Image Analysis and Computer Vision with OpenCV

Content

- Image Analysis and Computer Vision
- OpenCV and Python
- OpenCV and Deep Learning
- Installing OpenCV
- First Approaches to Image Processing and Analysis
- Image Analysis
- Edge Detection and Image Gradient Analysis
- A Deep Learning Example: The Face Detection
- Conclusions

Image Analysis and Computer Vision

- In recent years, especially because of the development of deep learning, image analysis has experienced huge development in solving problems that were previously impossible, giving rise to a new discipline called ***computer vision*****.**
- Computer vision intends to process a two-dimensional image and extract the same levels of representation from it. This is done through various operations that can be classified as follows:

Image Analysis and Computer Vision

- *Detection*: Detect shapes, objects, or other subjects of investigation in an image (for example finding cars)
- *Recognition*: The identifed subjects are then led back to generic classes (for example, subdividing cars by brands and types)
- *Identification*: An instance of the previous class is identifed (for example, find my car)

OpenCV and Python

- *OpenCV (Open Source Computer Vision)* is a library written in C ++ that is specialized for computer vision and image analysis (https://opencv.org/).
- OpenCV supports many algorithms related to computer vision and machine learning and is expanding day by day.

OpenCV and Deep Learning

- There is a close relationship between computer vision and deep learning.
- Since 2017 was a significant year for the development of deep learning (read my article about it at http://www.meccanismocomplesso.org/en/2017-year-of-deep-learningframeworks/), the release of the new version of OpenCV 3.3 has seen the enhancement of the library with many new features of deep learning and neural networks in general

OpenCV and Deep Learning

- In fact, the library has a module called dnn (deep neural networks) dedicated to this aspect.
- This module has been specifically developed for use with many deep learning frameworks, including Caffe2, TensorFlow, and PyTorch

Installing OpenCV

- Installing a OpenCV package on many operating systems (Windows, iOS, and Android) is done through the official website (https://opencv.org/releases.html)

First Approaches to Image Processing and Analysis

- First you will start to see how to upload and view images
- Then you will pass some simple operations to them, add and subtract two images, and see an example of image blending.
- All these operations will be very useful as they will serve as a basis for any other image analysis operation.

Load and Display an Image

- img = cv2.imread('italy2018.jpg')
- If you see the content of the first element of the image, you will get the following:
- img[0]
  array([[38, 43, 11],
  [37, 42, 10],
  [36, 41, 9],
  ...,
  [24, 37, 15],
  [22, 36, 12],
  [23, 36, 12]], dtype=uint8)

Load and Display an Image

- You will now use the imshow() method to create a window with the image loaded in the variable img.This method takes two parameters—the window name and the image variable. Once you have created the window, you can use the waitKey() method
- cv2.imshow('Image', img)
  cv2.waitKey(0)

Load and Display an Image

- cv2.imshow('Image', img)
  cv2.waitKey(0)

Load and Display an Image

- cv2.imshow('Image', img)
  cv2.waitKey(2000)

- cv2.imshow('Image', img)
  cv2.waitKey(2000)
  cv2.destroyWindow('Image')

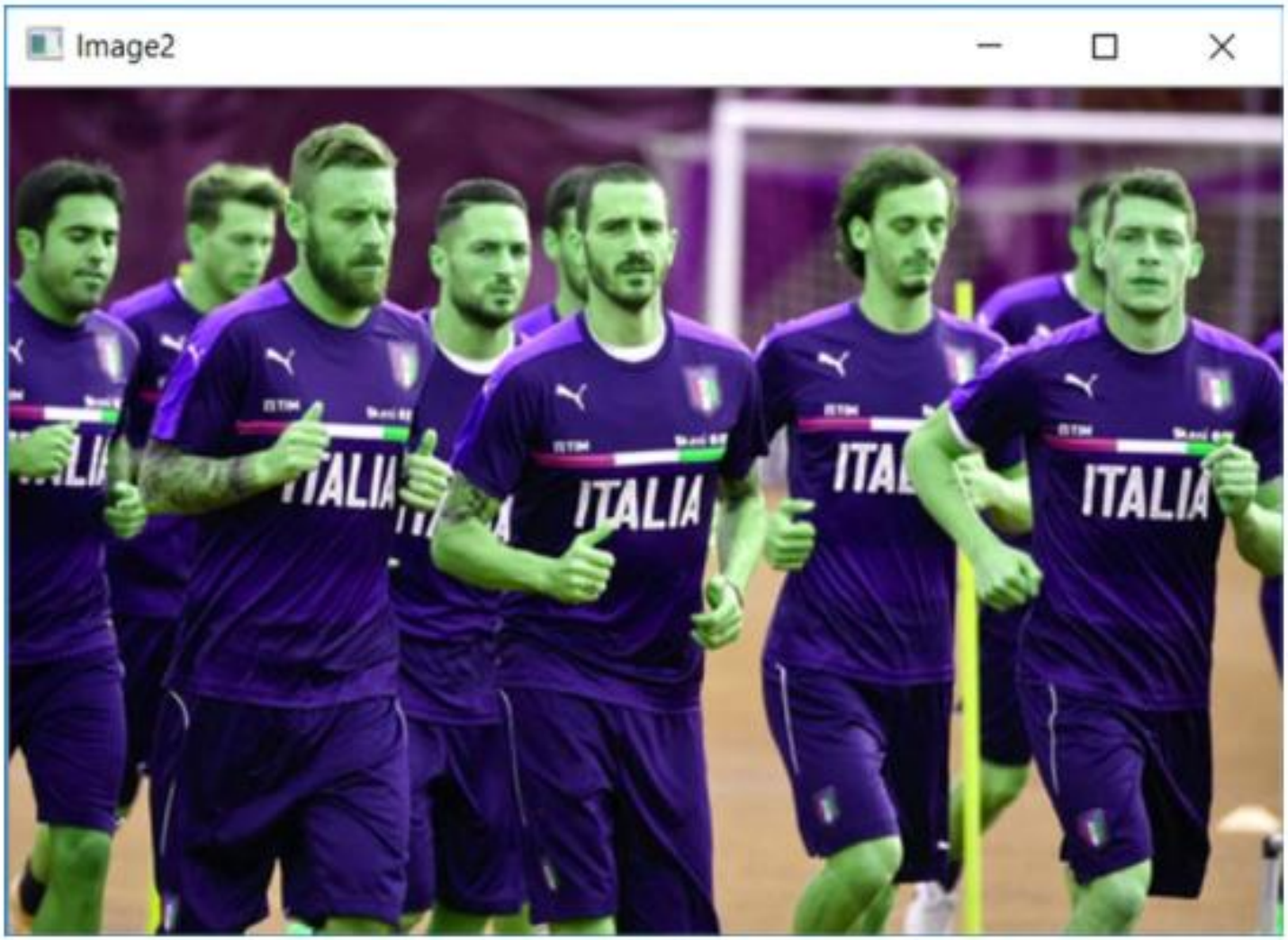
Working with Images

- After loading the image, decompose it into the three RGB channels. You can do this easily by using the split() method.
- b,r,g = cv2.split(img)

- img2 = cv2.merge((b,g,r))

- cv2.imshow('Image2', img2)
  cv2.waitKey(0)

Working with Images



Save the New Image

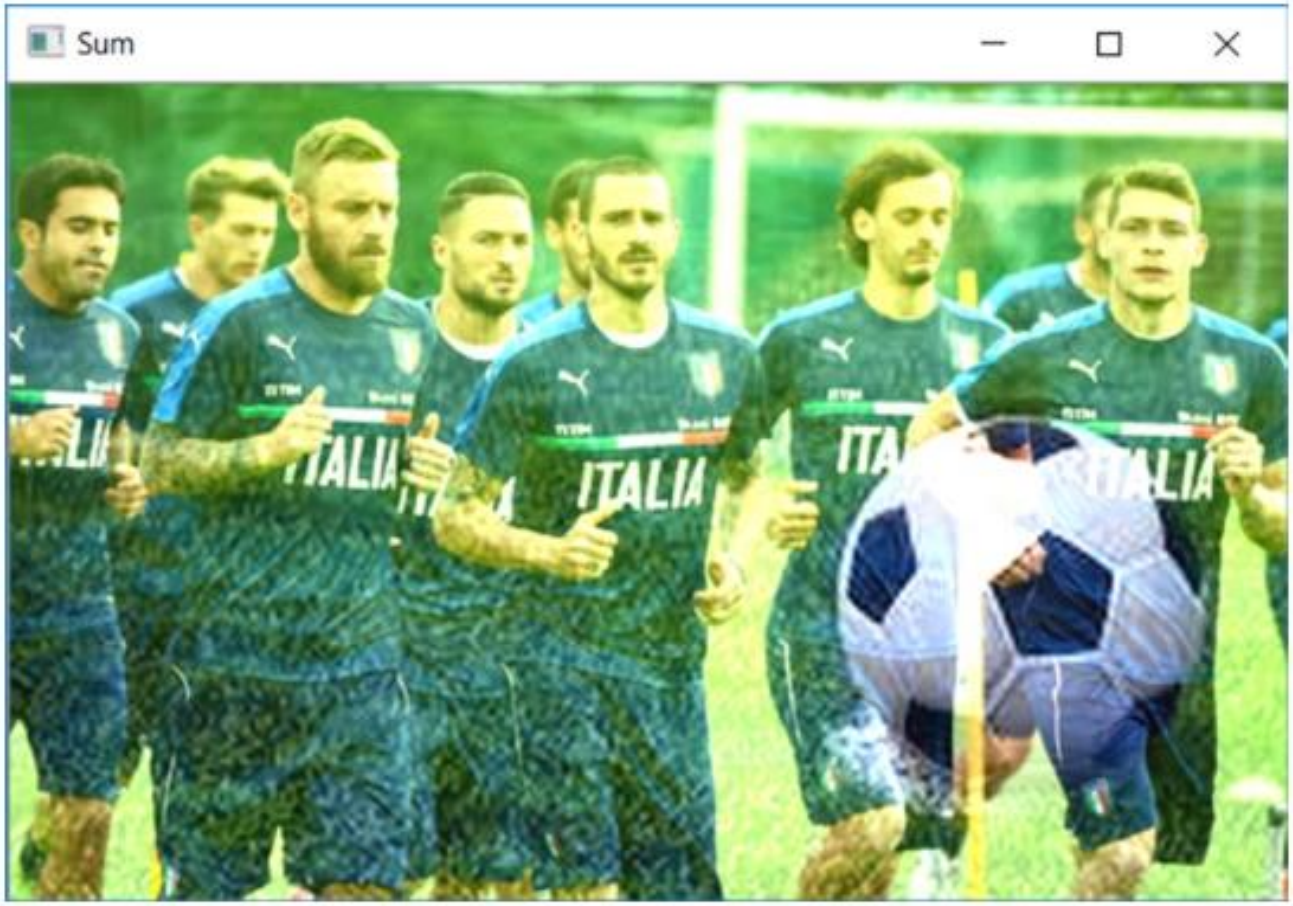
- cv2.imwrite('italy2018altered.png', img2)

Elementary Operations on Images

- img2 = cv2.imread('soccer.jpg')
  cv2.imshow('Image2', img2)
  cv2.waitKey(0)

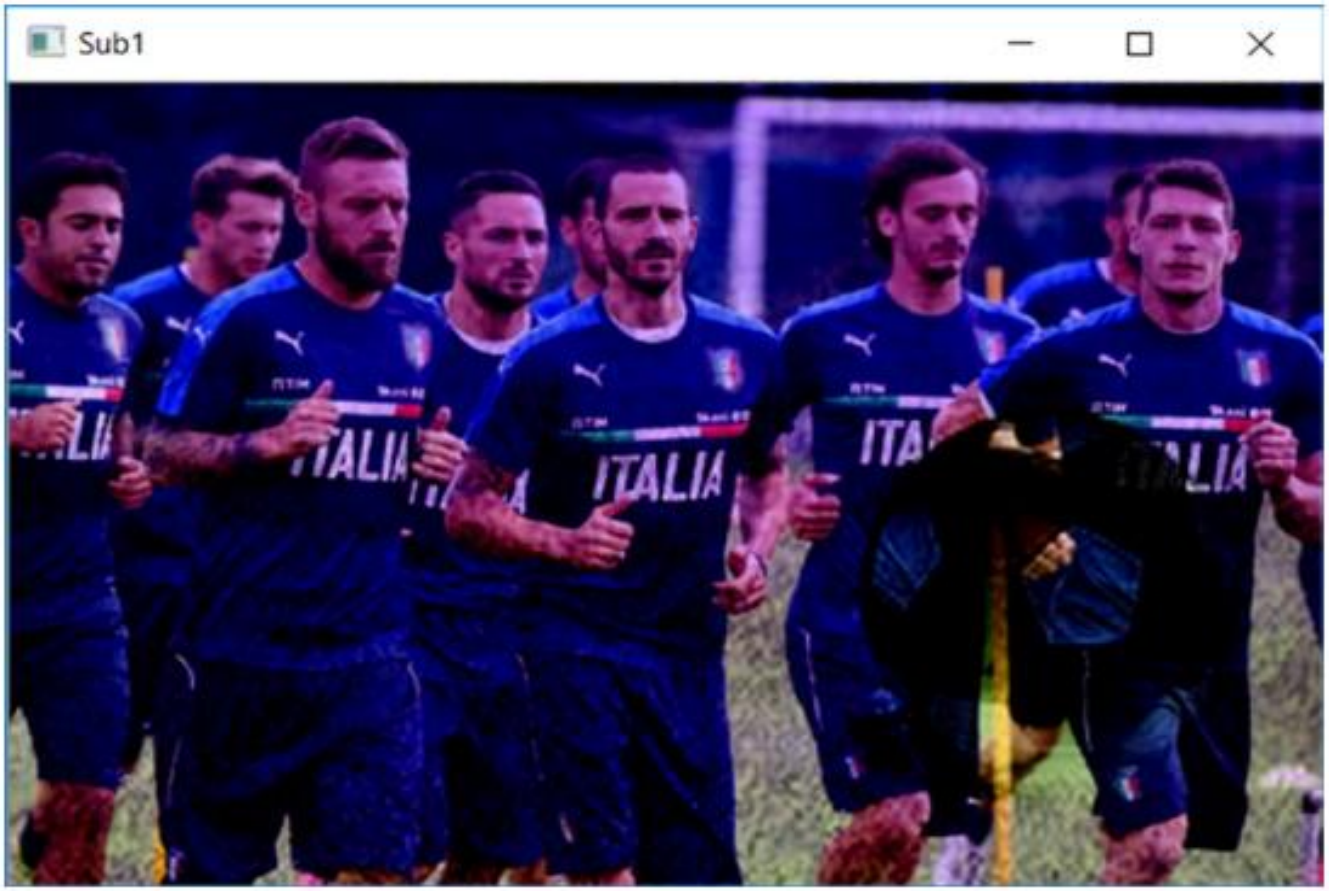
Elementary Operations on Images

- img = cv2.add(img,img2)
  cv2.imshow('Sum',img)



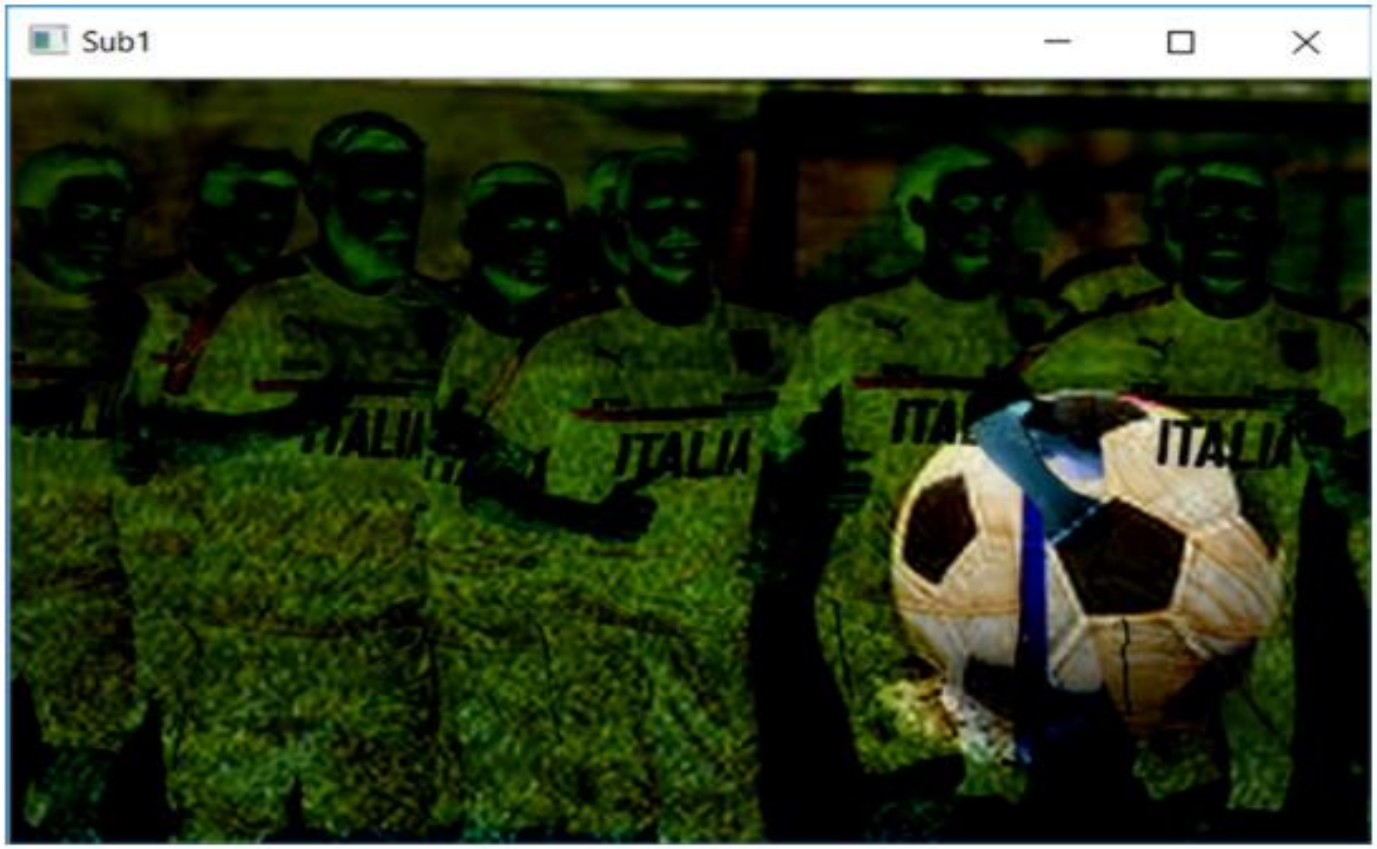
Elementary Operations on Images

- img3 = cv2.subtract(img, img2)
  cv2.imshow('Sub1',img3)
  cv2.waitKey(0)



Elementary Operations on Images

- img3 = cv2.subtract(img2, img)
  cv2.imshow('Sub1',img3)
  cv2.waitKey(0)



Elementary Operations on Images

- img3 = img + img2
  cv2.imshow('numpy',img3)
  cv2.waitKey(0)
- you will get an image with a very strong color contrast (they are the points over 255

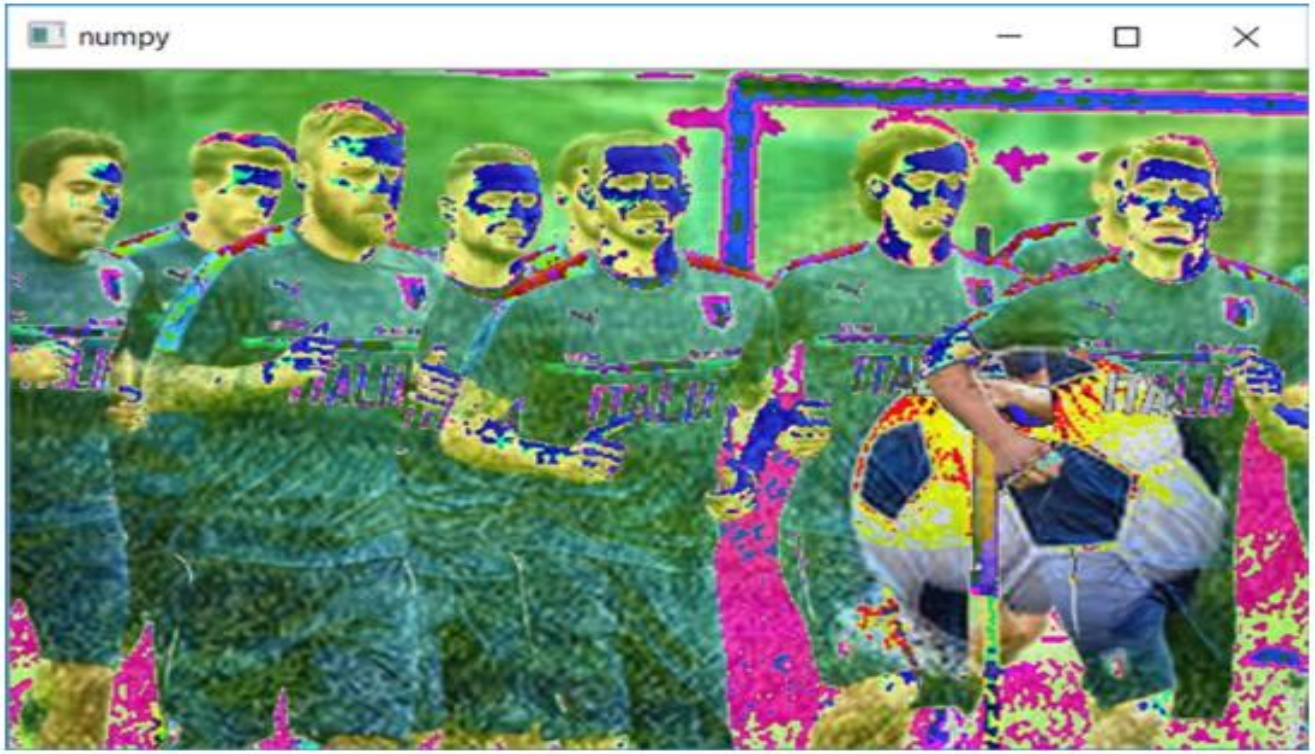


Image Blending

- img = α · img1 + (1 – α) · img2 with 0 ≥ α ≥ 1
  img3 = cv2.addWeighted(img, 0.3, img2, 0.7, 0)
  cv2.imshow('numpy',img3)
  cv2.waitKey(0)

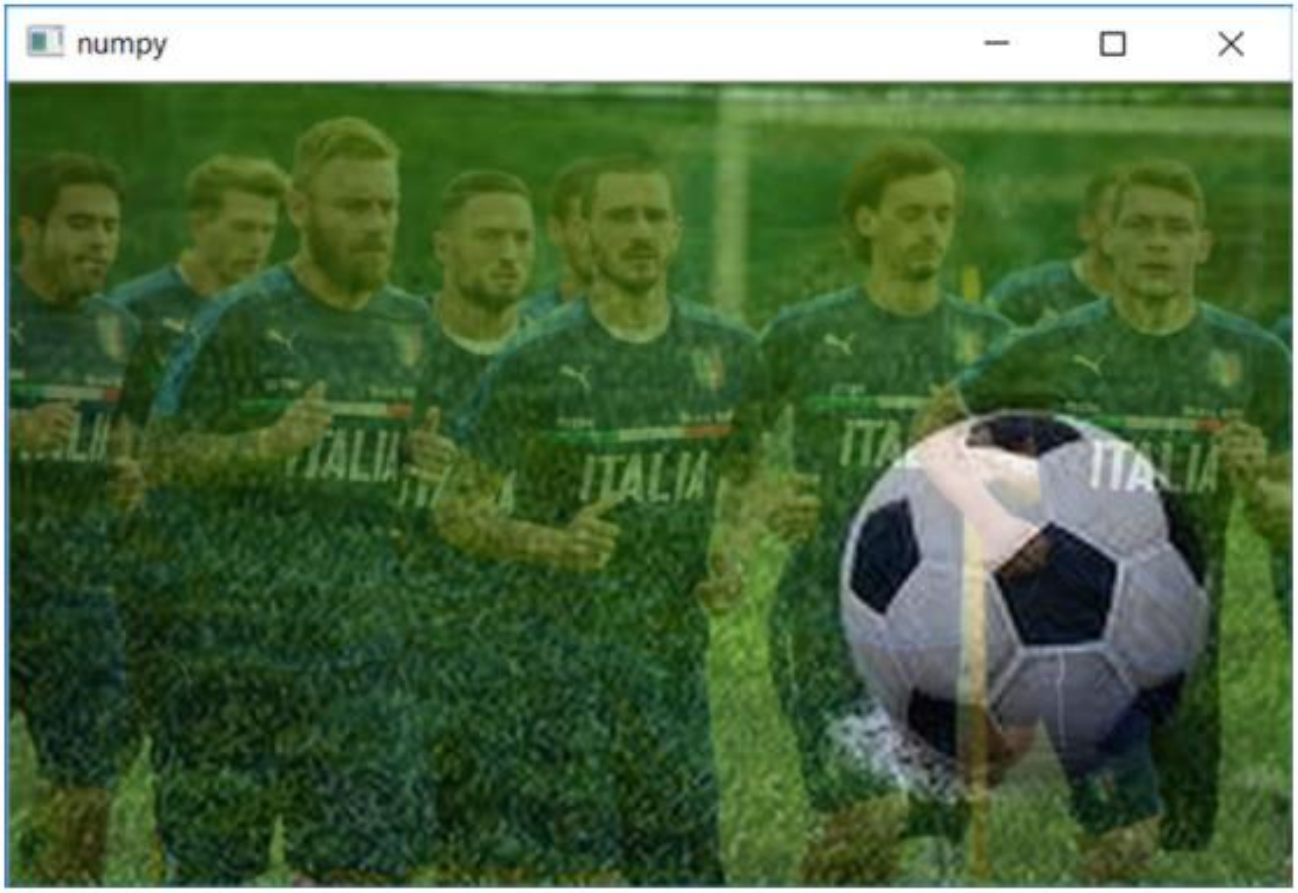
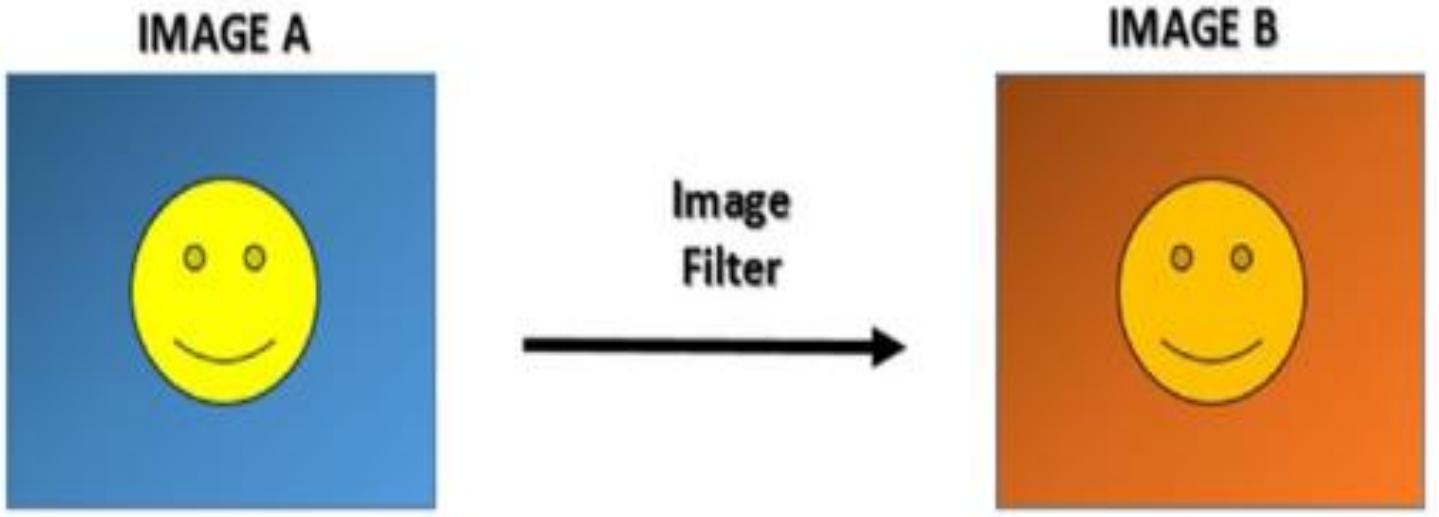


Image Analysis



$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix} \xrightarrow{f()} B = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix}$$

Edge Detection and Image Gradient Analysis

- In the previous sections, you saw how to perform some basic operations that are useful for image analysis. In this section, you start with a real case of image analysis, called *edge detection*.

Edge Detection

- To understand the geometries represented, it is necessary to recognize the outlines that delimit an object from the background or from other objects.
  This is precisely the task of the edge detection.
- In edge detection, a great many algorithms and techniques have been developed and they exploit different principles in order to determine the contours of objects correctly.
- Many of these techniques are based on the principle of color gradients, and exploit the image gradient analysis process.

The Image Gradient Theory

- *Convolutions* of an image
- *Derivative*
- *Gradient*

A Practical Example of Edge Detection with the Image Gradient Analysis

- *Convolutions* of an image
- *Derivative*
- *Gradient*

A Deep Learning Example: The Face Detection

Conclusions

- In this chapter, you saw some simple examples of techniques that form the basis of image analysis and in particular of computer vision.
- In fact, you saw how images are processed through image filters, and how some complex techniques can be built using edge detection. You also saw how computer vision works by using deep learning neural networks to recognize faces in an image (face detection).
- I hope this chapter is a good starting point for your further insights on the subject. If you are interested, you will find in-depth information on this topic on my website at https://meccanismocomplesso.org.