libreria contenente l'implementazione di algoritmi e tool matematici compatibili con NumPy

Scipy sub-modules

La Scipy library contiene diversi sub-moduli specializzati per particolari compiti

`scipy.ndimage` è il package per il processamento delle immagini

https://docs.scipy.org/doc/scipy/reference/ndimage.html

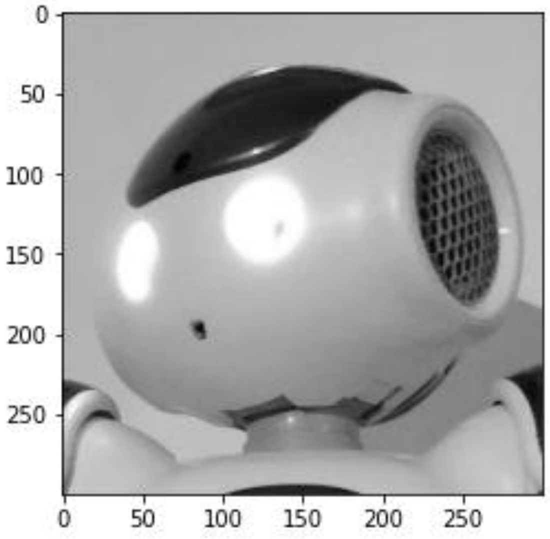
- Clustering package (scipy.cluster)
- Constants (scipy.constants)
- Discrete Fourier transforms (scipy.fft)
- Legacy discrete Fourier transforms (scipy.fftpack)
- Integration and ODEs (scipy.integrate)
- Interpolation (scipy.interpolate)
- Input and output (scipy.io)
- Linear algebra (scipy.linalg)
- Miscellaneous routines (scipy.misc)
- Multi-dimensional image processing (scipy.ndimage)
- Orthogonal distance regression (scipy.odr)
- Optimization and Root Finding (scipy.optimize)
- Signal processing (scipy.signal)
- Sparse matrices (scipy.sparse)
- Sparse linear algebra (scipy.sparse.linalg)
- Compressed Sparse Graph Routines (scipy.sparse.csgraph)
- Spatial algorithms and data structures (scipy.spatial)
- Special functions (scipy.special)
- Statistical functions (scipy.stats)
- Statistical functions for masked arrays (scipy.stats.mstats)
- Low-level callback functions

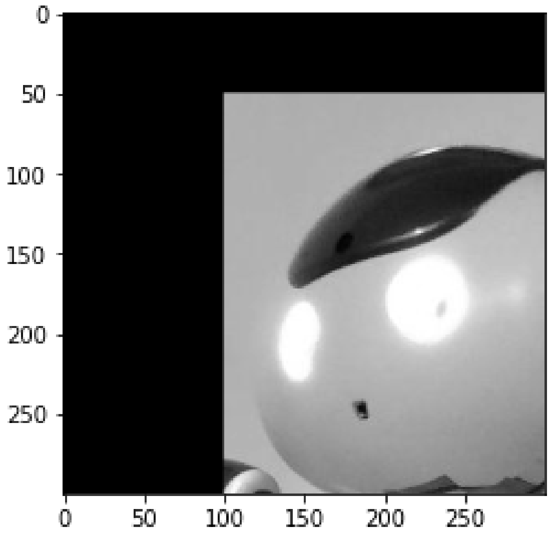
Trasformazioni geometriche in Scipy

- Shift
- Rotazione
- Zoom
- Flip

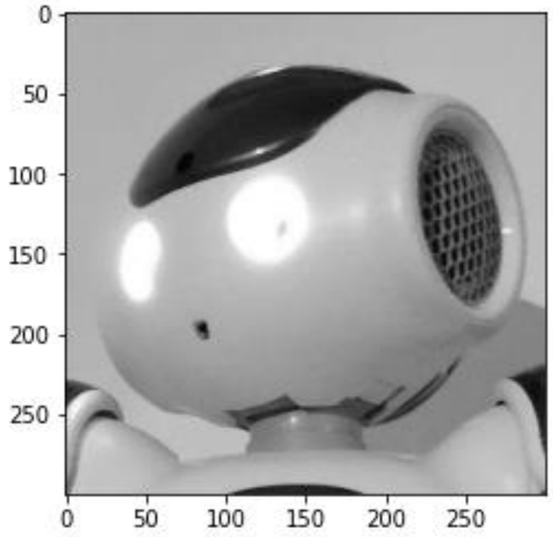
Shift

```
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt
from scipy import ndimage

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"
gray_img = Image.open(urlopen(url)).convert("L")
face = gray_img.crop((200,25,500,325))

shifted_face = ndimage.shift(face, (50, 100))

_ = plt.imshow(shifted_face, cmap="gray")
```

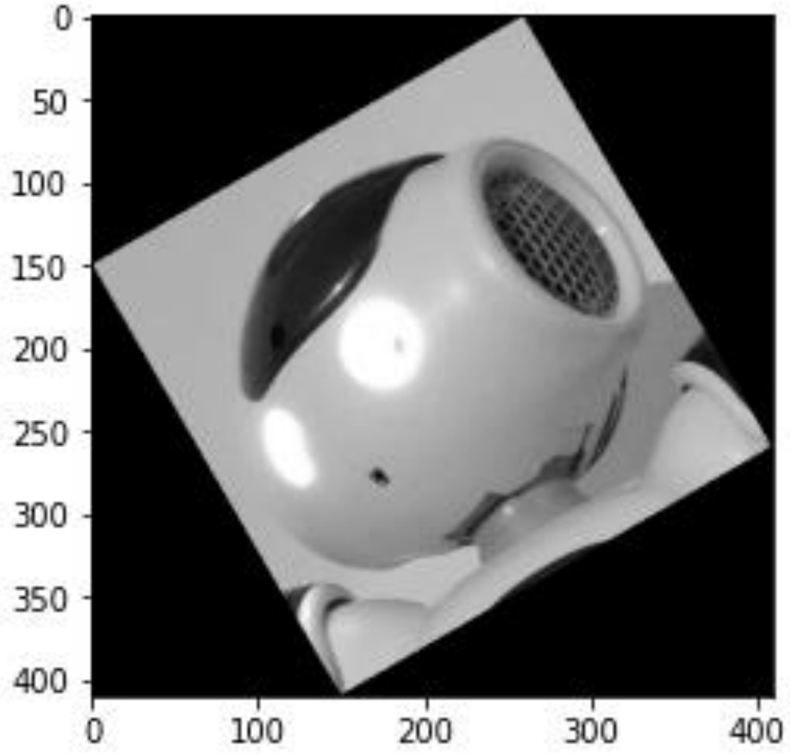




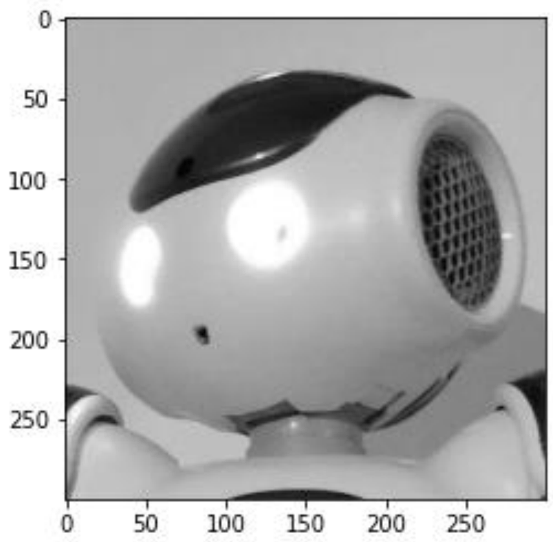
Rotazione

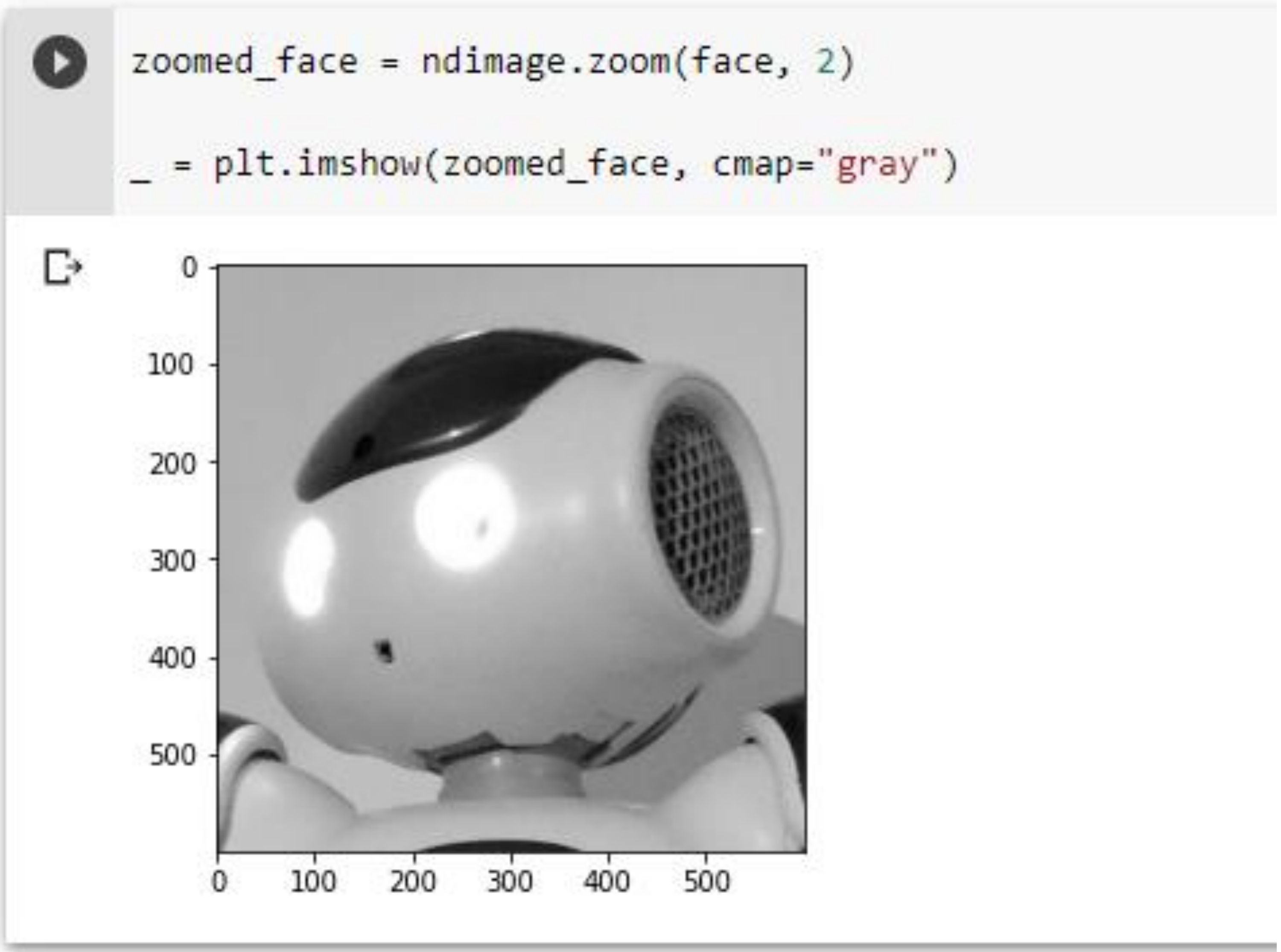



```
rotated_face = ndimage.rotate(face, 30)

_ = plt.imshow(rotated_face, cmap="gray")
```

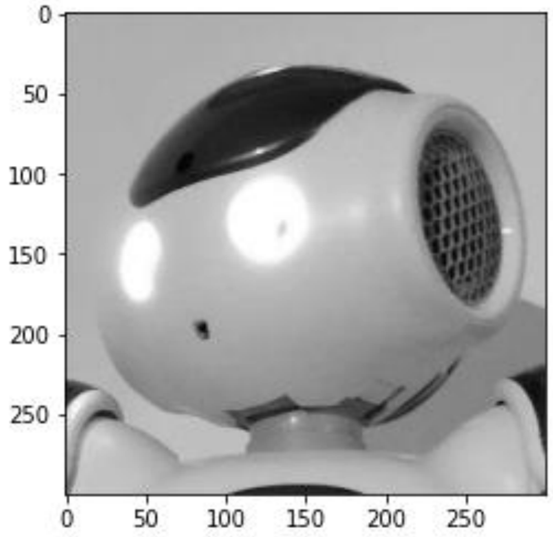


Zoom

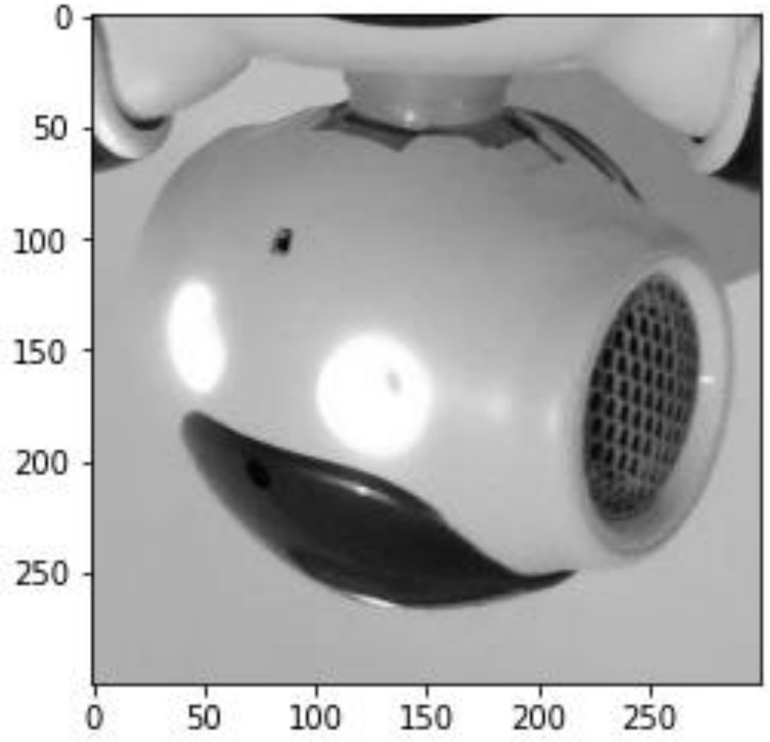

0
50
100
150
200
250
0
50
100
150
200
250


zoomed_face = ndimage.zoom(face, 2)
_ = plt.imshow(zoomed_face, cmap="gray")
0
100
200
300
400
500
0
100
200
300
400
500

Flip

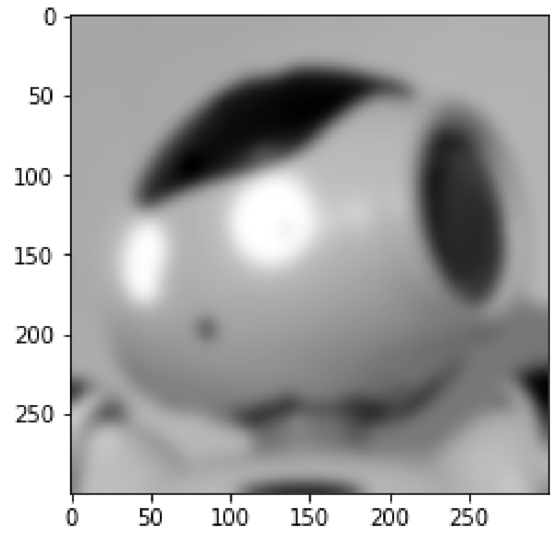



```python
# up <-> down flip
flip_ud_face = np.flipud(face)

_ = plt.imshow(flip_ud_face, cmap="gray")
```



Filtering in Scipy

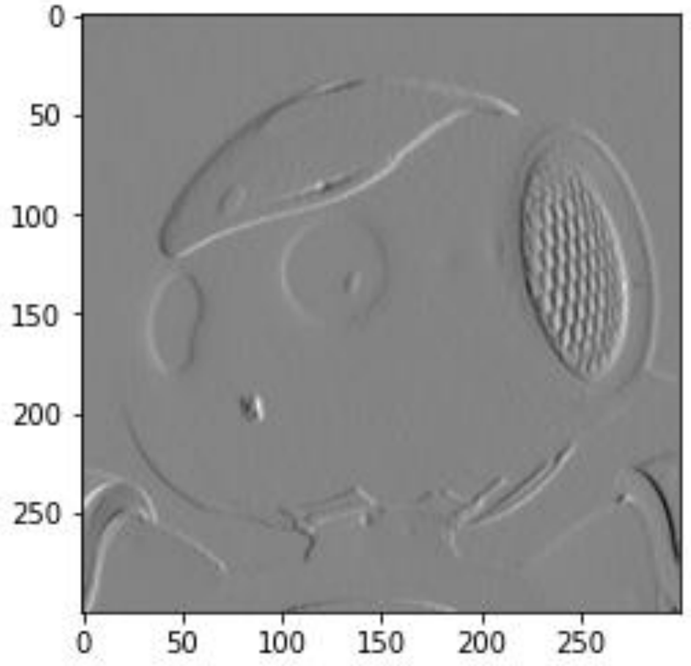
```
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt
from scipy.ndimage import filters

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"
gray_img = Image.open(urlopen(url)).convert("L")
face = gray_img.crop((200,25,500,325))

blurred_face = filters.gaussian_filter(face,5)

_ = plt.imshow(blurred_face, cmap="gray")
```



Il secondo parametro di `gaussian_filter()` è la standard deviation

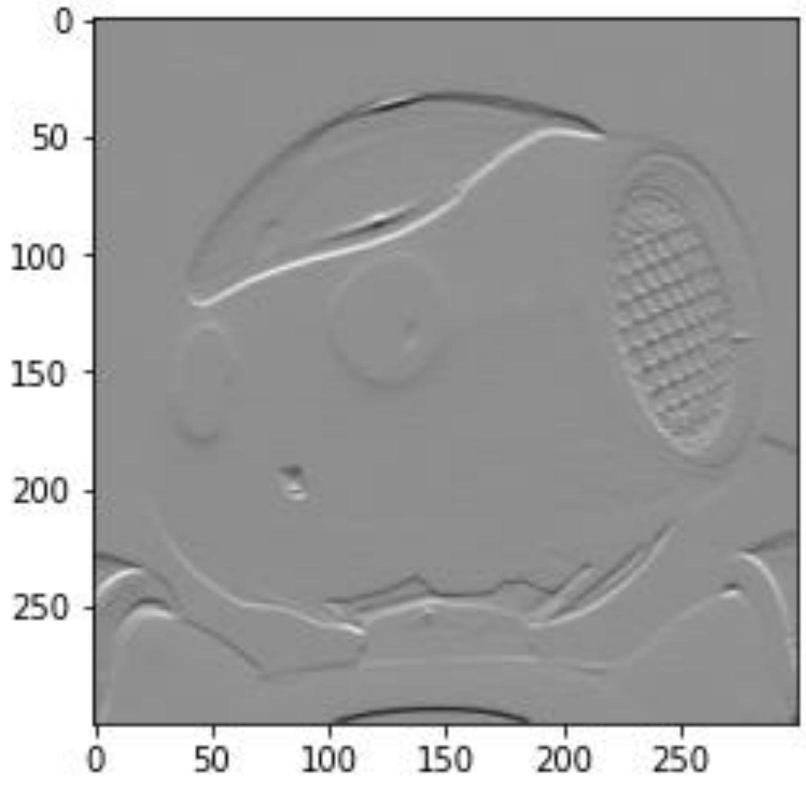
Derivate in Scipy

```
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt
from scipy.ndimage import filters
import numpy as np

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"
gray_img = Image.open(urlopen(url)).convert("L")
face = np.array(gray_img.crop((200,25,500,325)))

dx = np.zeros(face.shape)
filters.sobel(face,1,dx)
_ = plt.imshow(dx, cmap="gray")
```



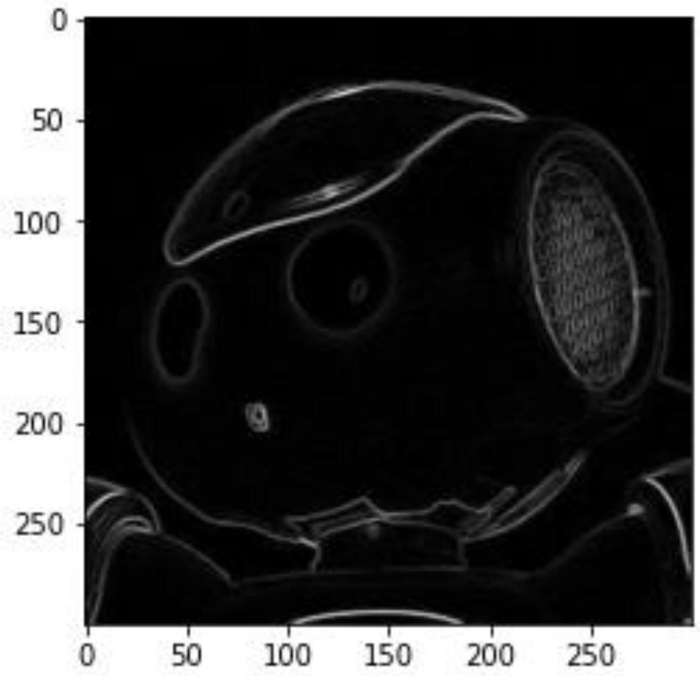
Derivate in Scipy

```
dy = np.zeros(face.shape)
filters.sobel(face,0,dy)
_ = plt.imshow(dy, cmap="gray")
```

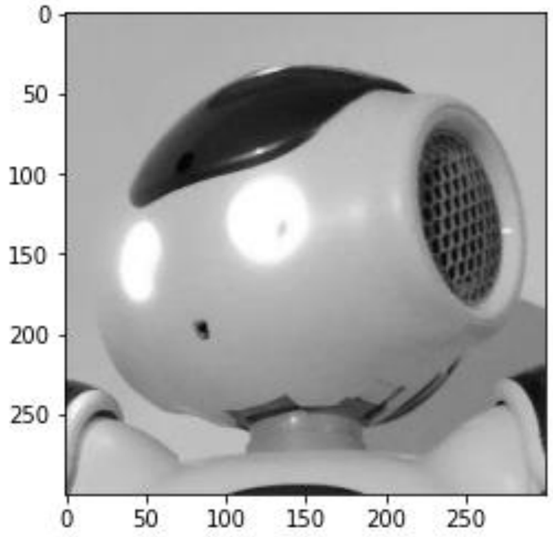


Gradient magnitude

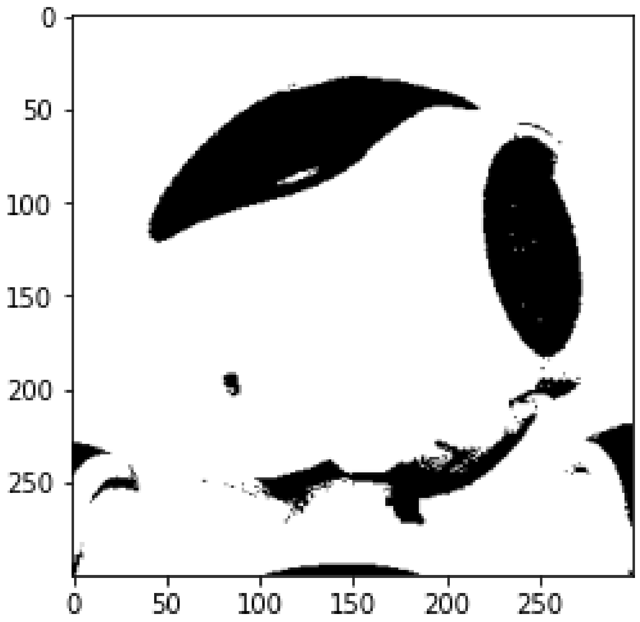
```
from numpy import sqrt

magnitude = sqrt(dx**2+dy**2)

_ = plt.imshow(magnitude, cmap="gray")
```



Thresholding




```python
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt
import numpy as np

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"
gray_img = Image.open(urlopen(url)).convert("L")
face = np.array(gray_img.crop((200,25,500,325)))

t = 120
mask = face > t
_ = plt.imshow(mask, cmap="gray")
```



Otsu Thresholding

```
import matplotlib.pyplot as plt
from skimage import data
from skimage import filters
from skimage import exposure
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt
import numpy as np

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"
gray_img = Image.open(urlopen(url)).convert("L")
face = np.array(gray_img.crop((200,25,500,325)))

val = filters.threshold_otsu(face)
print("val: %d" % val)
```

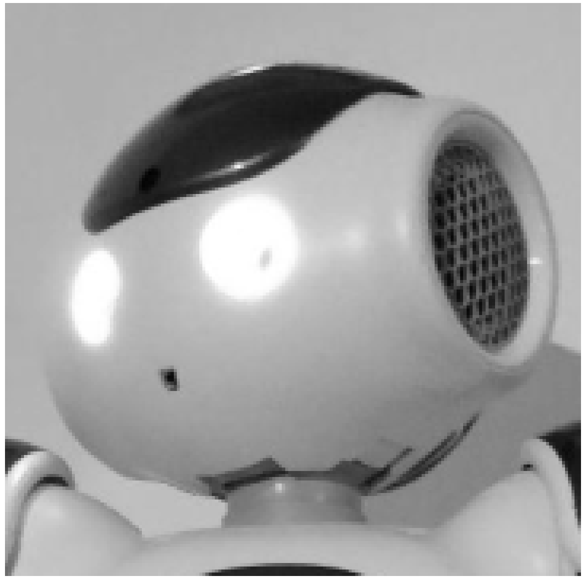
Assumption: the image histogram is bimodal

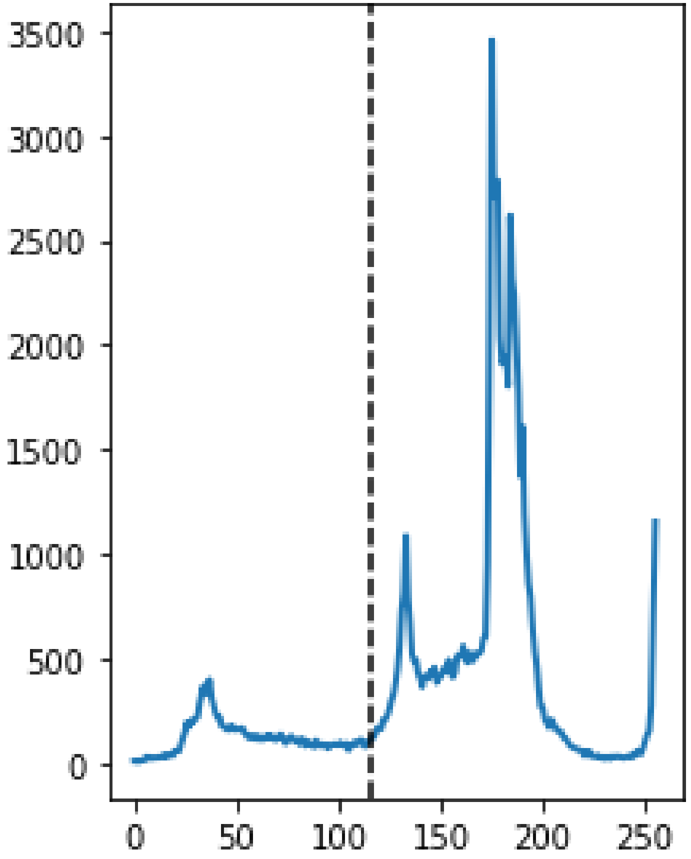
Otsu Thresholding

```
hist, bins_center = exposure.histogram(face)

plt.figure(figsize=(9, 4))
plt.subplot(131)
plt.imshow(face, cmap='gray')
plt.axis('off')
plt.subplot(132)
plt.imshow(face < val, cmap='gray')
plt.axis('off')
plt.subplot(133)
plt.plot(bins_center, hist, lw=2)
plt.axvline(val, color='k', ls='--')

plt.tight_layout()
plt.show()
```

Otsu Thresholding

```
val: 116
```

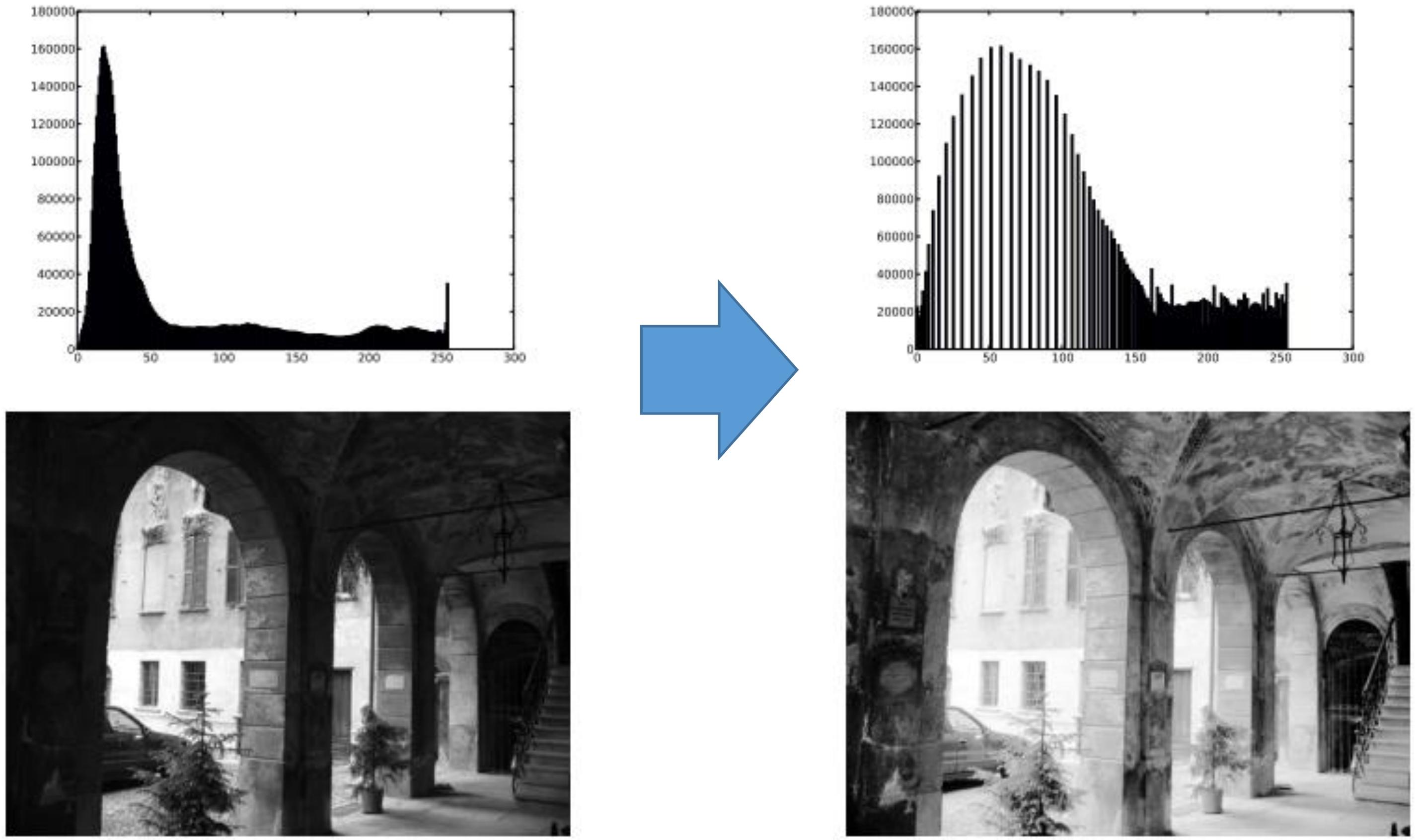




Histogram Equalization

Histogram equalization flattens the graylevel histogram of an image so that all intensities are as equally common as possible.

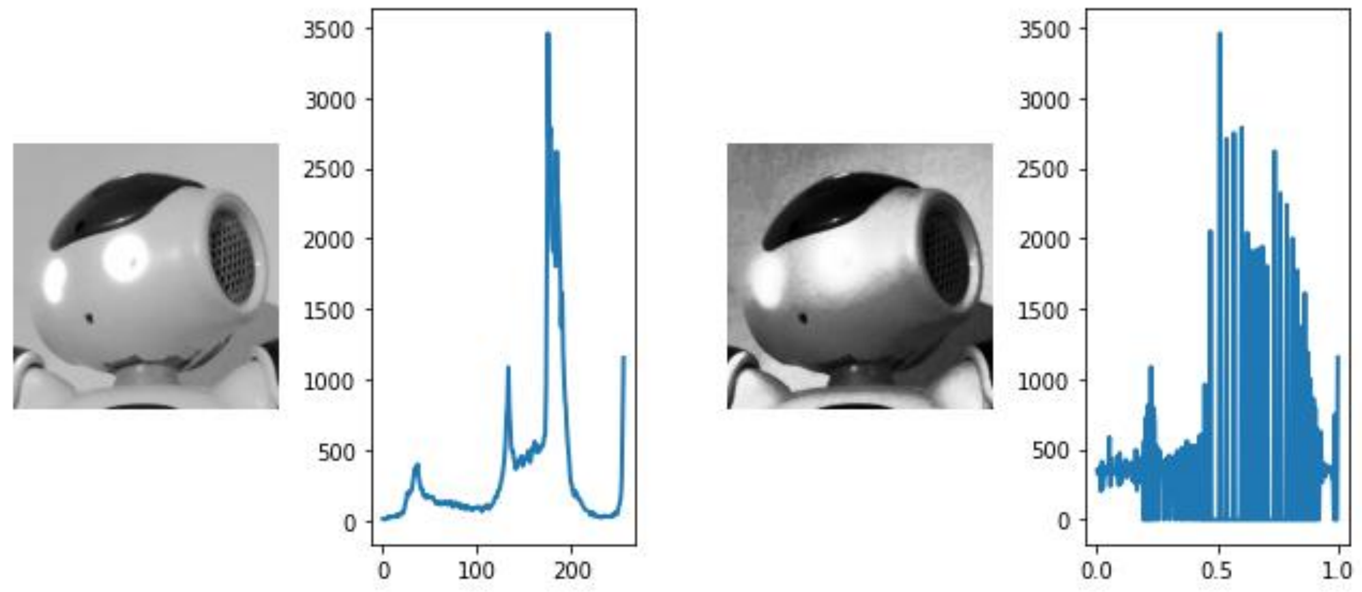
This is often a good way to normalize image intensity before further processing and also a way to increase image contrast.



http://programmingcomputervision.com/downloads/ProgrammingComputerVision_CCdraft.pdf

Histogram Equalization

```
equalized_face = exposure.equalize_hist(face)

hist_eq, bins_center_eq = exposure.histogram(equalized_face)

plt.figure(figsize=(9, 4))
plt.subplot(141)
plt.imshow(face, cmap='gray')
plt.axis('off')
plt.subplot(142)
plt.plot(bins_center, hist, lw=2)
plt.subplot(143)
plt.imshow(equalized_face, cmap='gray')
plt.axis('off')
plt.subplot(144)
plt.plot(bins_center_eq, hist_eq, lw=2)

plt.tight_layout()
plt.show()
```



Mathematical Morphology

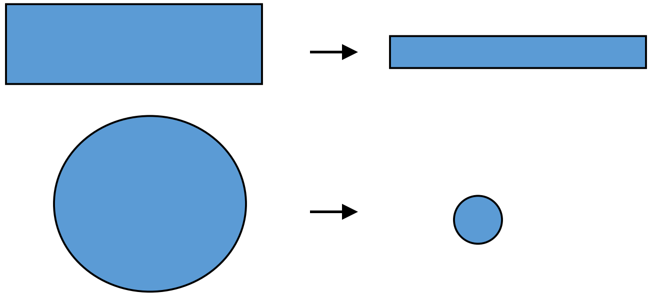
- Erosion
- Dilation
- Closing
- Opening

Erosion

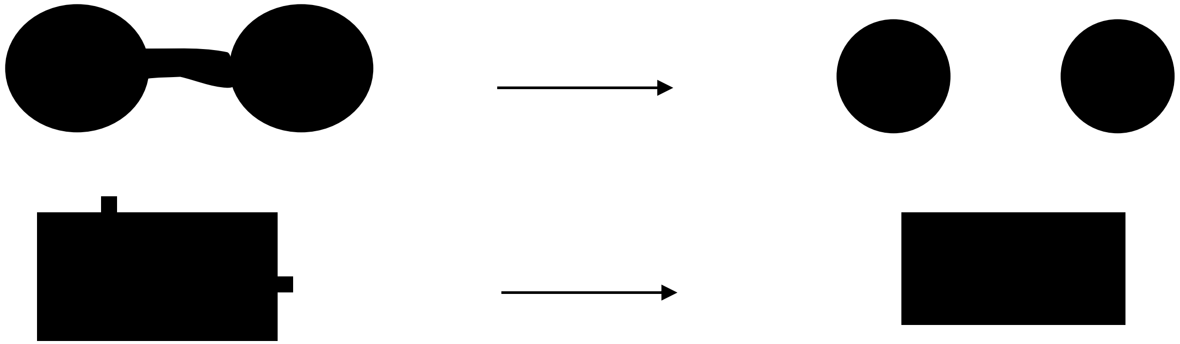
Erosion shrinks the connected sets of 1s of a binary image.

It can be used for

1. shrinking features



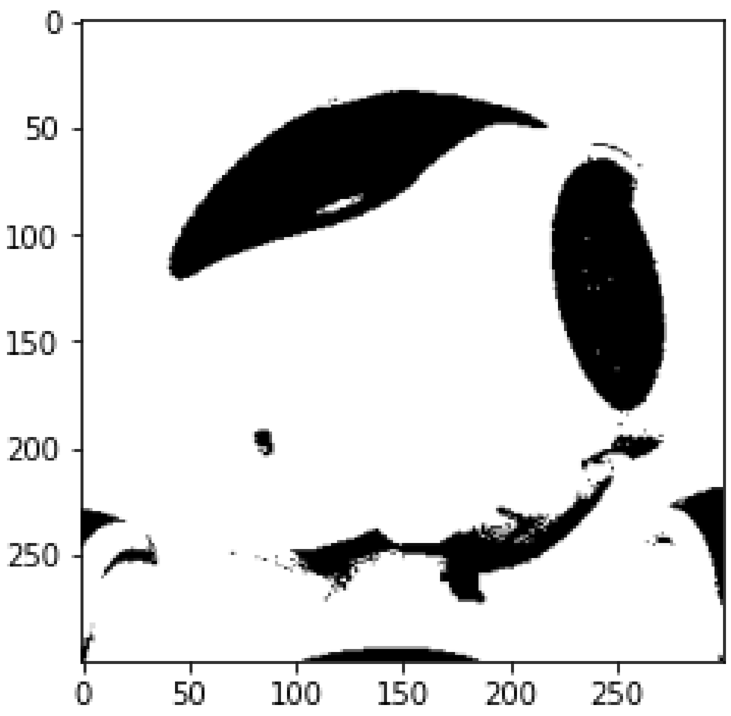
2. Removing bridges, branches and small protrusions

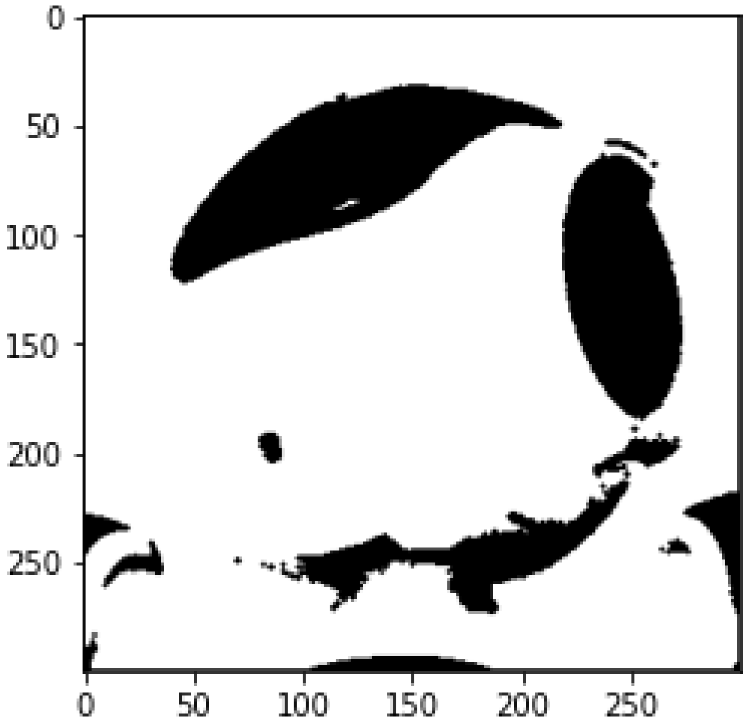


https://courses.cs.washington.edu/courses/cse576/19sp/notes/Basics1_white.pdf

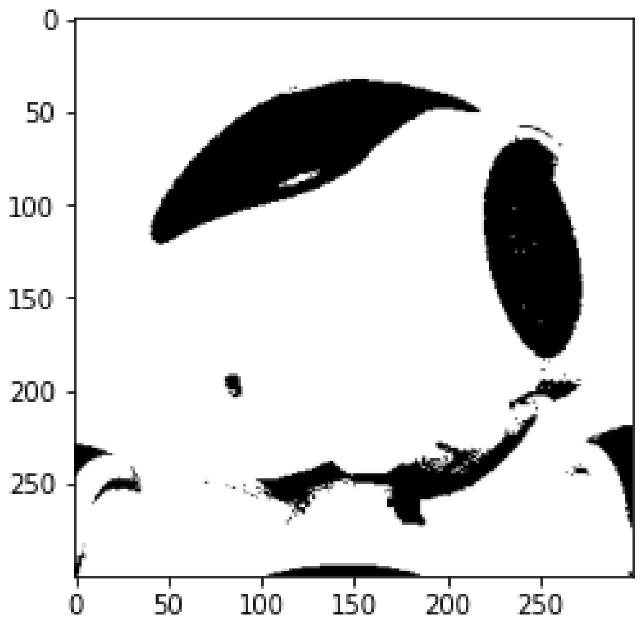
Erosion

```python
from scipy.ndimage import morphology

e = ndimage.binary_erosion(mask)

_ = plt.imshow(e, cmap="gray")
```

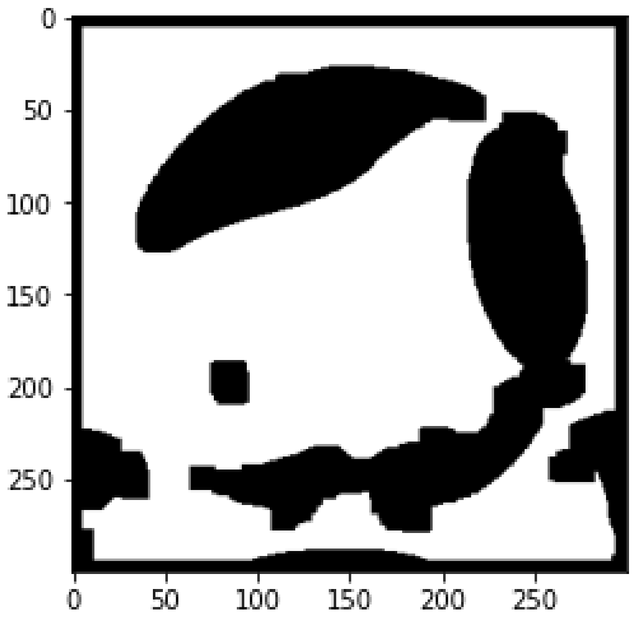




Erosion




```
e2 = ndimage.binary_erosion(mask,structure=np.ones((5,5)),iterations=3)
_ = plt.imshow(e2, cmap="gray")
```

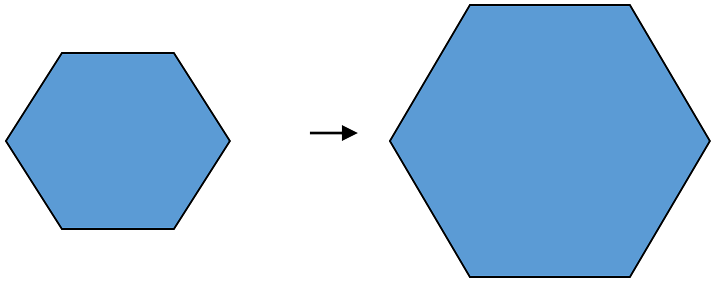


Dilation

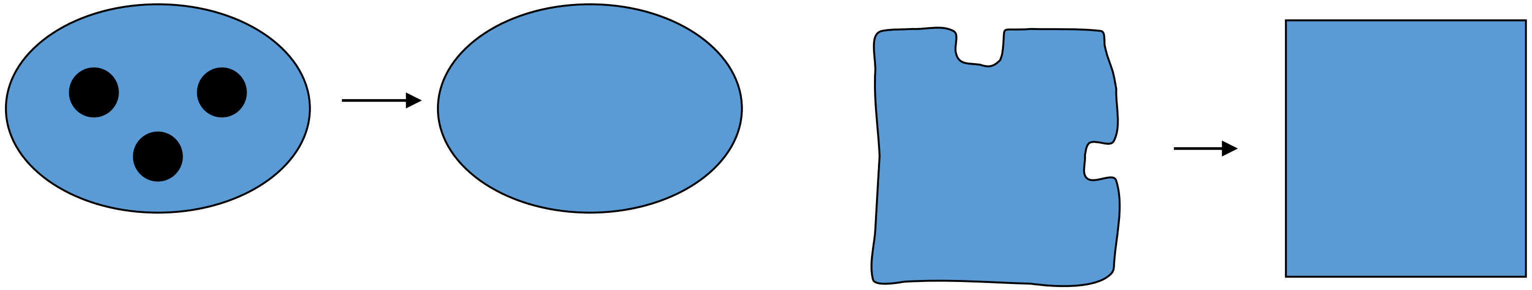
Dilation expands the connected sets of 1s of a binary image.

It can be used for

1. growing features

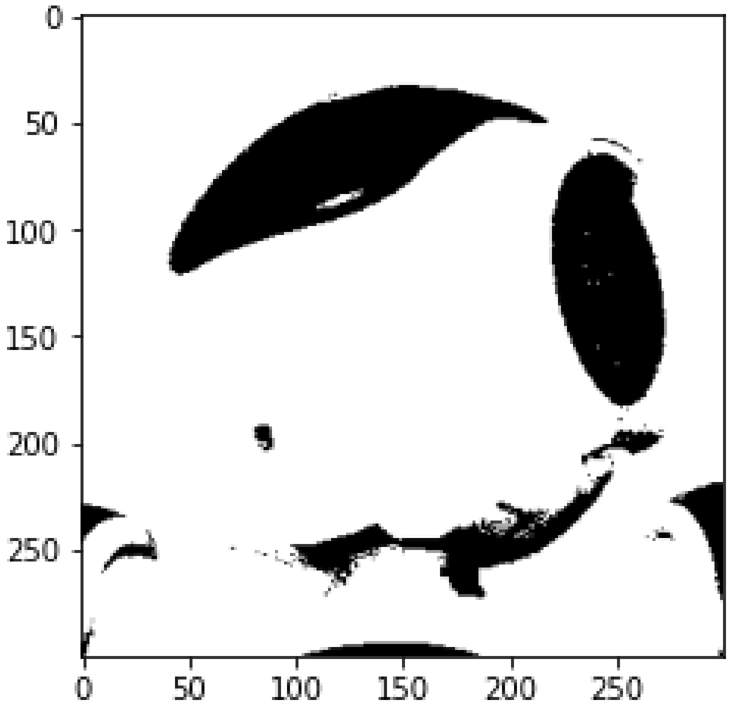


2. filling holes and gaps

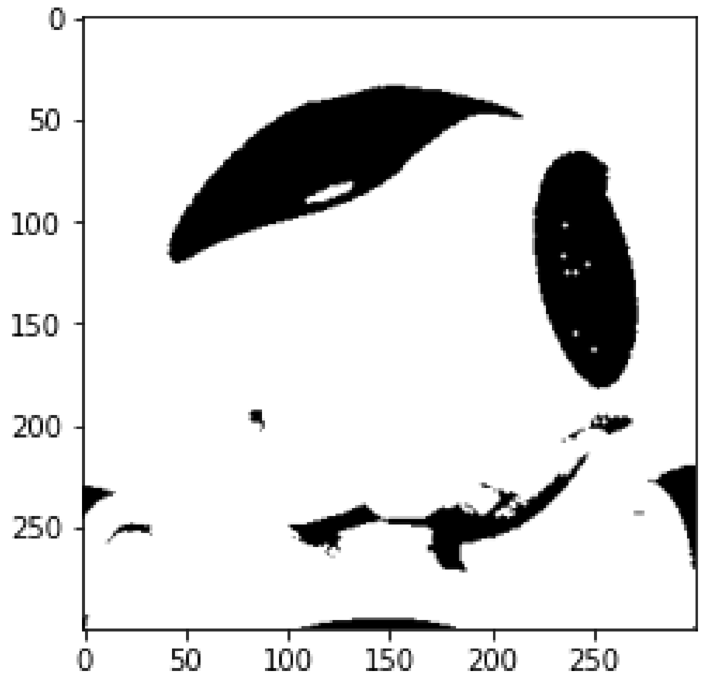


https://courses.cs.washington.edu/courses/cse576/19sp/notes/Basics1_white.pdf

Dilation

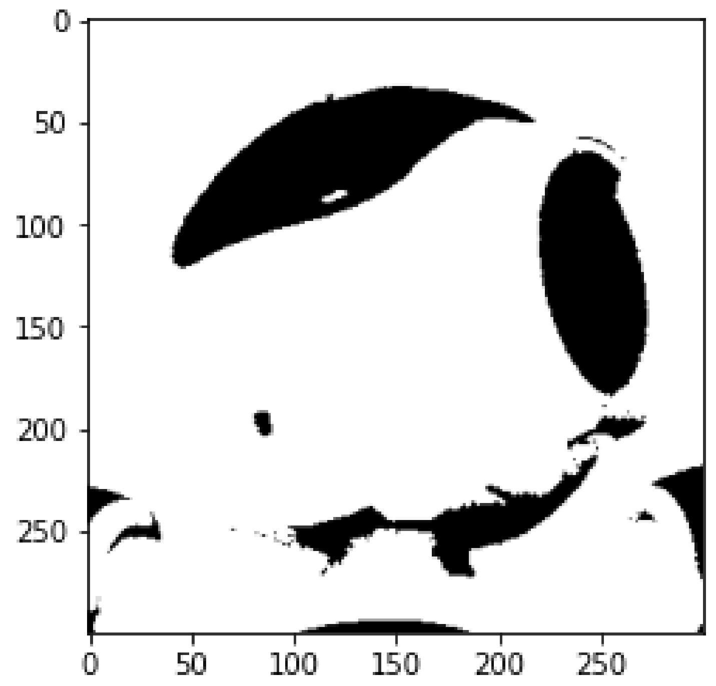



```python
from scipy.ndimage import morphology

d = ndimage.binary_dilation(mask)

_ = plt.imshow(d, cmap="gray")
```



Opening

```
from scipy.ndimage import morphology

o = ndimage.binary_opening(mask)

_ = plt.imshow(o, cmap="gray")
```



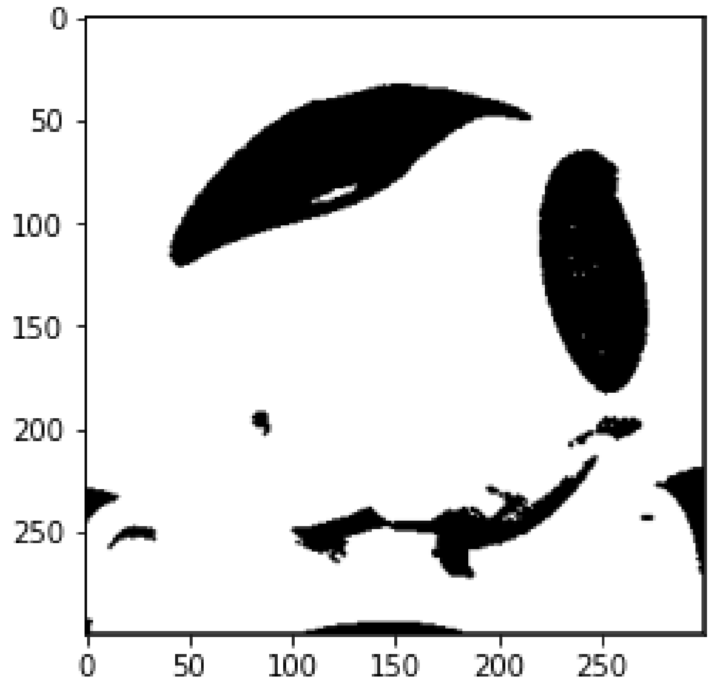
Opening is the compound operation of erosion followed by dilation

Opening is so called because it can open up a gap between objects connected by a thin bridge of pixels. Any regions that have survived the erosion are restored to their original size by the dilation.

https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

Closing

```
from scipy.ndimage import morphology

c = ndimage.binary_closing(mask)

_ = plt.imshow(c, cmap="gray")
```



Closing is the compound operation of dilation followed by erosion

Closing is so called because it can fill holes in the regions while keeping the initial region sizes.

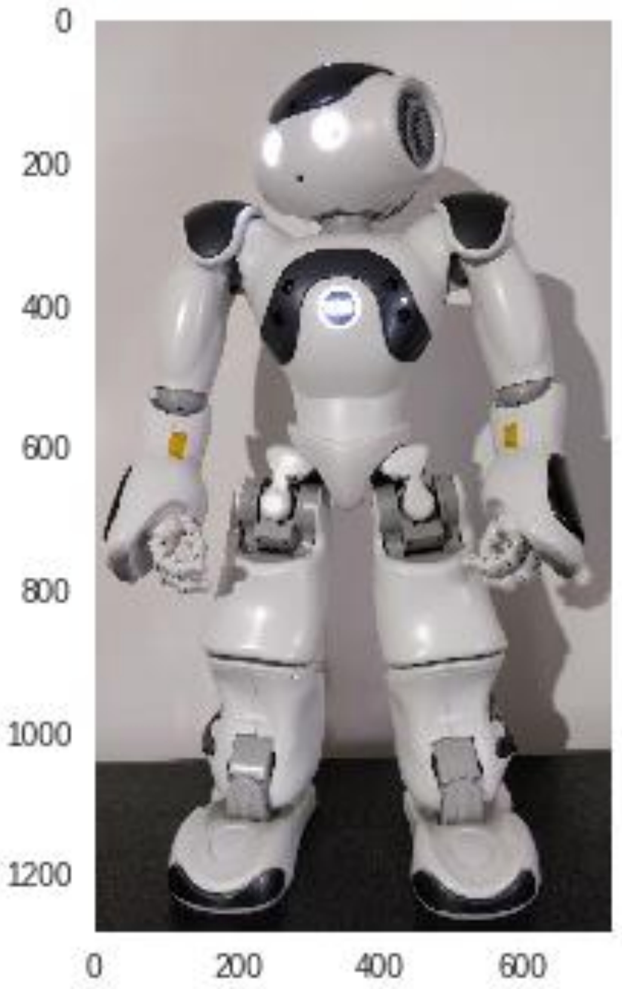
https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

Esercizio 3

Aprire l’immagine JPEG
http://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg
e trasformarla in PNG

Esercizio 3 - soluzione

```python
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))

img.save("nao.png")

!ls

img_png = Image.open("nao.png")

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



Esercizio 4

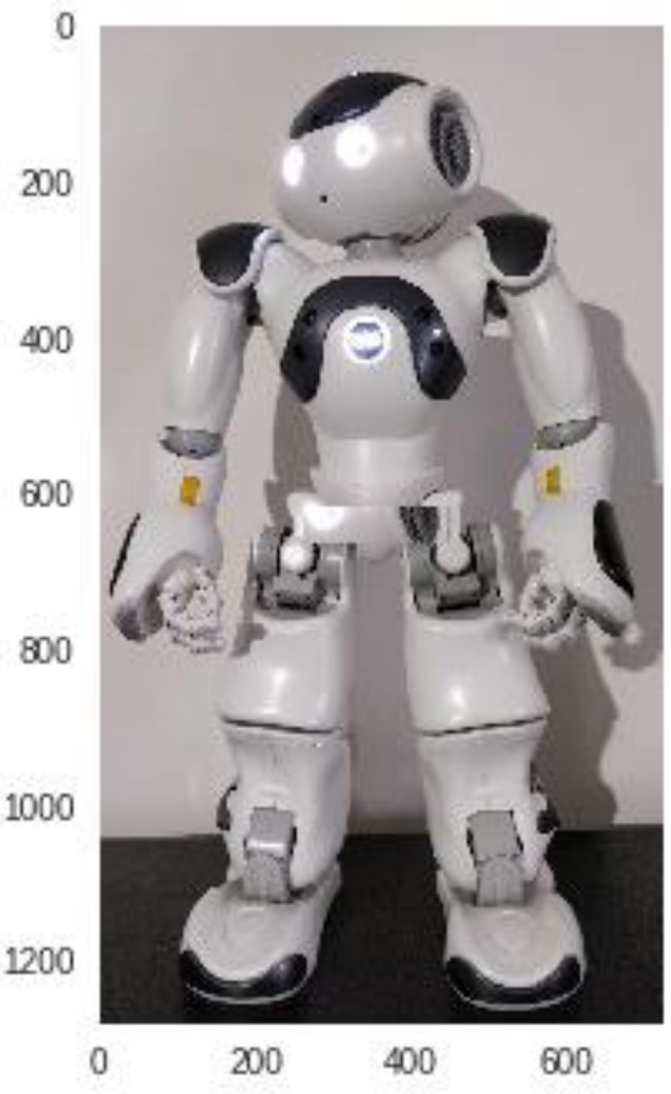
1. Aprire l’immagine a colori
https://dbloisi.github.io/corsi/images/nao-v6-spqr.jpg

2. Estrarre la ROI (300,150,500,200)

3. Incollare la ROI al centro dell’immagine

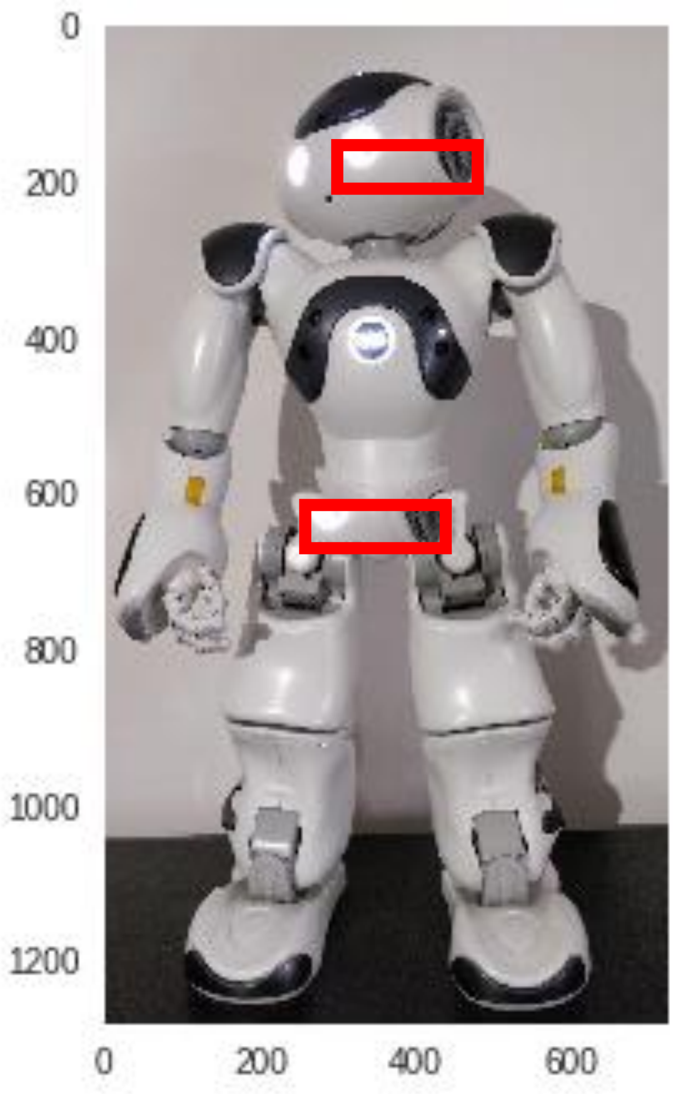
Esercizio 4 - soluzione

```python
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))
print(img.size)

roi = img.crop((300,150,500,200))
print(roi.size)

x = (img.size[0] - roi.size[0]) // 2
y = (img.size[1] - roi.size[1]) // 2

position = (x, y)

img_copy = img.copy()
img_copy.paste(roi, position)

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



Esercizio 4 - soluzione

```
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))
print(img.size)

roi = img.crop((300,150,500,200))
print(roi.size)

x = (img.size[0] - roi.size[0]) // 2
y = (img.size[1] - roi.size[1]) // 2

position = (x, y)

img_copy = img.copy()
img_copy.paste(roi, position)

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



Esercizio 5

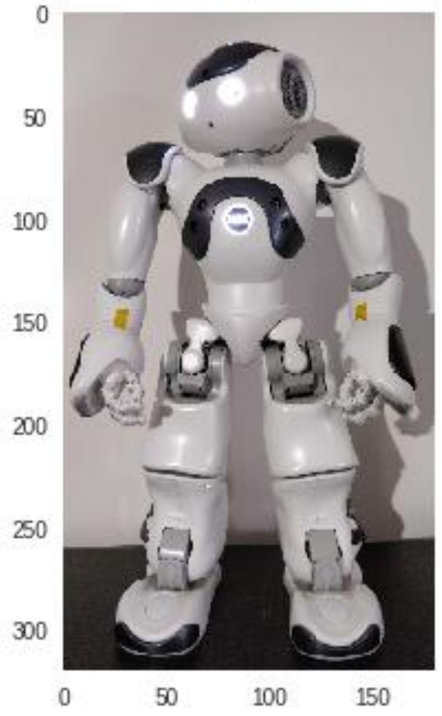
1. Aprire l'immagine a colori
   https://dbloisi.github.io/corsi/images/nao-v6-spqr.jpg

2. Salvare una nuova immagine che abbia dimensioni pari a ¼ dell'originale

Esercizio 5 - soluzione

```python
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_copy)

resized_img = img.resize((img.size[0] // 4, img.size[1] // 4))
resized_img.save('resized.jpg')

print(img.size)
print(resized_img.size)

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

```
(720, 1280)
(180, 320)
<matplotlib.image.AxesImage at 0x7fc05bf212e8>
```



Esercizio 6

1. Aprire l’immagine a colori
   https://dbloisi.github.io/corsi/images/nao-v6-spqr.jpg

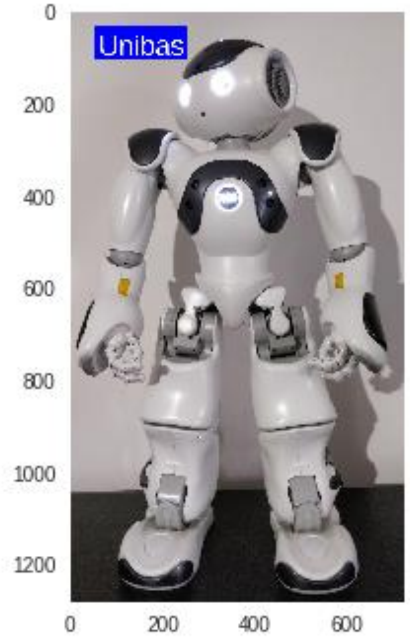
2. Inserire nell’angolo in alto a sinistra dell’immagine la stringa 'Unibas' così come mostrata sotto

Unibas

Esercizio 6 - soluzione

```python
from PIL import Image, ImageDraw, ImageFont

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))

img_draw = ImageDraw.Draw(img)

img_draw.rectangle((50, 30, 250, 100), fill='blue')

!ls '/usr/share/fonts/truetype/liberation'

font = ImageFont.truetype(font="LiberationSans-Regular.ttf",size=60)

img_draw.text((60, 40), 'Unibas', fill='white', font=font)

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

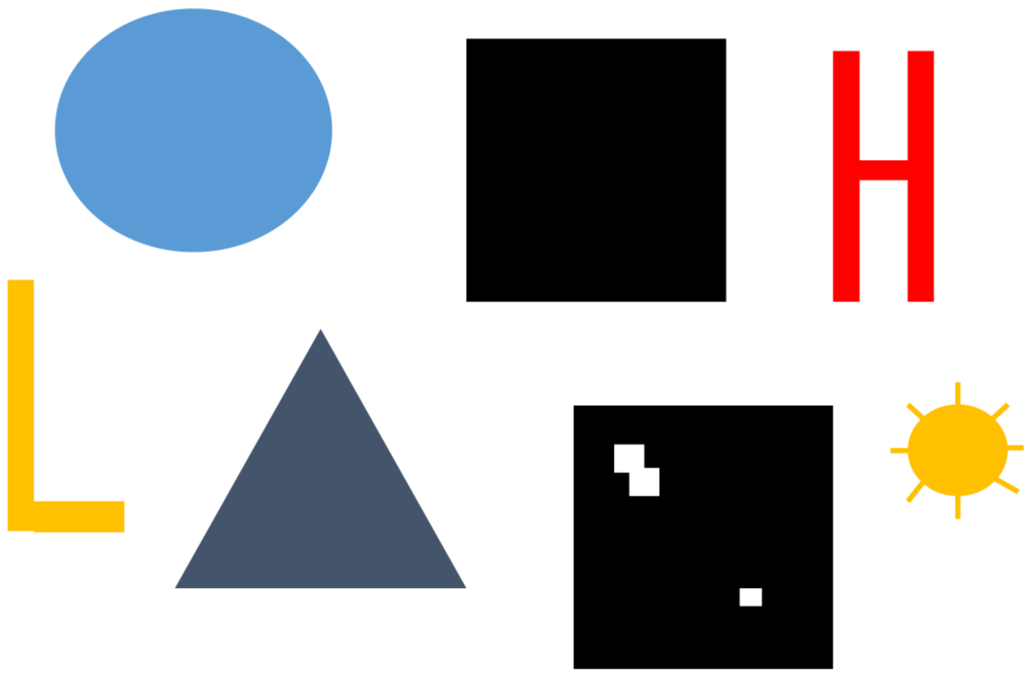
```
LiberationMono-BoldItalic.ttf        LiberationSansNarrow-Bold.ttf
LiberationMono-Bold.ttf              LiberationSansNarrow-Italic.ttf
LiberationMono-Italic.ttf            LiberationSansNarrow-Regular.ttf
LiberationMono-Regular.ttf           LiberationSans-Regular.ttf
LiberationSans-BoldItalic.ttf        LiberationSerif-BoldItalic.ttf
LiberationSans-Bold.ttf              LiberationSerif-Bold.ttf
LiberationSans-Italic.ttf            LiberationSerif-Italic.ttf
LiberationSansNarrow-BoldItalic.ttf  LiberationSerif-Regular.ttf
<matplotlib.image.AxesImage at 0x7fc5aa4a9b38>
```



Esercizio 7

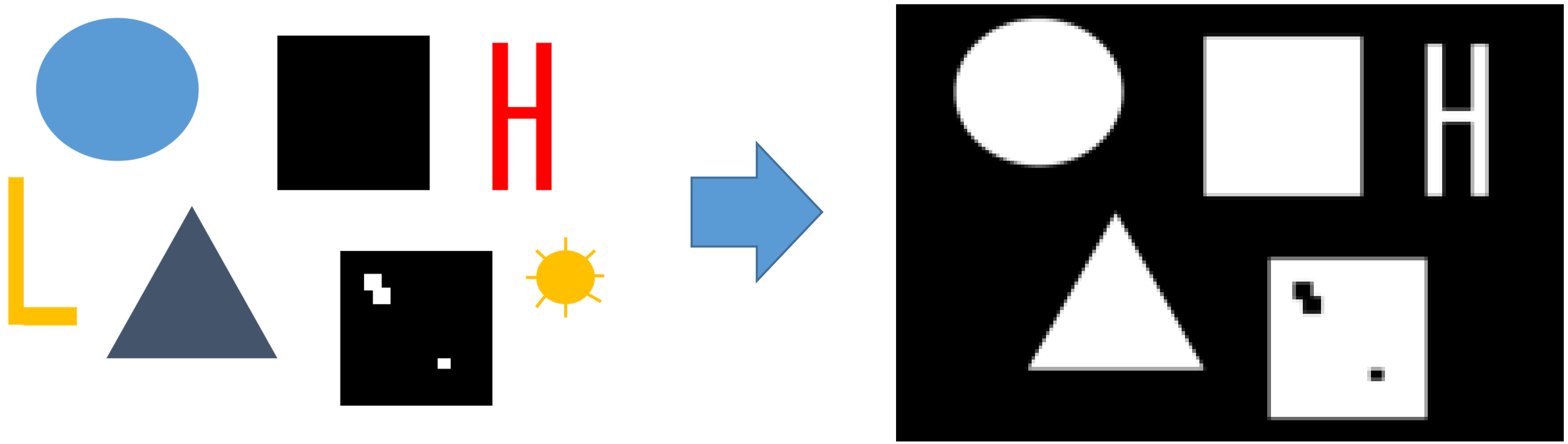
Applicare all’immagine
https://web.unibas.it/bloisi/corsi/images/forme.png
le operazioni di

- erosion
- dilation
- aperture
- closing



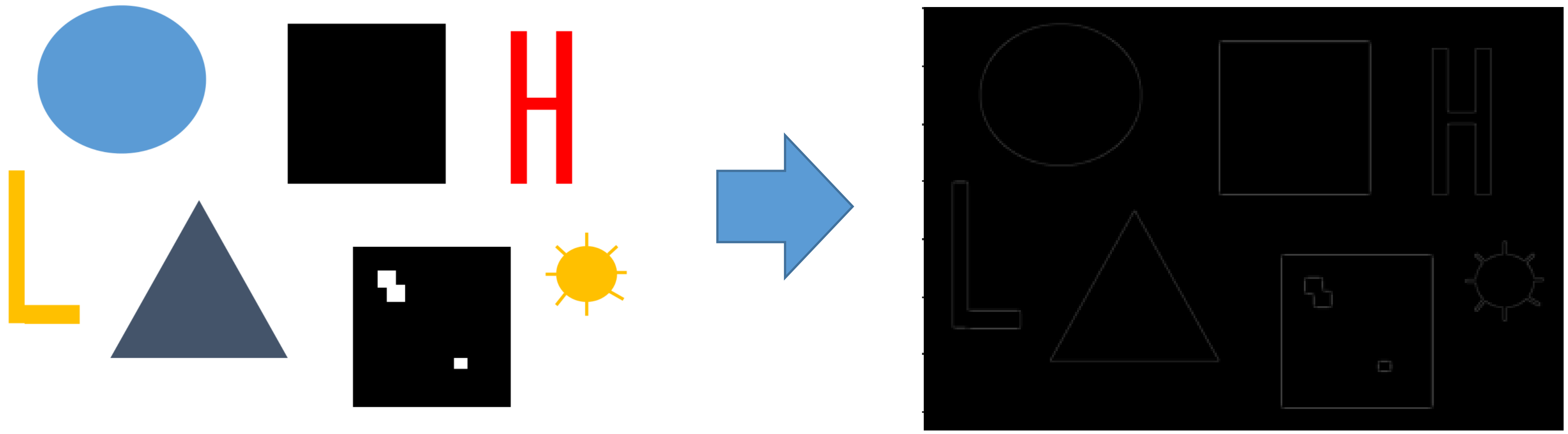
Esercizio 8

Applicare all’immagine
https://web.unibas.it/bloisi/corsi/images/forme.png
il metodo di thresholding di Otsu



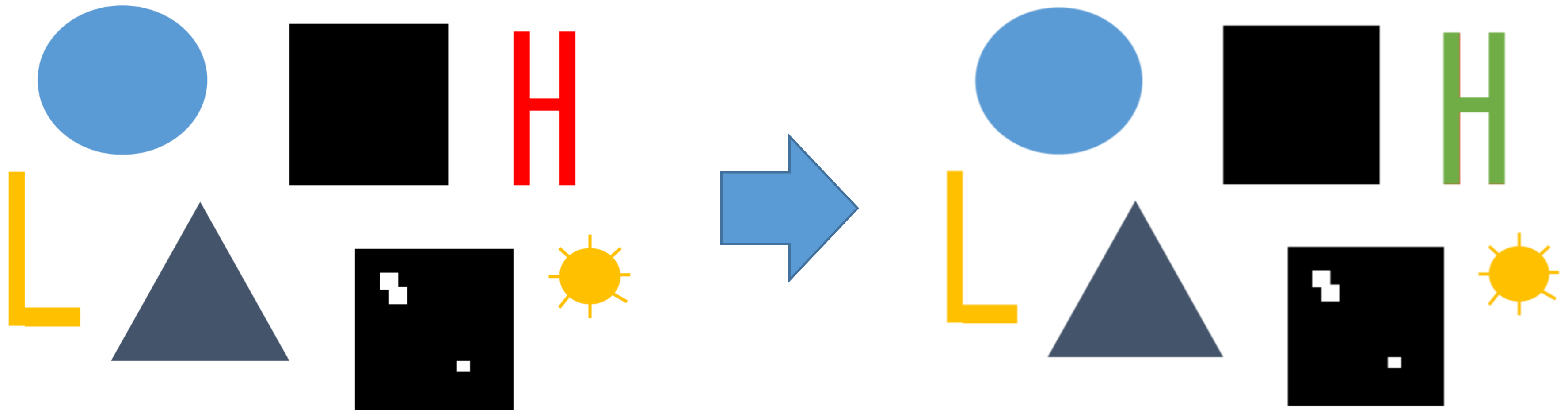
Esercizio 9

Estrarre i contorni dall’immagine
https://web.unibas.it/bloisi/corsi/images/forme.png



Esercizio 10

Ricolorare la figura in rosso nella immagine
https://web.unibas.it/bloisi/corsi/images/forme.png
con il colore verde



UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Visione e Percezione

Docente

Domenico D. Bloisi

Trasformazioni