



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

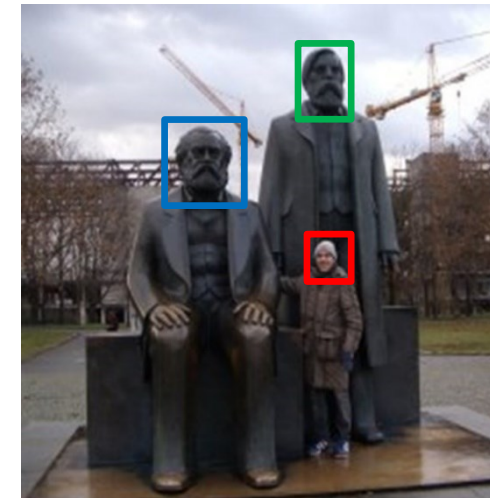
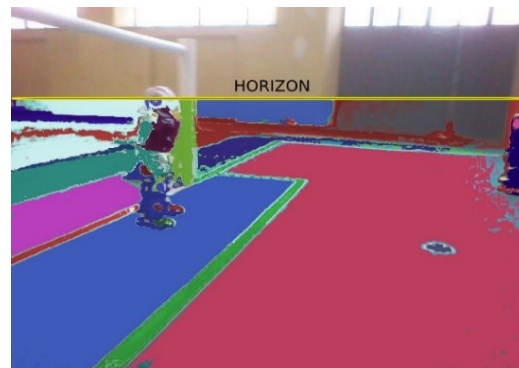
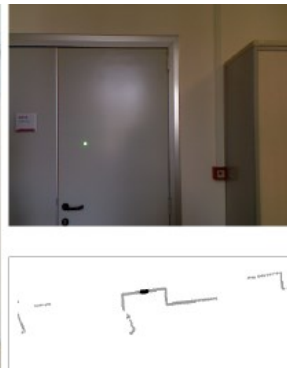
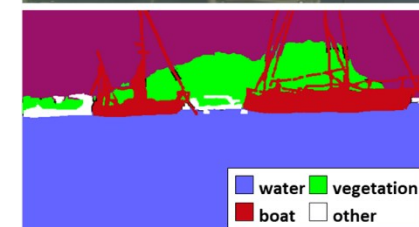
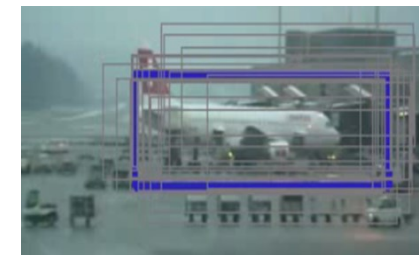
Corso di Visione e Percezione

OpenCV (Python)



Docente

Domenico D. Bloisi



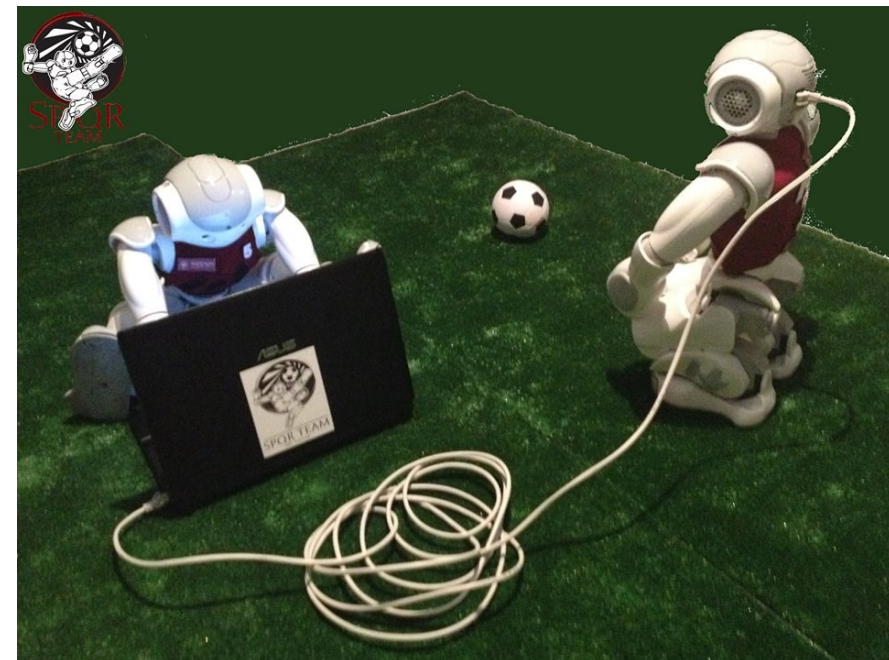
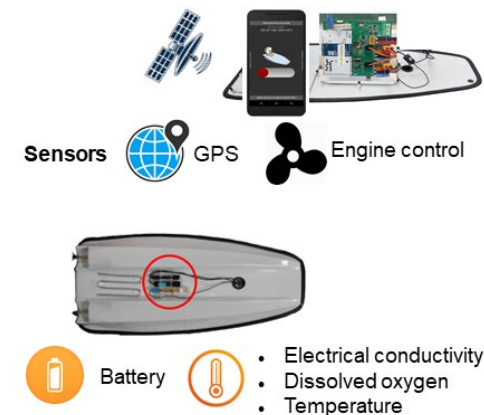
Domenico Daniele Bloisi

- Ricercatore RTD B
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>



Informazioni sul corso

- Home page del corso
<http://web.unibas.it/bloisi/corsi/visione-e-percezione.html>
- Docente: Domenico Daniele Bloisi
- Periodo: **Il semestre** marzo 2021 – giugno 2021

Martedì 17:00-19:00 (Aula COPERNICO)

Mercoledì 8:30-10:30 (Aula COPERNICO)



Codice corso Google Classroom:

[https://classroom.google.com/c/
NjI2MjA4MzgzNDFa?cjc=xgolays](https://classroom.google.com/c/NjI2MjA4MzgzNDFa?cjc=xgolays)

Ricevimento

- Su appuntamento tramite Google Meet

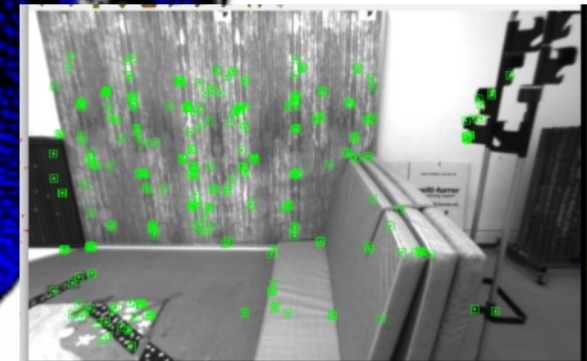
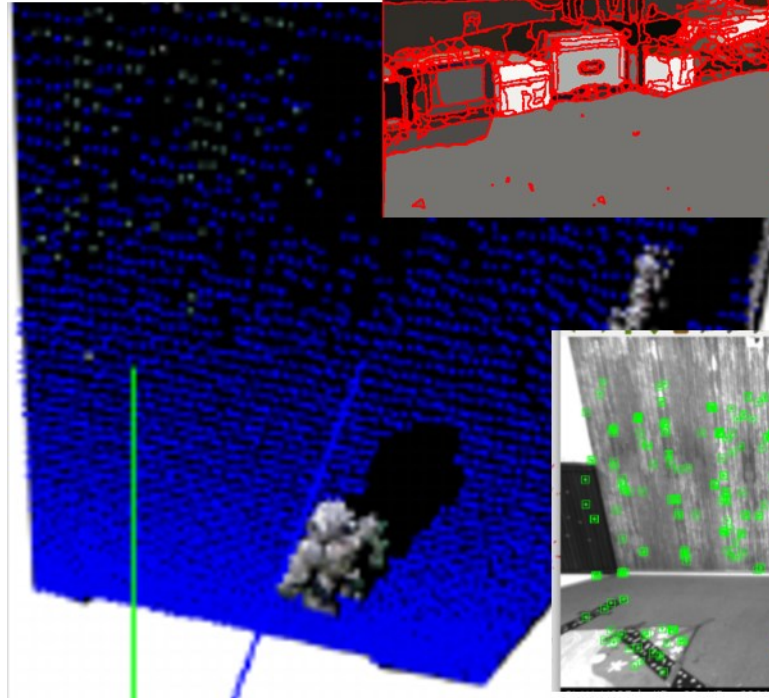
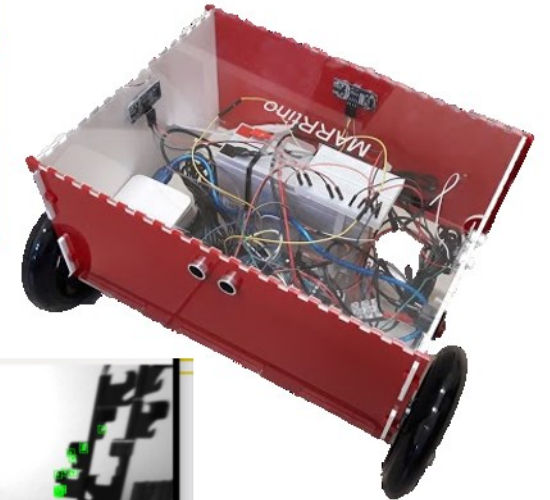
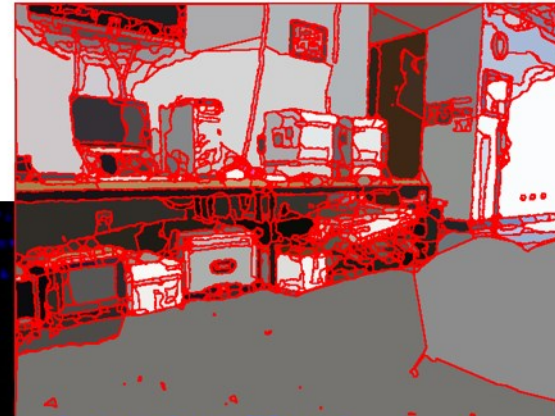
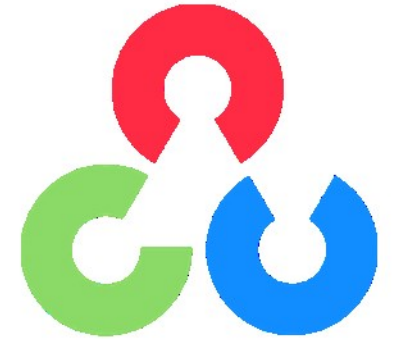
Per prenotare un appuntamento inviare una email a

domenico.bloisi@unibas.it



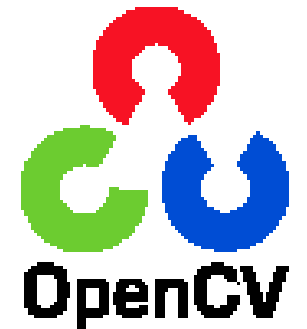
Programma – Visione e Percezione

- 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



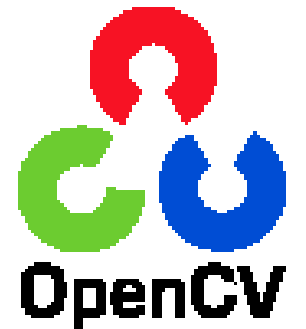
OpenCV

- **OpenCV** (Open Source Computer Vision Library) è una libreria software open source per la computer vision e il machine learning
- Distribuita con licenza **BSD** (è possibile utilizzarla per fini commerciali)
- Più di 2500 algoritmi disponibili
- Più di 47000 utenti nella community
- Più di 14 milioni di download



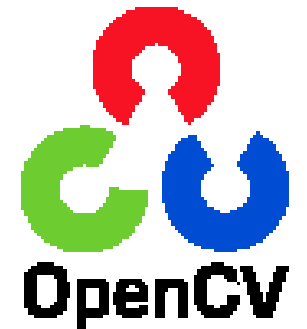
OpenCV

- Può essere utilizzata con C++, Python, Java e MATLAB
- Può essere installata su Windows, Linux, Android e Mac OS
- Dispone di interface per CUDA e OpenCL
- Viene usata da Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota



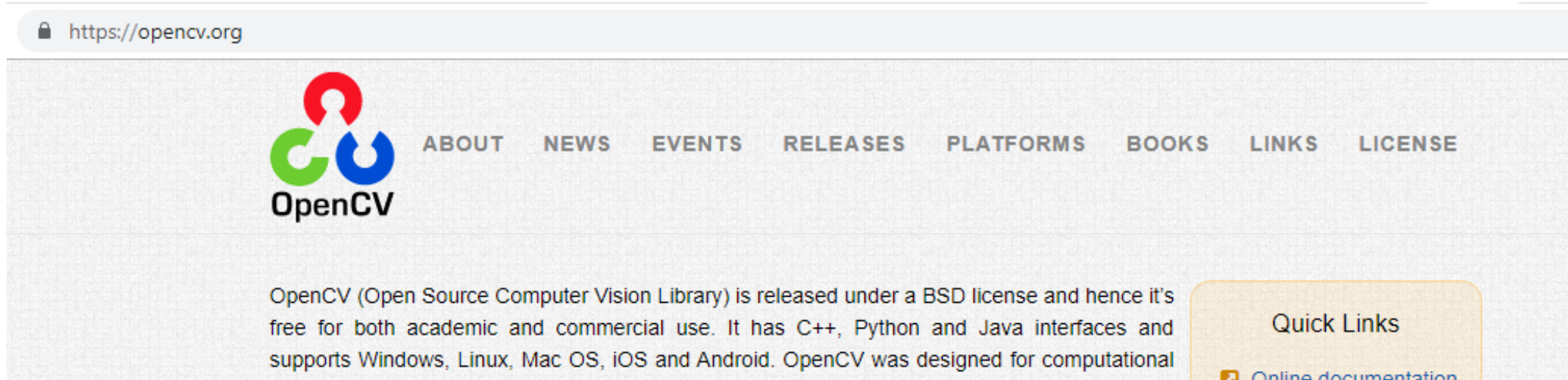
OpenCV - storia

- OpenCV was started at Intel in 1999 by **Gary Bradsky**, and the first release came out in 2000. **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team.
- In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge.
- Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project.



OpenCV - links

- Home: <https://opencv.org/>
- Documentatation: <https://docs.opencv.org/>
- Q&A forum: <http://answers.opencv.org>
- GitHub: <https://github.com/opencv/>

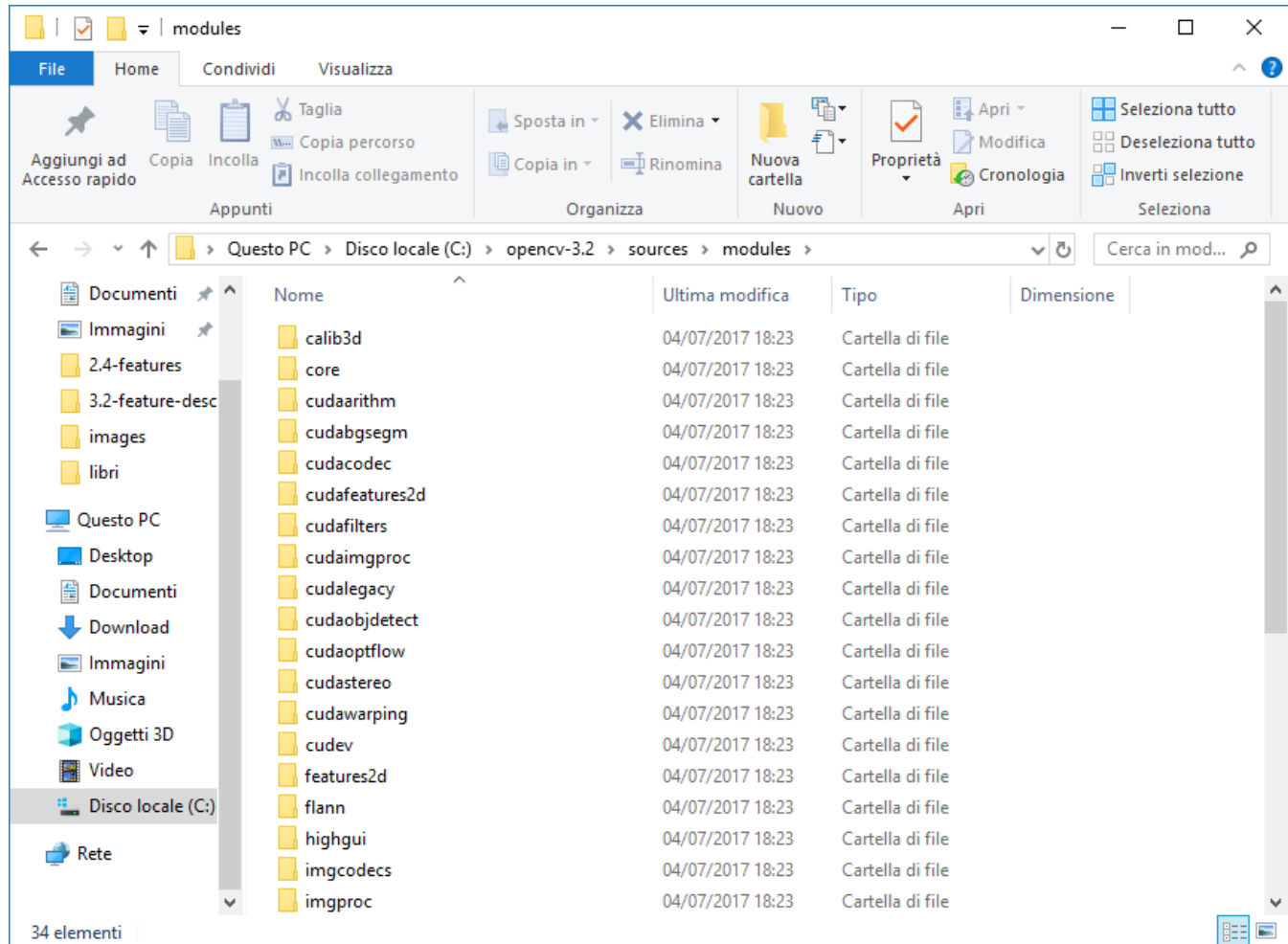


OpenCV - moduli

OpenCV ha una struttura **modulare**

I principali moduli sono:

- core
- imgproc
- video
- calib3d
- features2d
- objdetect
- highgui



OpenCV – core e imgproc

Core functionality (core)

A compact module defining basic data structures, including the dense multi-dimensional array **Mat** and basic functions used by all other modules.

Image Processing (imgproc)

An image processing module that includes linear and non-linear **image filtering**, geometrical **image transformations** (resize, affine and perspective warping, generic table-based remapping), **color space conversion**, **histograms**, and so on.

OpenCV – video e calib3d

Video Analysis (video)

A video analysis module that includes [motion estimation](#), [background subtraction](#), and [object tracking](#) algorithms.

Camera Calibration and 3D Reconstruction (calib3d)

Basic multiple-view geometry algorithms, single and stereo [camera calibration](#), object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

OpenCV – features2d e objdetect

2D Features Framework (features2d)

Salient [feature detectors](#), [descriptors](#), and descriptor matchers.

Object Detection (objdetect)

[Detection](#) of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

OpenCV – highgui e videoio

High-level GUI (highgui)

an easy-to-use interface to [simple UI](#) capabilities.

Video I/O (videoio)

An easy-to-use interface to [video capturing and video codecs](#).

OpenCV – Python

- Python is slower compared to C++ or C. Python is built for its simplicity, portability and moreover, creativity where users need to worry only about their algorithm, not programming troubles.
- Python-OpenCV is just a **wrapper** around the original C/C++ code. It is normally used for combining best features of both the languages.
Performance of C/C++ & Simplicity of Python.
- So when you call a function in OpenCV from Python, what actually runs is underlying C/C++ source.
- Performance penalty is < 4%


OpenCV Timeline

Version	Released	Reason	Lifetime
pre 1.0	2000 (first alpha)	-	6 years
1.0	2006 (ChangeLog)	maturity	3 years
2.0	2009 (ChangeLog)	C++ API	>3 years
3.0	2014	several (next level maturity, ...)	
4.0	Nov. 2018	better DNN support	

Installare OpenCV-Python

← → ↻ docs.opencv.org/master/da/df6/tutorial_py_table_of_contents_setup.html

ENHANCED BY Google

 **OpenCV** 4.5.2-dev ▾
Open Source Computer Vision

Main Page | Related Pages | Modules | Namespaces ▾ | Classes ▾ | Files ▾ | Examples | Java documentation

OpenCV-Python Tutorials >

Introduction to OpenCV

- [Introduction to OpenCV-Python Tutorials](#)
 - Getting Started with OpenCV-Python
- [Install OpenCV-Python in Windows](#)
 - Set Up OpenCV-Python in Windows
- [Install OpenCV-Python in Fedora](#)
 - Set Up OpenCV-Python in Fedora
- [Install OpenCV-Python in Ubuntu](#)
 - Set Up OpenCV-Python in Ubuntu

```
pip install opencv-python
```

OpenCV in Colab

Per prima cosa andremo a vedere qual è la versione di OpenCV disponibile in Google Colab



```
import cv2 as cv  
  
print(cv.__version__)
```



4.1.2

Import cv2


Perche usiamo `import cv2` se la versione di OpenCV è la 4.x.x?

`cv2` non rappresenta la versione di OpenCV, ma la versione delle `API C++` sopra le quali poggiano le nuove versioni di OpenCV, in contrasto con la vecchia interfaccia C (denominata `cv`)

OpenCV 4.1.2 docs

← → ↻ docs.opencv.org/4.1.2/

Google Custom Search

 **OpenCV** 4.1.2
Open Source Computer Vision

Main Page Related Pages Modules Namespaces ▾ Classes ▾ Files ▾ Examples

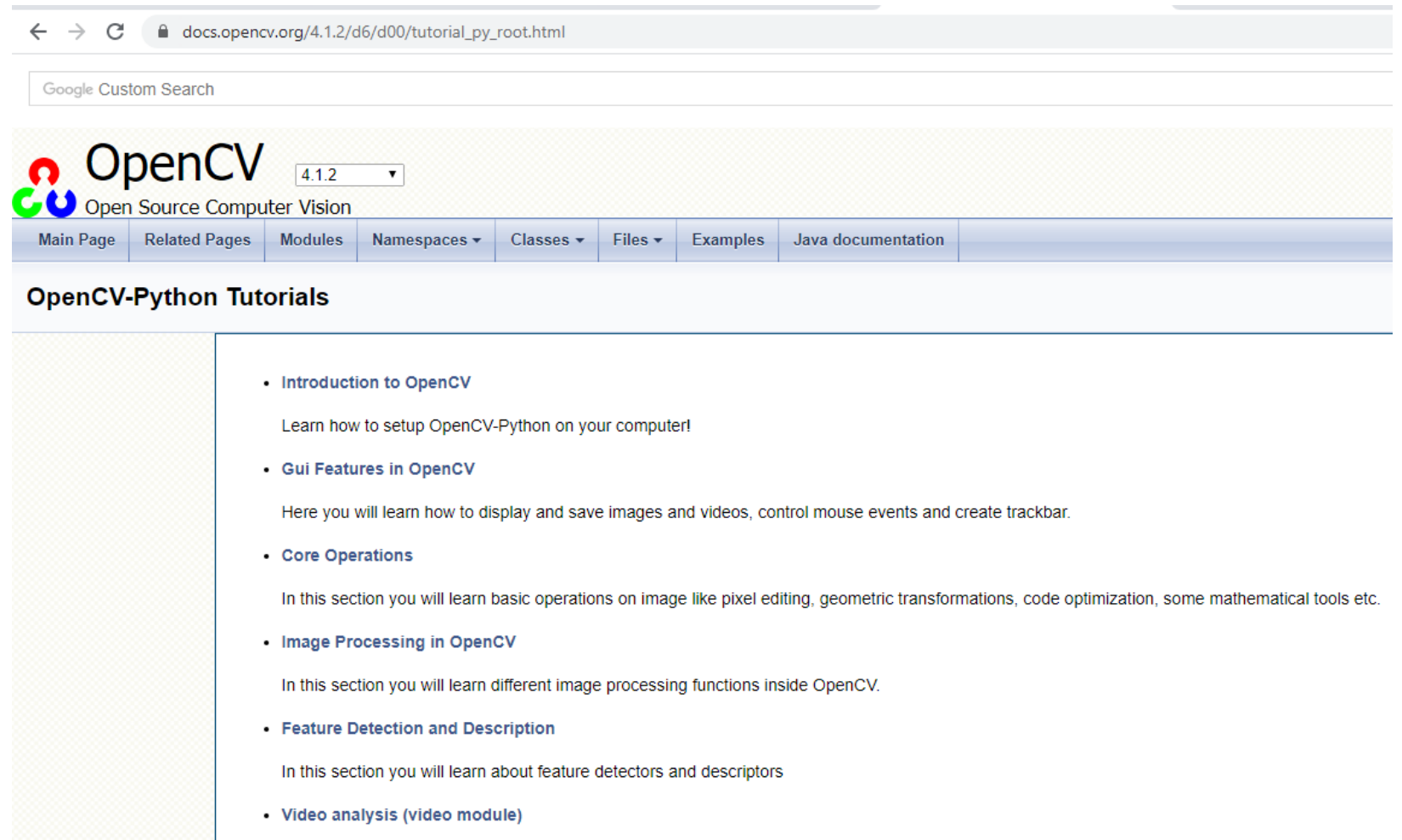
OpenCV modules

- [Introduction](#)
- [OpenCV Tutorials](#)
- [OpenCV-Python Tutorials](#)
- [OpenCV.js Tutorials](#)
- [Tutorials for contrib modules](#)
- [Frequently Asked Questions](#)
- [Bibliography](#)
- Main modules:
 - core. **Core functionality**
 - imgproc. **Image Processing**
 - imgcodecs. **Image file reading and writing**
 - videoio. **Video I/O**
 - highgui. **High-level GUI**
 - video. **Video Analysis**
 - calib3d. **Camera Calibration and 3D Reconstruction**
 - features2d. **2D Features Framework**
 - objdetect. **Object Detection**
 - dnn. **Deep Neural Network module**

<https://docs.opencv.org/4.1.2/>

OpenCV-Python Tutorials

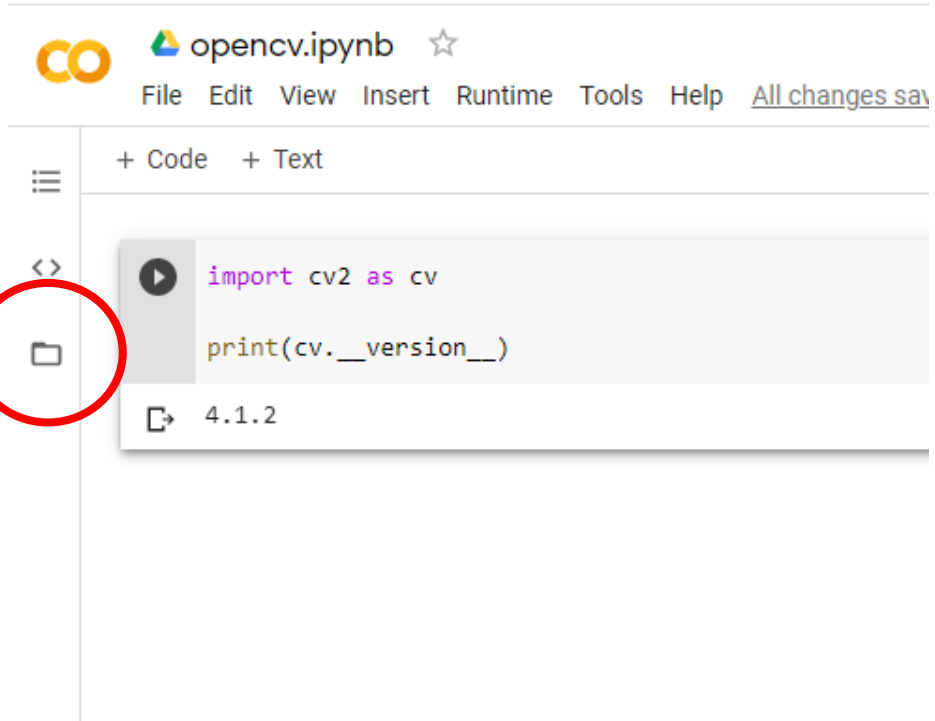
OpenCV fornisce una serie di tutorial specifici per Python che possono essere utilizzati per imparare ad utilizzare la libreria attraverso esempi pratici



The screenshot shows a web browser window displaying the OpenCV Python Tutorials page. The address bar shows the URL: docs.opencv.org/4.1.2/d6/d00/tutorial_py_root.html. Below the address bar is a search bar labeled "Google Custom Search". The main content area features the OpenCV logo and the text "Open Source Computer Vision" with a version dropdown menu set to "4.1.2". A navigation menu includes links for "Main Page", "Related Pages", "Modules", "Namespaces", "Classes", "Files", "Examples", and "Java documentation". The page title is "OpenCV-Python Tutorials". The main content area lists several tutorial topics:

- **Introduction to OpenCV**
Learn how to setup OpenCV-Python on your computer!
- **Gui Features in OpenCV**
Here you will learn how to display and save images and videos, control mouse events and create trackbar.
- **Core Operations**
In this section you will learn basic operations on image like pixel editing, geometric transformations, code optimization, some mathematical tools etc.
- **Image Processing in OpenCV**
In this section you will learn different image processing functions inside OpenCV.
- **Feature Detection and Description**
In this section you will learn about feature detectors and descriptors
- **Video analysis (video module)**

Load an image in Colab



opencv.ipynb ☆

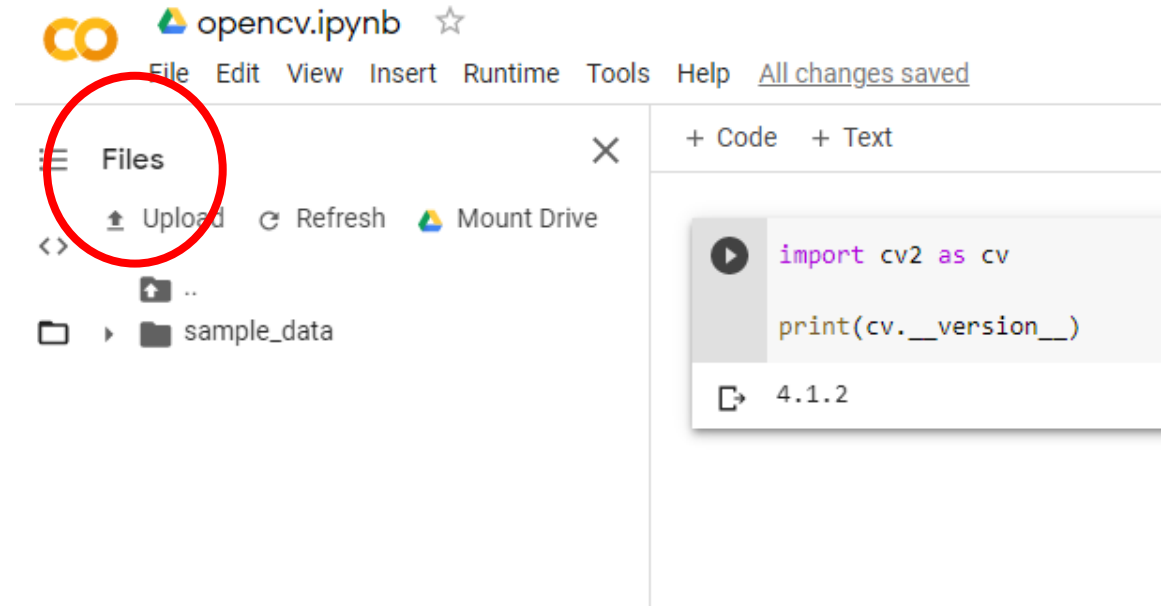
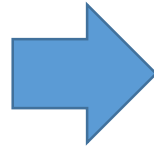
File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

```
import cv2 as cv
print(cv.__version__)
```

4.1.2

A red circle highlights the file icon in the left sidebar.



opencv.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

```
import cv2 as cv
print(cv.__version__)
```

4.1.2

Files

Upload Refresh Mount Drive

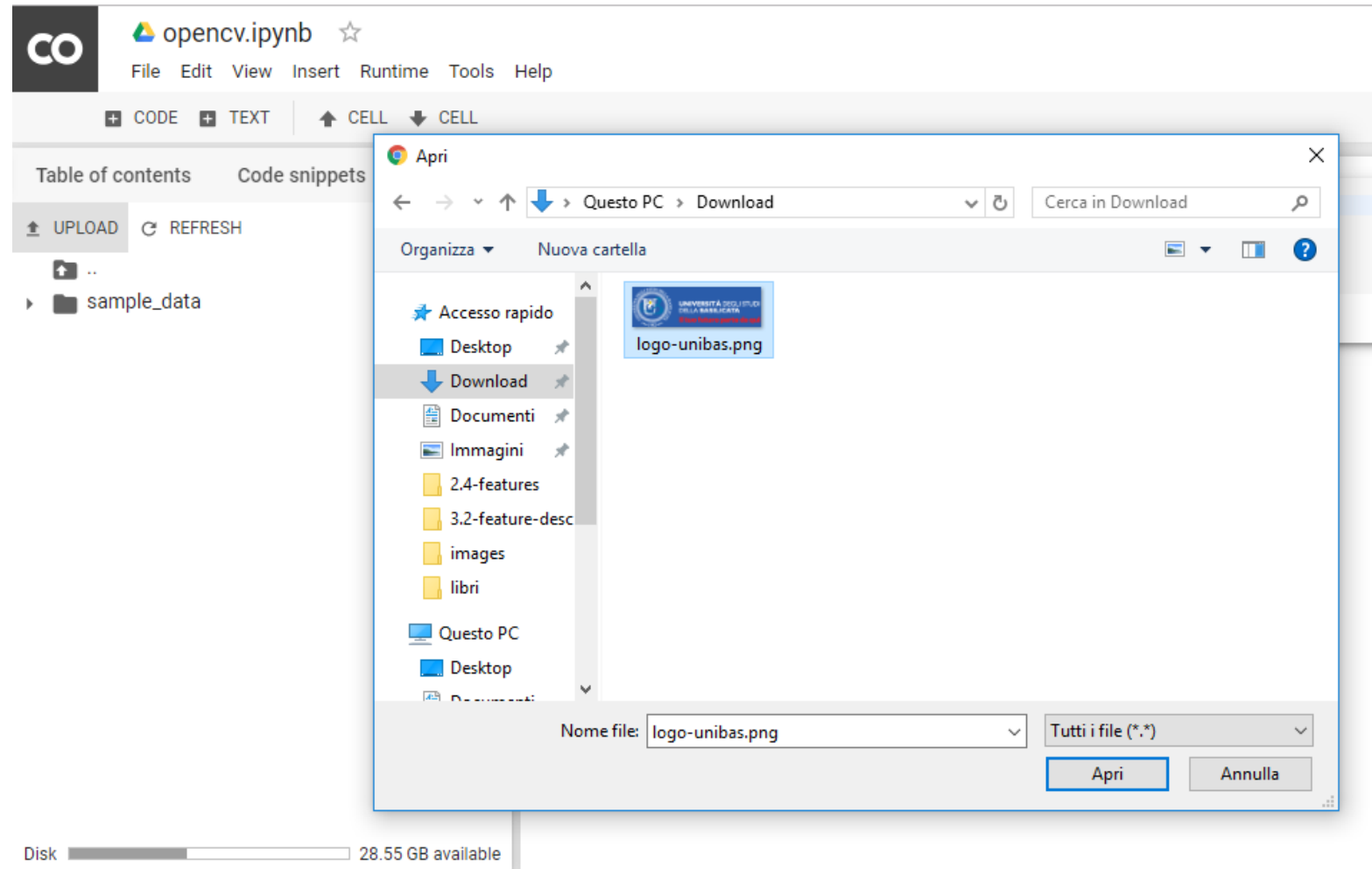
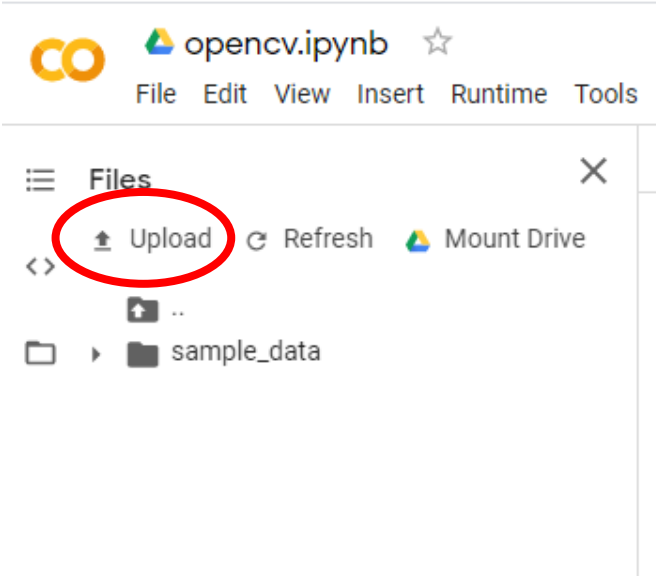
..

sample_data

A red circle highlights the 'Files' panel header.

Load an image in Colab

<http://portale.unibas.it/contents/instance1/images/logo-unibas.png>

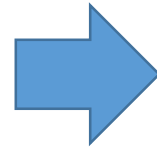


Load an image in Colab

Reminder, uploaded files will get deleted when this runtime is recycled.

[More info](#)

OK



The screenshot shows the Google Colab interface for a notebook named 'opencv.ipynb'. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with a status indicator 'All changes saved'. The left sidebar displays a file explorer with options for 'Upload', 'Refresh', and 'Mount Drive'. A folder named 'sample_data' is expanded, and a file named 'logo-unibas.png' is highlighted with a red circle. The main code editor area shows a code cell with the following Python code:

```
import cv2 as cv  
print(cv.__version__)
```

The output of the code cell is '4.1.2'. Below the code editor, a preview of the 'logo-unibas.png' file is shown. At the bottom of the interface, a disk usage indicator shows '79.82 GB available'. In the bottom right corner, there is a banner for the University of Basilicata with the text 'UNIVERSITÀ DEGLI STUDI DELLA BASILICATA' and 'Il tuo futuro parte da qui'.

Read an image with OpenCV

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png')
plt.imshow(img)
plt.xticks([], plt.yticks([])) # to hide tick values on X and Y axis
plt.show()
```



Source image

warning

Color image loaded by OpenCV is in **BGR** mode. But Matplotlib displays in RGB mode. So color images will not be displayed correctly in Matplotlib if image is read with OpenCV.



Why does OpenCV use BGR color format?

The reason the early developers at OpenCV chose BGR color format is that back then BGR color format was popular among camera manufacturers and software providers.

E.g. in Windows, when specifying color value using COLORREF they use the BGR format **0x00bbggrr**.

BGR was a choice made for historical reasons and now we have to live with it.

Images as NumPy arrays



Images in
OpenCV-Python
are NumPy arrays

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png')

print(type(img))
print(img.ndim)
print(img.shape)

plt.imshow(img)
plt.xticks([], plt.yticks([])) # to hide tick values on X and Y axis
plt.show()
```

```
<class 'numpy.ndarray'>
3
(97, 312, 3)
```



RGB visualization in Matplotlib

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR color space

print(type(img))
print(img.ndim)
print(img.shape)

img_rgb = img[:, :, ::-1]

plt.imshow(img_rgb)
plt.xticks([], plt.yticks([])) # to hide tick values on X and Y axis
plt.show()
```



```
<class 'numpy.ndarray'>
3
(97, 312, 3)
```

```
a = [1, 2, 3]
b = a[0:3:1] # start=0, stop=3, step=1
print(b)
c = a[::1]
print(c)
d = a[::-1]
print(d)
```

```
[1, 2, 3]
[1, 2, 3]
[3, 2, 1]
```



UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA

Il tuo futuro parte da qui

Accessing and modifying pixel values



```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR color space

# accessing pixel in position (50,100)
px = img[50,100] #[y-value, x-value]
print(px)

# accessing only blue pixel
blue = img[50,100,0]
print(blue)

img[50,100] = [255,255,255]
print(img[50,100])
```

```
↳ [170  92  42]
   170
   [255 255 255]
```

warning

Numpy is a optimized library for fast array calculations. So simply accessing each and every pixel values and modifying it will be very slow and it is discouraged.

item e itemset



```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR color space

# accessing only blue pixel
blue = img.item(50,100,0)
print(blue)

img.itemset((50,100,0),255)
print(img[50,100])
```



```
170
[255  92  42]
```

I metodi Numpy `array.item()` e `array.itemset()` sono considerati migliori per accedere agli elementi di una immagine.

Tuttavia, se si vuole accedere a tutti e tre i canali B,G,R è necessario ripetere la chiamata tre volte separatamente.

Accessing image properties



number of rows, columns, and channels (if image is color)

```
[28] print(img.shape)
```

```
↳ (97, 312, 3)
```

Total number of pixels

```
[29] print(img.size)
```

```
↳ 90792
```

Image datatype

```
▶ print(img.dtype)
```

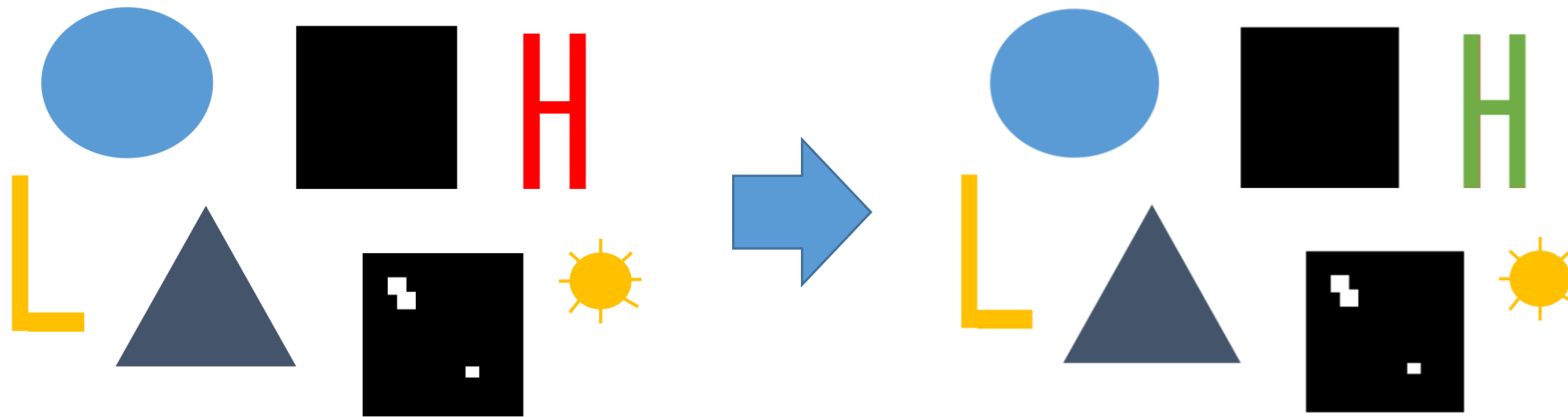
```
↳ uint8
```

Esercizio 10 – Lezione 2.4

Ricolorare la figura in rosso nella immagine

<https://web.unibas.it/bloisi/corsi/images/forme.png>

con il colore verde



Soluzione con NumPy

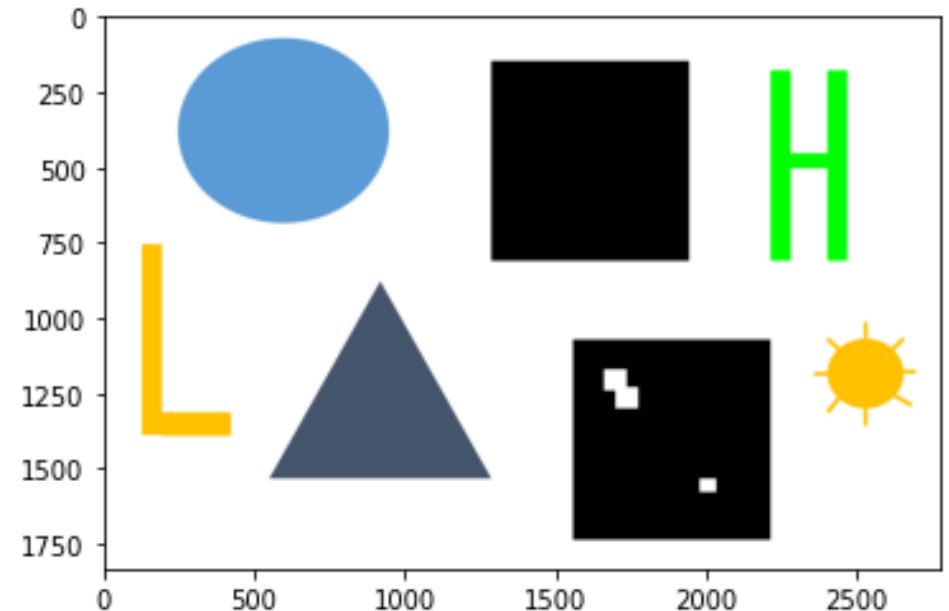
```
▶ from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt
import numpy as np

url = "http://web.unibas.it/bloisi/corsi/images/forme.png"
pil_img = Image.open(urlopen(url)).convert('RGB')
img = np.array(pil_img)

h = img.shape[0]
w = img.shape[1]

for y in range(0, h):
    for x in range(0, w):
        if img.item(y,x,0) > 200 and \
           img.item(y,x,1) < 20 and \
           img.item(y,x,2) < 20 :
            img.itemset((y,x,0),0) #red
            img.itemset((y,x,1),255) #green
            img.itemset((y,x,2),0) #blue

_ = plt.imshow(img)
```



Soluzione con OpenCV

```
from PIL import Image
from urllib.request import urlopen
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv

url = "http://web.unibas.it/bloisi/corsi/images/forme.png"
pil_img = Image.open(urlopen(url)).convert('RGB')
img = np.array(pil_img)
print(img.shape)

img2 = cv.inRange(img, (200, 0, 0), (255, 20, 20))
print(img2.shape)

_ = plt.imshow(img2, cmap="gray")
```

(1832, 2772, 3)
(1832, 2772)

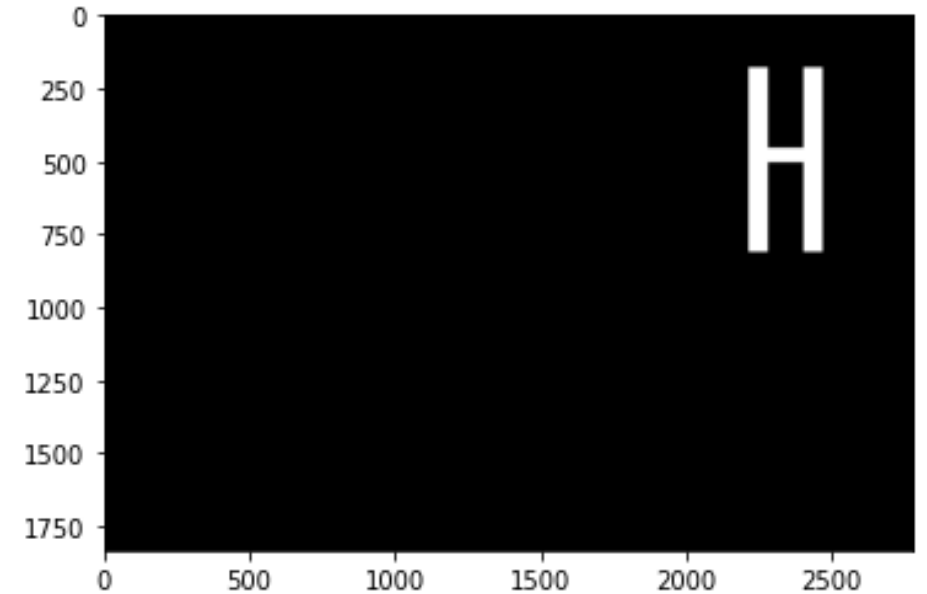


Image ROI

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR color space

logo = img[0:98,0:98]
img[0:98, 100:198] = logo
img[0:98, 200:298] = logo

img_rgb = img[:, :, ::-1]

plt.imshow(img_rgb)
plt.xticks([]), plt.yticks([]) # to hide tick values on X and Y axis
plt.show()
```



Changing color-space

Ci sono tantissimi metodi per cambiare il color-space disponibili in OpenCV.

```
import cv2 as cv
flags = [i for i in dir(cv) if i.startswith('COLOR_')]
print(flags)
print(len(flags))
```

```
['COLOR_BAYER_BG2BGR', 'COLOR_BAYER_BG2BGRA', 'COLOR_BAYER_BG2BGR_EA', 'COLOR_BAYER_BG2BGR_VNG', 'COLOR_BAYER_BG2GRAY']
274
```

BGR2GRAY



```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR color space

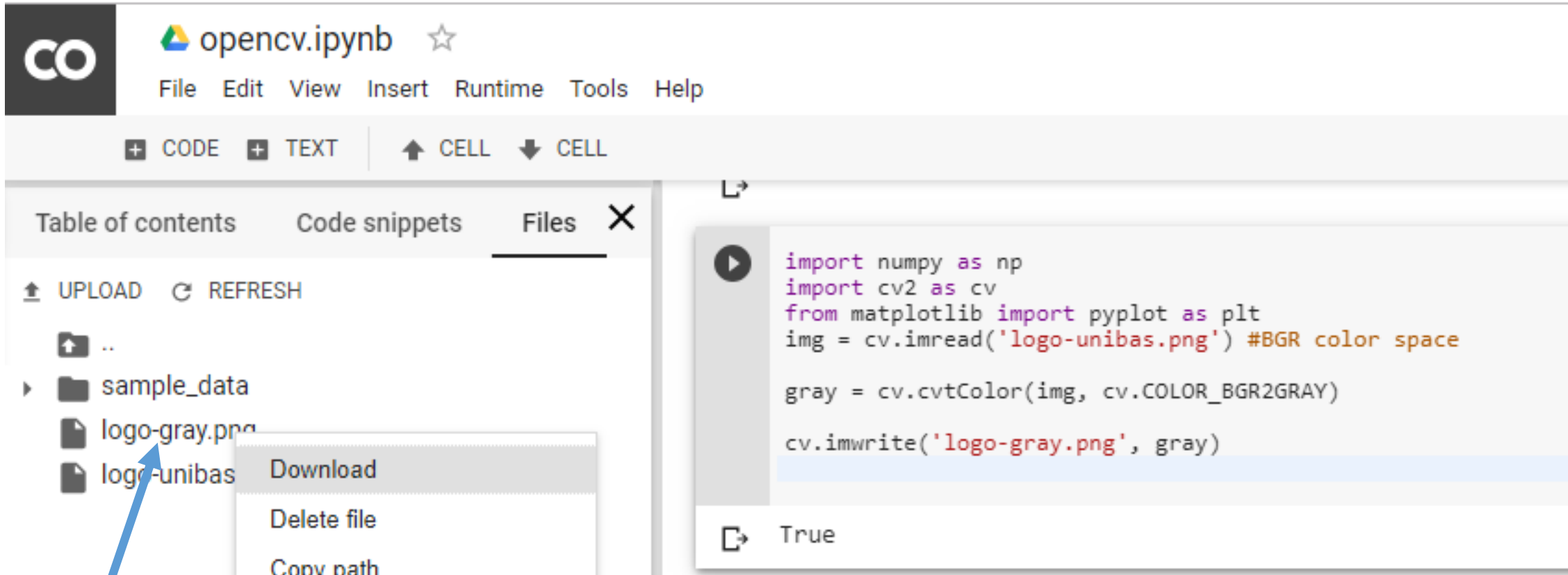
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

cv.imwrite('logo-gray.png', gray)
```



True

Grayscale conversion



The screenshot shows the OpenCV Jupyter Notebook interface. The top bar includes the 'CO' logo, the title 'opencv.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The main area is divided into a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with 'sample_data' and two files: 'logo-gray.png' and 'logo-unibas.png'. A right-click context menu is open over 'logo-unibas.png', showing options: 'Download', 'Delete file', and 'Copy path'. A blue arrow points from the text 'tasto destro del mouse' to the 'logo-unibas.png' file. The code editor contains the following Python code:

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR color space

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

cv.imwrite('logo-gray.png', gray)
```

Below the code editor, the output 'True' is displayed.

tasto
destro del
mouse



Read an image from URL with OpenCV

```
▶ import numpy as np
import cv2 as cv

import matplotlib.pyplot as plt
import urllib.request

url = "http://portale.unibas.it/contents/instance1/images/logo-unibas.png"
url_response = urllib.request.urlopen(url)

numpy_img = np.array(bytearray(url_response.read()), dtype=np.uint8)
img = cv.imdecode(numpy_img, -1)

plt.imshow(img)
plt.xticks([], plt.yticks([])) # to hide tick values on X and Y axis
plt.show()
```



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Il tuo futuro parte da qui

BGR2RGB

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import urllib.request

url = "http://portale.unibas.it/contents/instance1/images/logo-unibas.png"

url_response = urllib.request.urlopen(url)
numpy_img = np.array(bytearray(url_response.read()), dtype=np.uint8)
img = cv.imdecode(numpy_img, -1)

rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)

plt.axis('off')
plt.imshow(rgb)
plt.show()
```



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

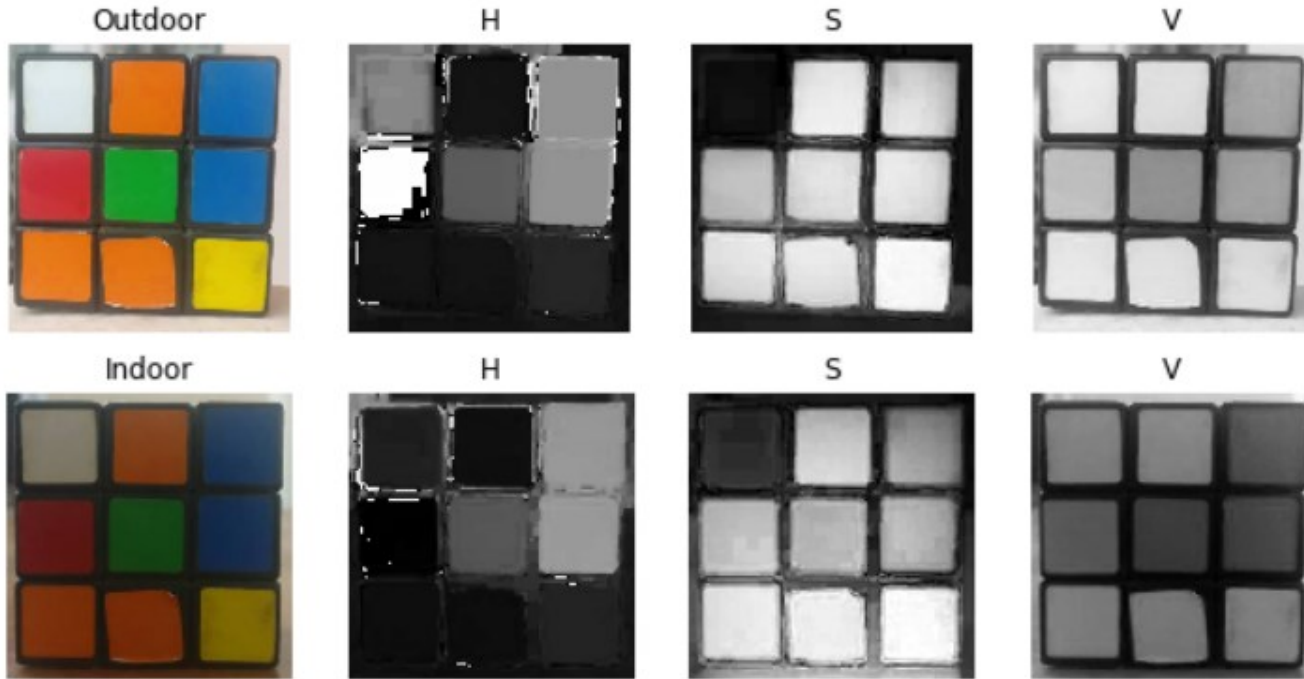
Il tuo futuro parte da qui

HSV color space

The HSV color space has the following three components:

1. H – Hue (Dominant Wavelength)
2. S – Saturation (Purity/shades of the color)
3. V – Value (Intensity)

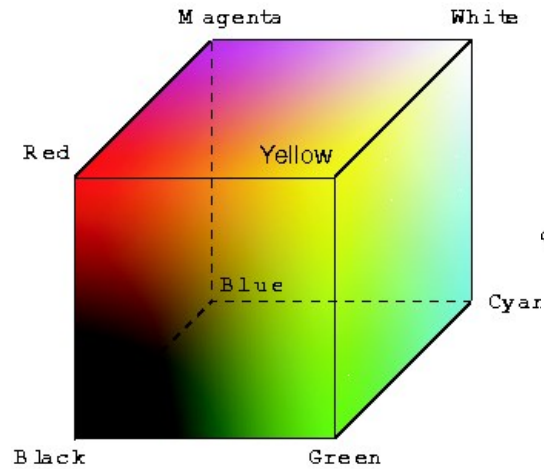
HSV color space



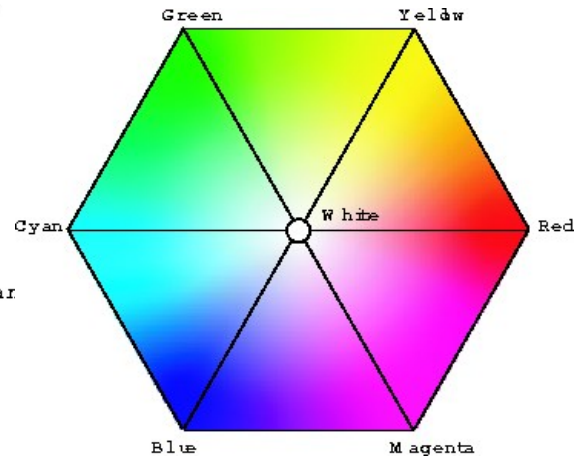
- The H component is very similar in both the images which indicates the color information is intact even under illumination changes
- The S component is also very similar in both images
- The V component captures the amount of light falling on it thus it changes due to illumination changes

HSV color-space

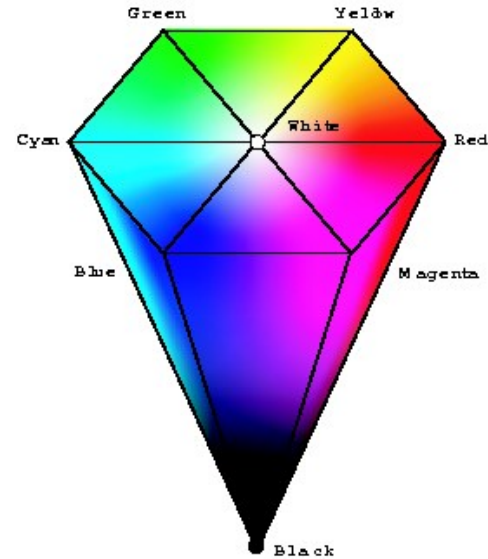
HSV is a projection of the RGB space



RGB cube

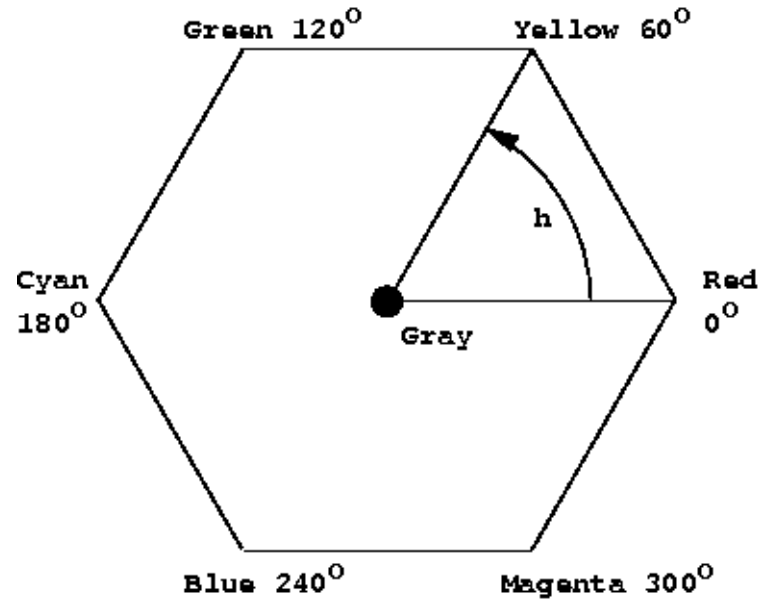
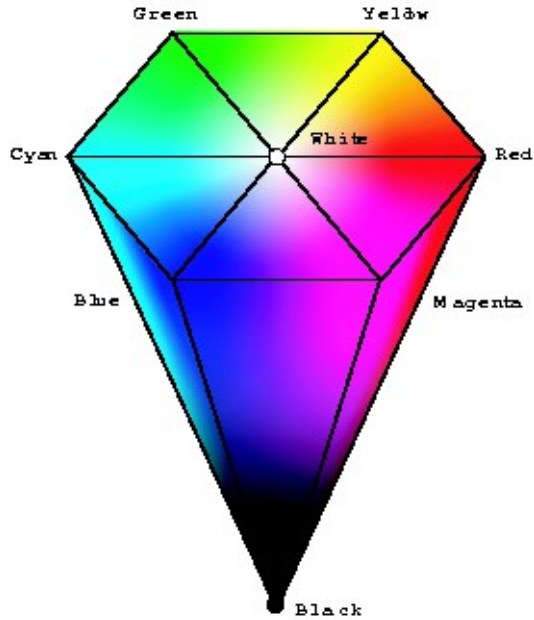


HSV top view



HSV cone

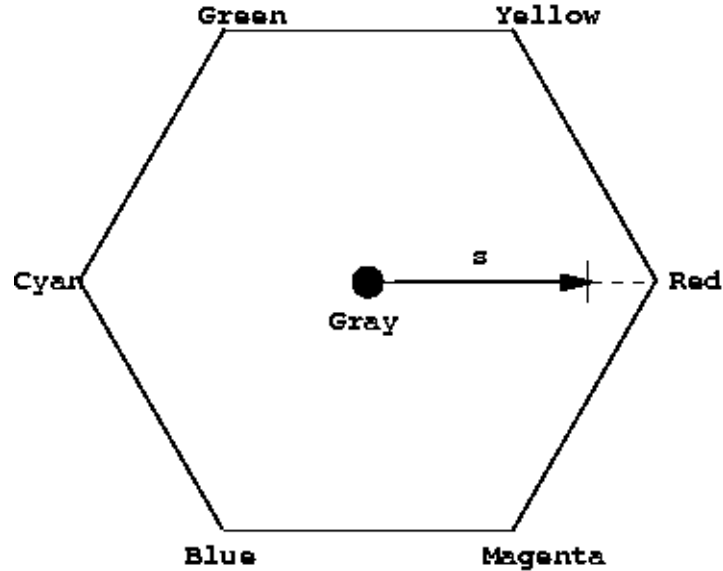
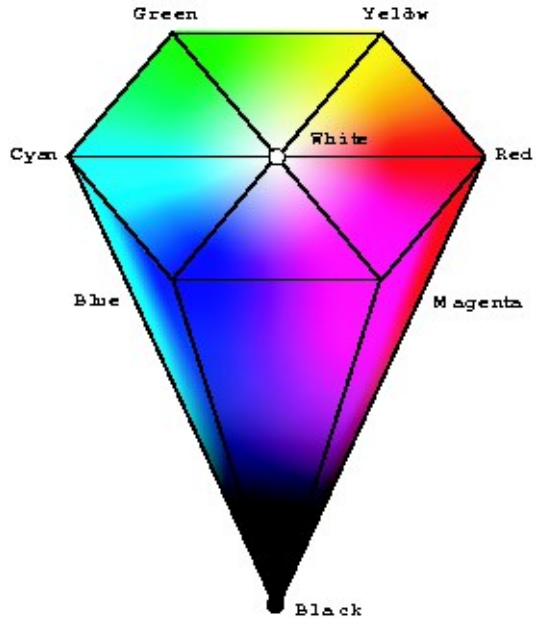
Hue



Hue, an angular measure (0 ... 360)

Hue range is [0,179] in OpenCV

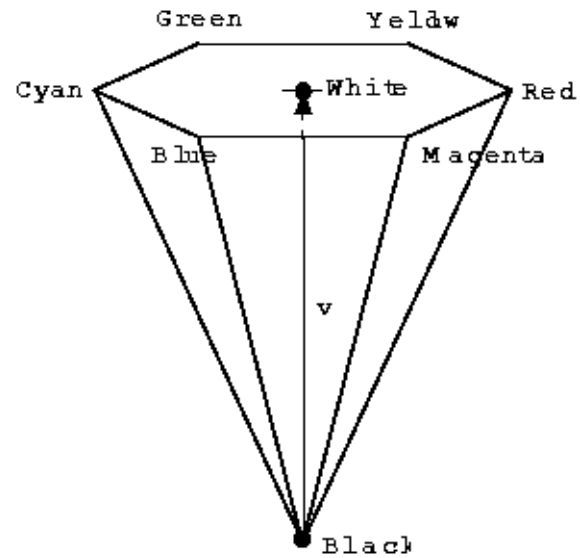
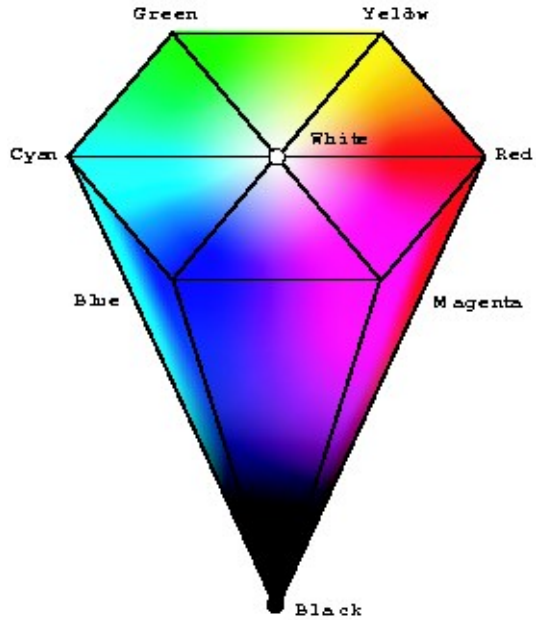
Saturation



Saturation, a fractional measure (0.0 ... 1.0)

Saturation range is [0,255] in OpenCV

Value



Value, a fractional measure (0.0 ... 1.0)

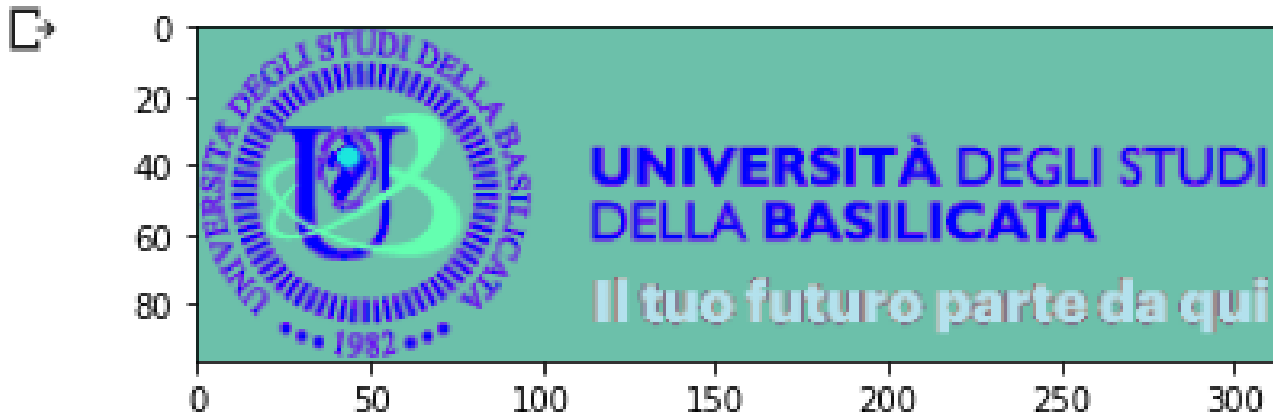
Value range is [0,255] in OpenCV

HSV conversion

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR colorspace

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

_ = plt.imshow(hsv)
```



H channel

```
▶ import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR colorspace

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

h,s,v = cv.split(hsv)

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



S channel

```
▶ import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR colorspace

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

h,s,v = cv.split(hsv)

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



V channel

```
▶ import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR colorspace

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

h,s,v = cv.split(hsv)

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



Merge

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('logo-unibas.png') #BGR color space

hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

h,s,v = cv.split(hsv)

hsv_merged = cv.merge((h,s,v))

plt.imshow(hsv_merged)
plt.xticks([], plt.yticks([])) # to hide tick values on X and Y axis
plt.show()
```



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Il tuo futuro parte da qui

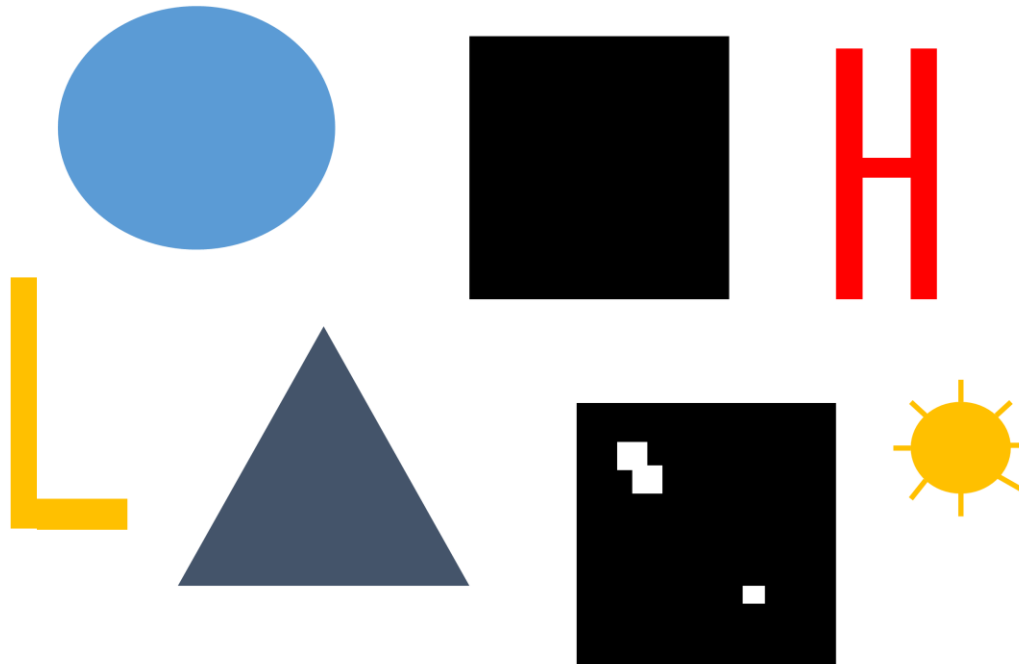
Esercizio 7 – Lezione 2.4

Applicare all'immagine

<https://web.unibas.it/bloisi/corsi/images/forme.png>

le operazioni di

- erosion
- dilation
- aperture
- closing



Soluzione

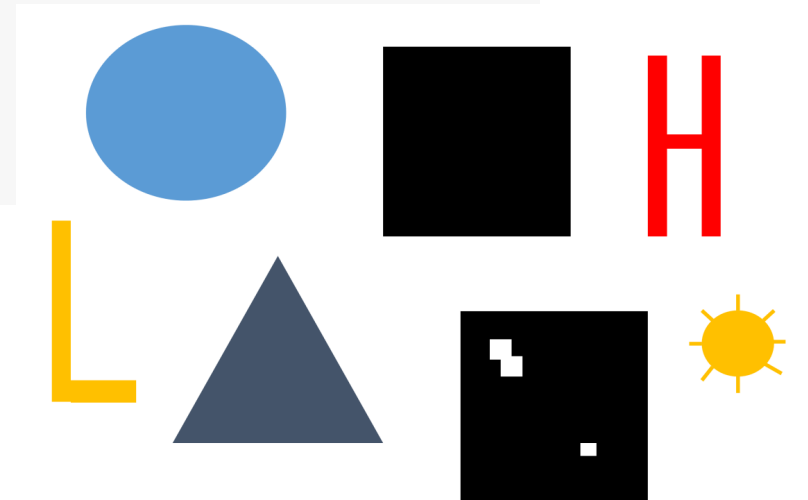
1. Carichiamo l'immagine in memoria
2. Trasformiamola in grayscale
3. Applichiamo la trasformazione
4. Mostriamo l'immagine trasformata

Carichiamo l'immagine in memoria

```
import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req = urlopen('http://web.unibas.it/bloisi/corsi/images/forme.png')
arr = np.array(bytearray(req.read()), dtype=np.uint8)
img = cv.imdecode(arr, -1) # 'Load it as it is'

cv2_imshow(img)
```

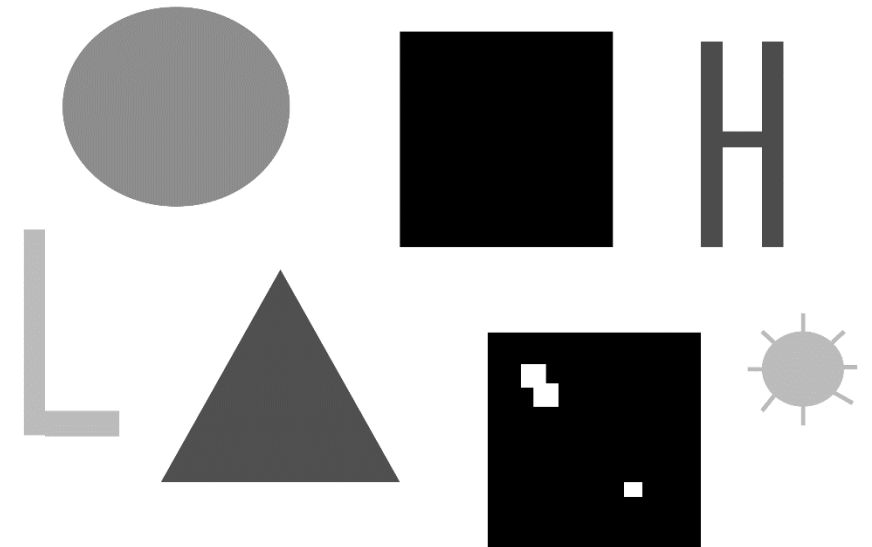


Trasformiamola in grayscale

```
▶ import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req = urlopen('http://web.unibas.it/bloisi/corsi/images/forme.png')
arr = np.array(bytearray(req.read()), dtype=np.uint8)
gray_img = cv.imdecode(arr, cv.IMREAD_GRAYSCALE) # Load as grayscale image

cv2_imshow(gray_img)
```



Erosione

```
▶ import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req = urlopen('http://web.unibas.it/bloisi/corsi/images/forme.png')
arr = np.array(bytearray(req.read()), dtype=np.uint8)
gray_img = cv.imdecode(arr, cv.IMREAD_GRAYSCALE) # Load as grayscale image

kernel = np.ones((21,21),np.uint8)
erosion = cv.erode(gray_img,kernel,iterations = 3)

cv2_imshow(erosion)
```

Qualcosa non torna!



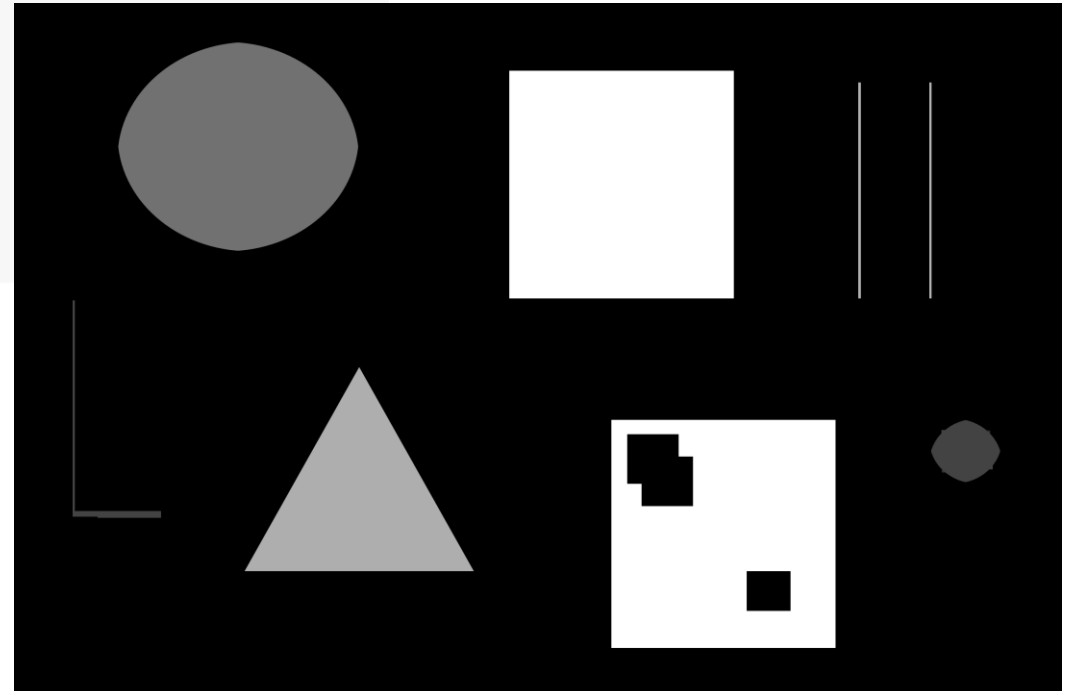
Erosione

```
▶ import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req = urlopen('http://web.unibas.it/bloisi/corsi/images/forme.png')
arr = np.array(bytearray(req.read()), dtype=np.uint8)
gray_img = cv.imdecode(arr, cv.IMREAD_GRAYSCALE) # Load as grayscale image
inv_img = cv.bitwise_not(gray_img) ←

kernel = np.ones((21,21),np.uint8)
erosion = cv.erode(inv_img,kernel,iterations = 3)

cv2_imshow(erosion)
```

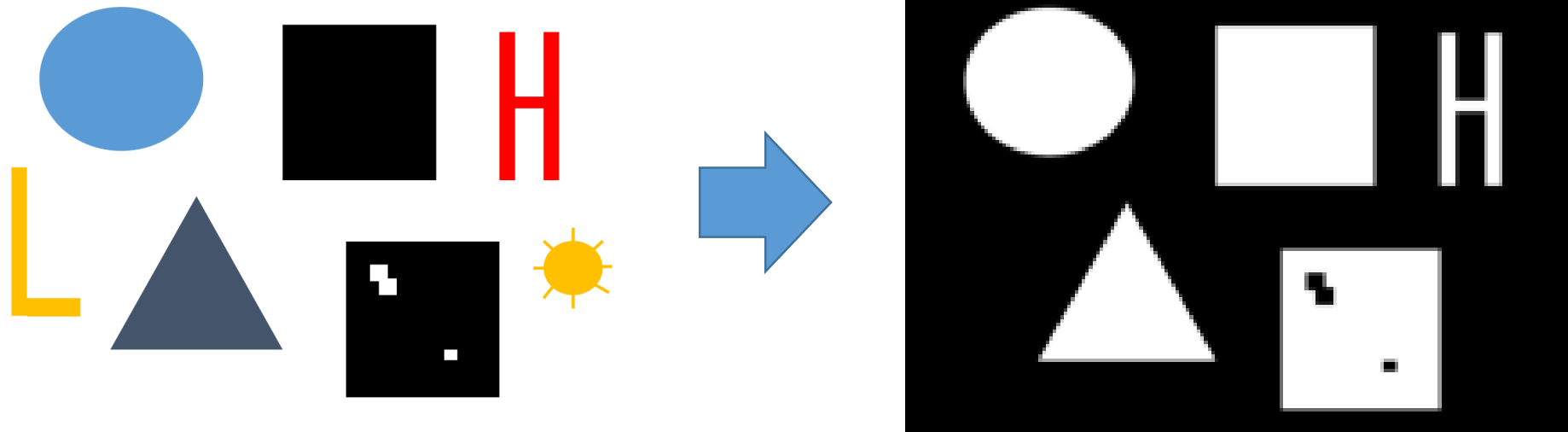


Esercizio 8 – Lezione 2.4

Applicare all'immagine

<https://web.unibas.it/bloisi/corsi/images/forme.png>

il metodo di thresholding di Otsu



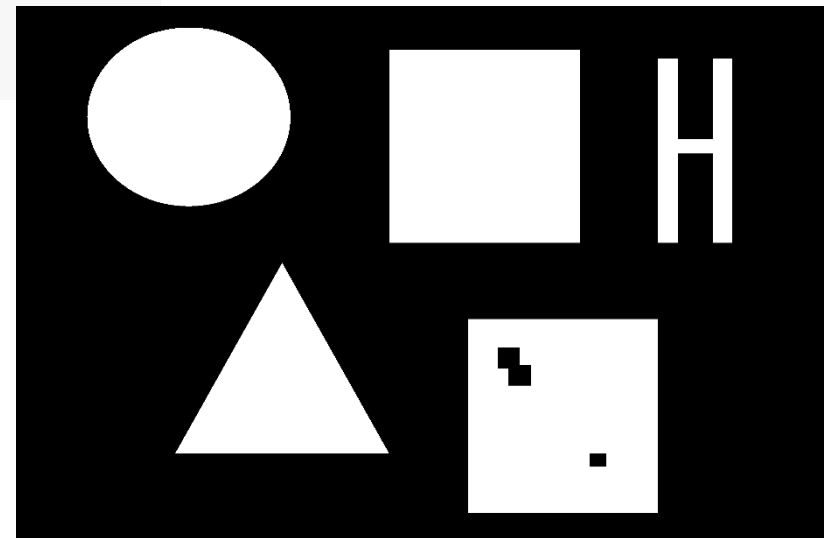
Soluzione con OpenCV

```
▶ import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req = urlopen('http://web.unibas.it/bloisi/corsi/images/forme.png')
arr = np.array(bytearray(req.read()), dtype=np.uint8)
gray_img = cv.imdecode(arr, cv.IMREAD_GRAYSCALE) # Load as grayscale image
inv_img = cv.bitwise_not(gray_img)

val,otsu = cv.threshold(inv_img,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
print("Otsu's threshold:",val)
cv2_imshow(otsu)
```

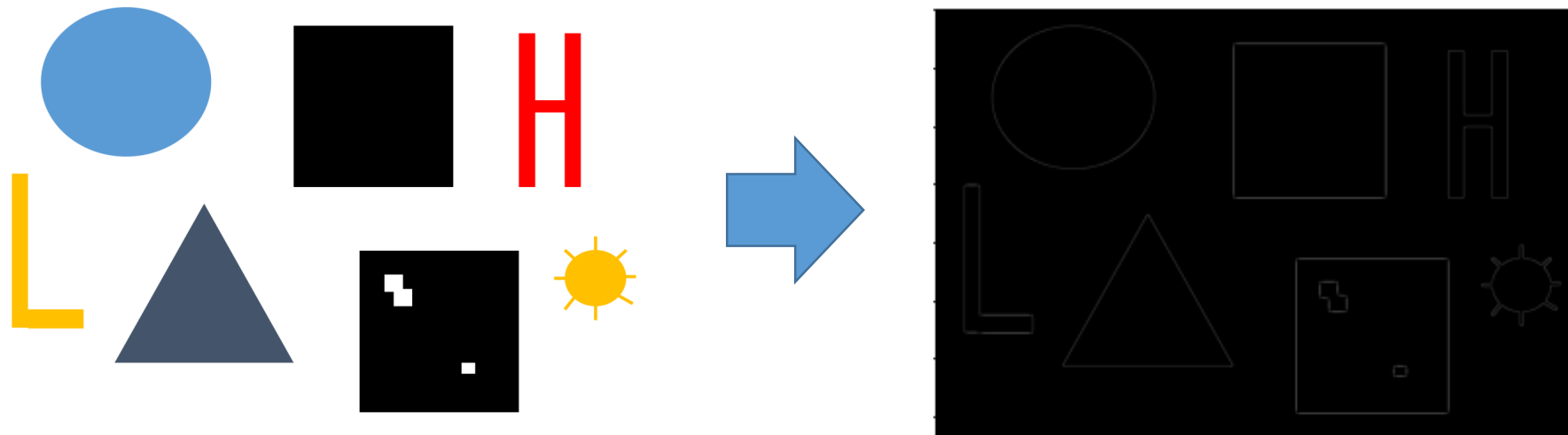
Otsu's threshold: 103.0



Esercizio 9 – Lezione 2.4

Estrarre i contorni dall'immagine

<https://web.unibas.it/bloisi/corsi/images/forme.png>



Soluzione con OpenCV

```
▶ import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req = urlopen('http://web.unibas.it/bloisi/corsi/images/forme.png')
arr = np.array(bytearray(req.read()), dtype=np.uint8)
gray_img = cv.imdecode(arr, cv.IMREAD_GRAYSCALE) # Load as grayscale image

edges = cv.Canny(gray_img,100,200)

cv2_imshow(edges)
```

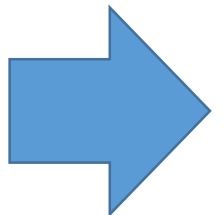


Harris corner detection in OpenCV



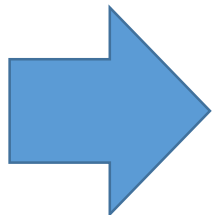
```
import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req = urlopen('http://web.unibas.it/bloisi/corsi/images/forme.png')
arr = np.array(bytearray(req.read()), dtype=np.uint8)
img = cv.imdecode(arr, cv.IMREAD_COLOR)
gray_img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```



Harris corner detection in OpenCV

```
thresh = 100
# Detector parameters
blockSize = 2
apertureSize = 3
k = 0.04
# Detecting corners
dst = cv.cornerHarris(gray_img, blockSize, apertureSize, k)
```

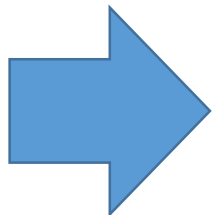


Harris corner detection in OpenCV

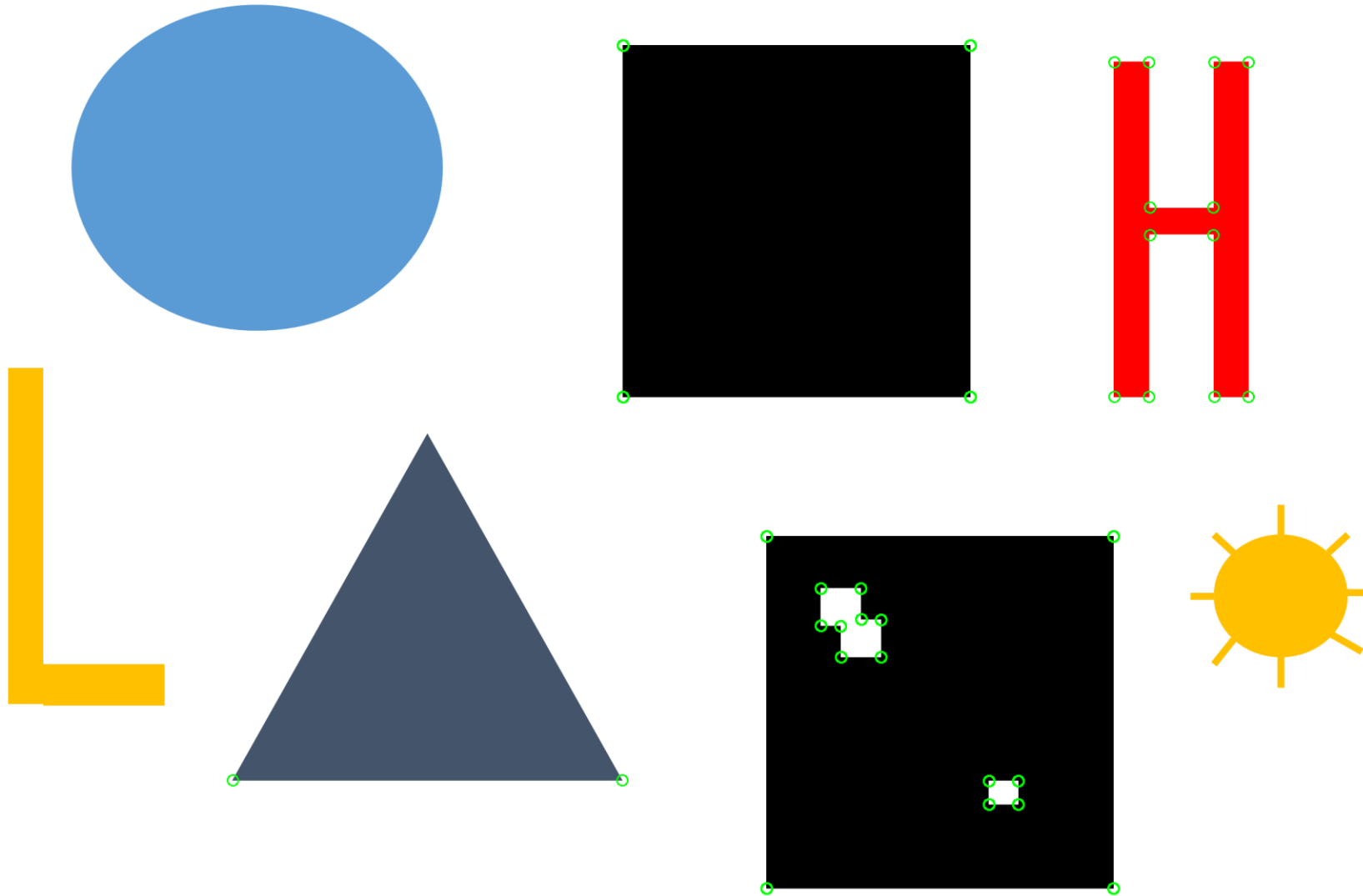
```
# Normalizing
dst_norm = np.empty(dst.shape, dtype=np.float32)
cv.normalize(dst, dst_norm, alpha=0, beta=255, norm_type=cv.NORM_MINMAX)

# Drawing a circle around corners
for i in range(dst_norm.shape[0]):
    for j in range(dst_norm.shape[1]):
        if int(dst_norm[i,j]) > thresh:
            cv.circle(img, (j,i), 10, (0,255,0), 2)

cv2_imshow(img)
```



Risultato Harris corner detection in OpenCV



Esercizio GoodFeatures

Usare la funzione OpenCV `goodFeaturesToTrack()` per trovare i corner nell'immagine

<https://web.unibas.it/bloisi/corsi/images/forme.png>

Suggerimento: si veda il tutorial a questo indirizzo

https://docs.opencv.org/4.1.2/d4/d8c/tutorial_py_shi_tomasi.html



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Visione e Percezione

OpenCV (Python)



Docente

Domenico D. Bloisi

