Computer vision in the new era of Artificial Intelligence and Deep Learning

Visión por computador en la nueva era de la Inteligencia Artificial y el Deep Learning

Rubén Usamentiaga*, Alberto Fernández°
*** University of Oviedo**
° TSK

Gijón (Spain)
5 – 16 April 2021

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

OpenCV

Basic image treatment in OpenCV

OpenCV

Notebook: basic_image_treatment_with_opencv.ipynb

- basic_image_treatment_with_opencv.ipynb

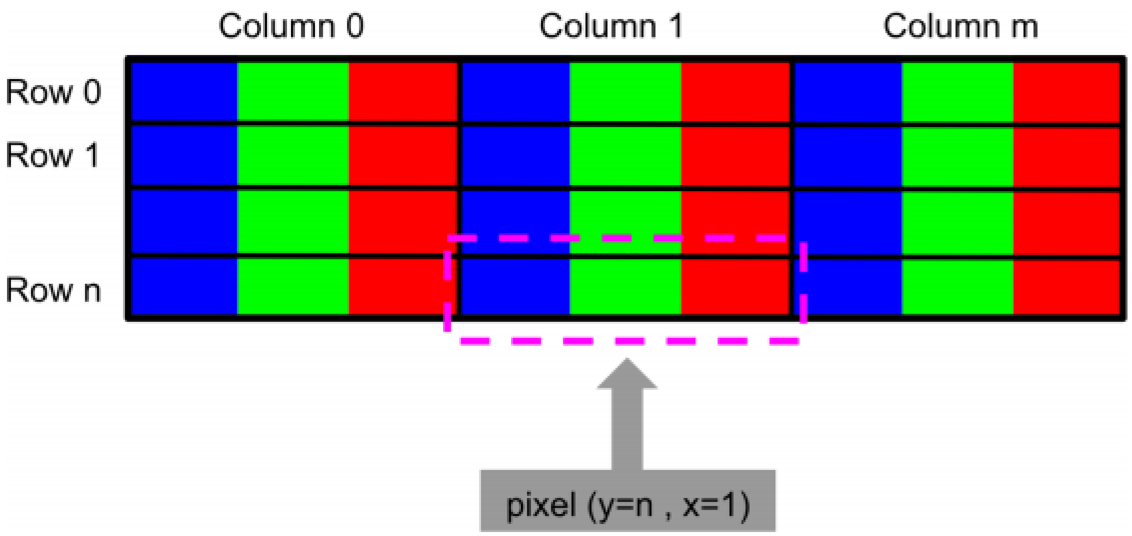
Open in Colab

- basic_image_treatment_with_opencv.ipynb

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

BGR color format in OpenCV

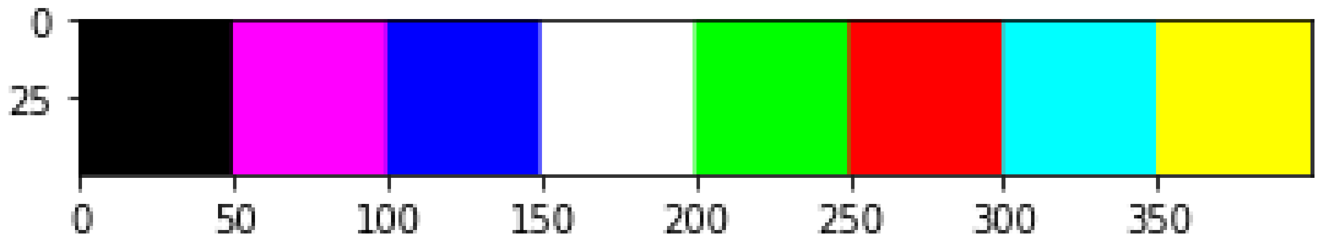
- Accessing one pixel in OpenCV give us three values corresponding to the blue, green and red channels



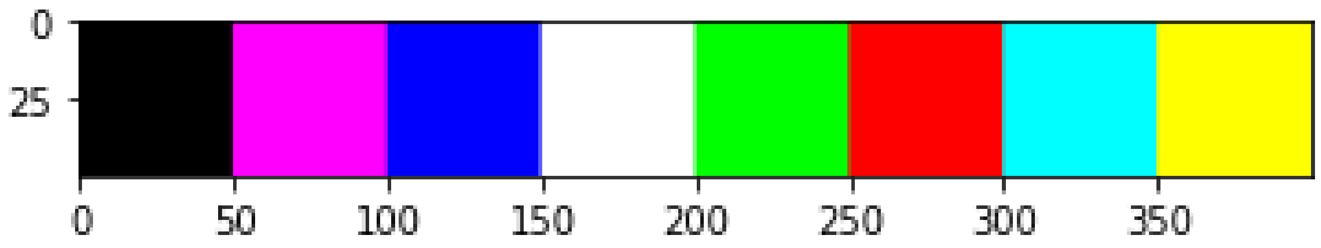
Sample color image (using NumPy)

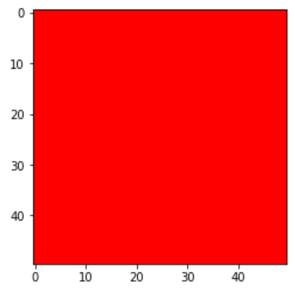
```
import numpy as np

def build_sample_image_color(colors):
    """Builds a sample image with 50x50 regions of different colors"""

    # Initialize result with the first 50x50 region with black color
    result = np.zeros((50, 50, 3), dtype="uint8")

    # Build the image concatenating horizontally the regions:
    for color in colors:
      img = np.ones((50, 50, 3), dtype="uint8") * color
      result = np.concatenate((result, img), axis=1)

    return result
```

```
colors = np.array([(255,0,255),(255,0,0),(255
,255,255),(0,255,0),(0,0,255),(255,255,0),(0,
255,255)], dtype="uint8")
color_img = build_sample_image_color(colors)
```



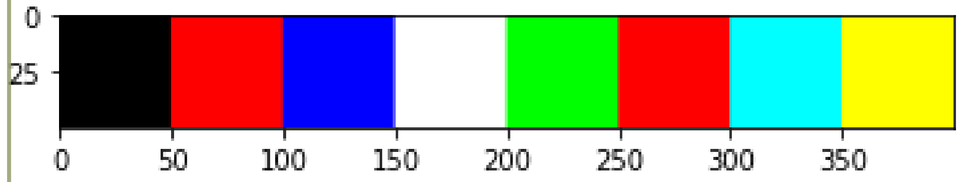
Working with color images





```
# Get the dimensions (shape):
color_img.shape: (50, 400, 3)

# Get total number of elements:
color_img.size: 60000

# Type of the elements:
color_img.dtype: uint8

# Get the pixel value at y=10,x=60
color_img[10, 60]: (255,0,255)

# Get only green value at y=10,x=310
color_img[10, 310, 1]: 255

# Set value to the pixel y=10,x=20
color_img[10, 20] = (255, 255, 0)
```

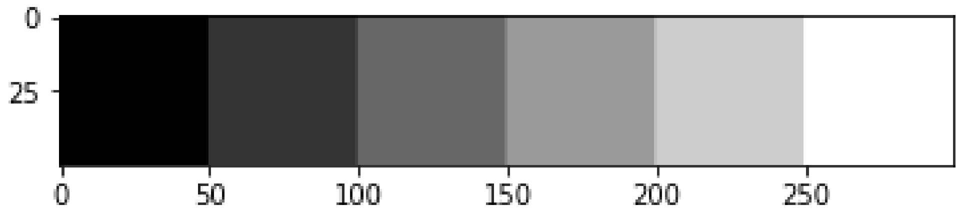
```
# Get ROI of the image:
roi_red = color_img[0:50, 250:300]

# Check the shape:
roi_red.shape: (50, 50, 3)

# Set ROI of the image
color_img[0:50, 50:100] = roi_red
```



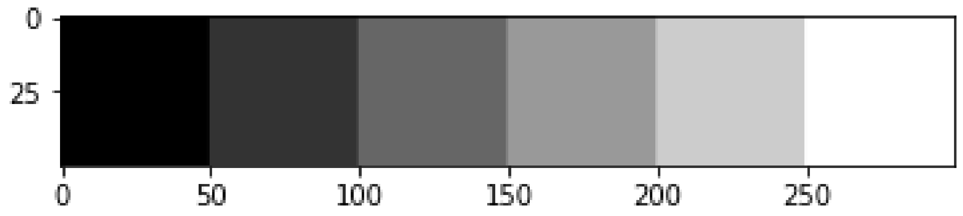
Sample gray image (using NumPy)

```
import numpy as np

def build_sample_image_grayscale():
    """Builds a sample image with 50x50 regions of different tones of gray"""

    # Define the different tones.
    # The end of interval is not included
    tones = np.arange(start=50, stop=300, step=50)
    # print(tones)

    # Initialize result with the first 50x50 region with 0-intensity level
    result = np.zeros((50, 50), dtype="uint8")

    # Build the image concatenating horizontally the regions:
    for tone in tones:
        img = np.ones((50, 50), dtype="uint8") * tone
        result = np.concatenate((result, img), axis=1)

    return result
```

```
# Create the gray image using the function defined above:
gray_img = build_sample_image_grayscale()
```



Working with gray images



```
# Get the dimensions (shape):
gray_img.shape: (50, 300)

# Get total number of elements
gray_img.size: 15000

# Type of the elements:
gray_img.dtype: uint8

# Get the pixel value at y=10,x=60
gray_img[10, 60]: 50

# Set value to the pixel y=10,x=20
gray_img[10, 20] = 65
```

OpenCV

Basic image treatment in OpenCV

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course