JÖNKÖPING UNIVERSITY
School of Engineering

DIGITAL IMAGE PROCESSING

A practical introduction with Python and OpenCV

2024-04-17 - Jérôme Landré – jerome.landre@ju.se

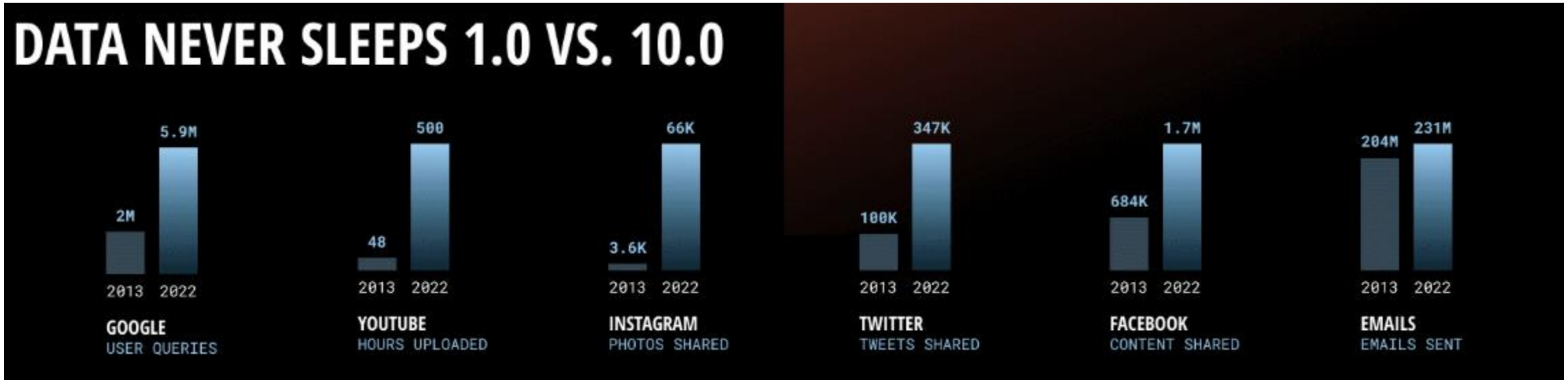
OUTLINE

1. What is Digital Image Processing?
2. What is an image?
3. Methodology for Image Processing
    1. Images sources
    2. Fixed or moving camera
    3. Type of images
    4. Color images or not…
    5. Background removal
    6. Edges and regions
    7. Region of interest
    8. Extract information
    9. Machine Learning
    10. Tools

1. What is Digital Image Processing?

DIGITAL IMAGE PROCESSING

- More and more digital images are produced every day from our phones, cameras, computers…
- Every MINUTE on the Internet:



Source: https://www.domo.com/data-never-sleeps

DIGITAL IMAGE PROCESSING

- “Digital image processing is the use of a digital computer to process digital images through an algorithm.” from https://en.wikipedia.org/wiki/Digital_image_processing

DIGITAL IMAGE PROCESSING

- The goal is to extract useful information from images in order to detect/track/find/identify/sort/classify objects in one or more image(s).
- Applications of Digital Image Processing:
  - Image restoration,
  - Robot vision,
  - Medical Imaging,
  - Pattern recognition,
  - Image/Video analysis…

2. What is a digital image?

WHAT IS A DIGITAL IMAGE?

WHAT IS A DIGITAL IMAGE?

- A 2D array M of pixels

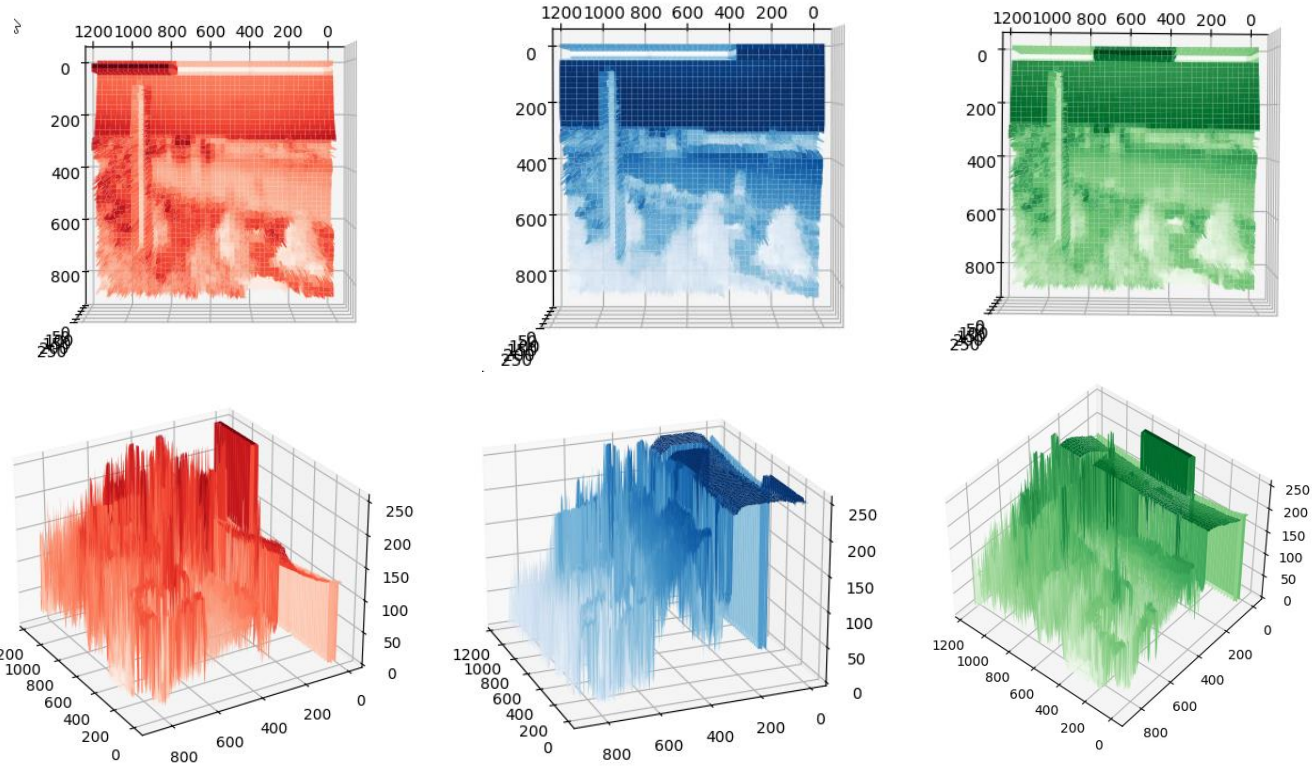
#RGB: 755844

WHAT IS A DIGITAL IMAGE?

- A 2D array of pixels

#RGB: 755844

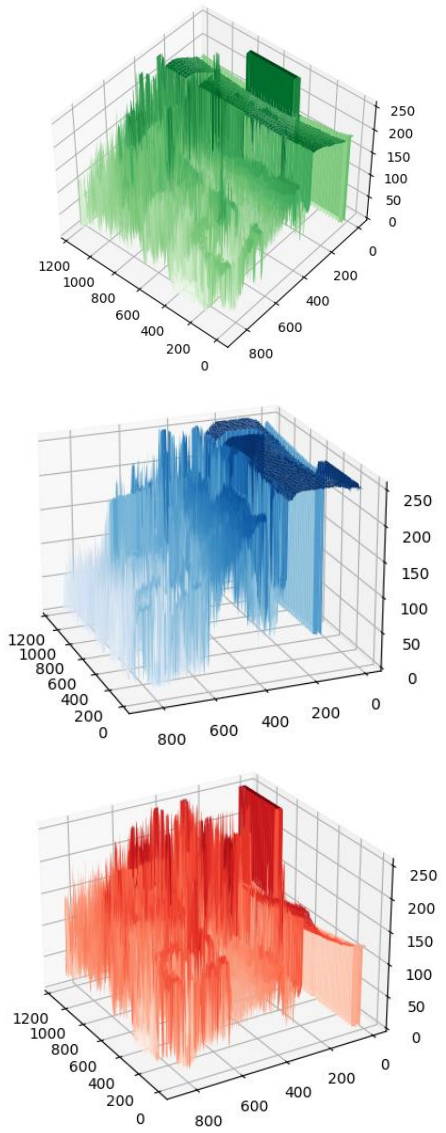
- A 2D signal f(x,y) obtained from a real measurable phenomenon



GENERATING 3D VIEWS

improc-0005.py > ...

```python
# we import all libraries used in our program
import cv2
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
# we read an image from disk to memory
img = cv2.imread('img/jonkoping-01.jpg')
# we flip the image to get the good view during 3d projection
img2 = cv2.flip(src = img, flipCode = 1)
# we show the image and wait for a key to be pressed
cv2.imshow('Jonkoping', img)
key = cv2.waitKey(0)
# 3d mesh grid of an image
#source: https://www.geeksforgeeks.org/3d-surface-plotting-in-python-using-matplotlib/
# we make the data from our image
X = np.arange(img2.shape[1])
Y = np.arange(img2.shape[0])
X, Y = np.meshgrid(X, Y)
# we plot the surface
fig1, ax1 = plt.subplots(subplot_kw={"projection": "3d"})
fig2, ax2 = plt.subplots(subplot_kw={"projection": "3d"})
fig3, ax3 = plt.subplots(subplot_kw={"projection": "3d"})
ax1.plot_surface(X, Y, img2[:,:,0], vmin = img2[:,:,0].min() * 2, cmap = cm.Blues)
ax2.plot_surface(X, Y, img2[:,:,1], vmin = img2[:,:,1].min() * 2, cmap = cm.Greens)
ax3.plot_surface(X, Y, img2[:,:,2], vmin = img2[:,:,2].min() * 2, cmap = cm.Reds)
# we show the surfaces
plt.show()
cv2.destroyAllWindows()

```



WHAT IS A DIGITAL IMAGE?

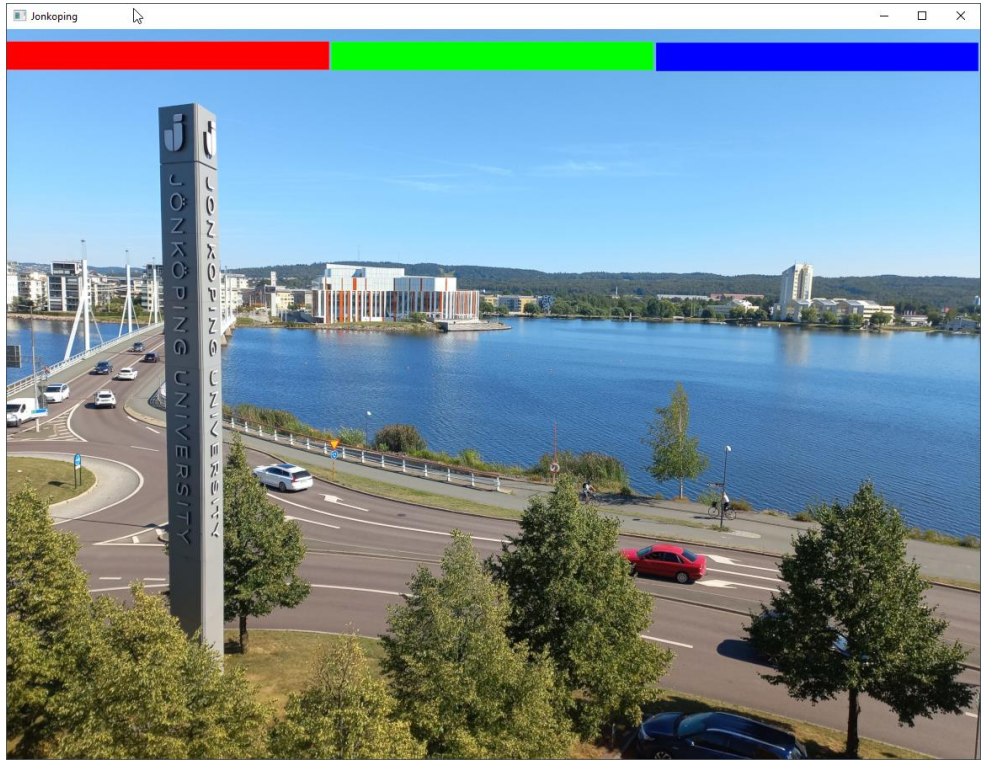
- A 2D array M of pixels
  - Many algorithms available working on pixels, regions, sliding window, columns, rows:
    - Color spaces
    - Image resizing
    - Image thresholding
    - Image transformations: translation, rotation, scale, perspective projections…
    - Image segmentation
    - Sliding Window
    - …

- A 2D signal f(x,y) obtained from a real measurable phenomenon
  - Many algorithms coming from signal processing available:
    - Image filtering (convolution)
    - Edge detection
    - Fourier Transform
    - Wavelets Transforms
    - …

READING AN IMAGE

improc-0000.py > ...

```python
# we import all libraries used in our program
import cv2

# we read an image from disk to memory
img = cv2.imread('img/jonkoping-01.jpg')

# we print the information concerning this image
print('---> Shape: ', img.shape)
print('---> Size: ', img.size)
print('---> Number of dimensions', img.ndim)
print('---> Image nd-array: ', img)

# we show the image and wait for a key to be pressed
cv2.imshow('Jonkoping', img)
key = cv2.waitKey(0)

cv2.destroyAllWindows()

```



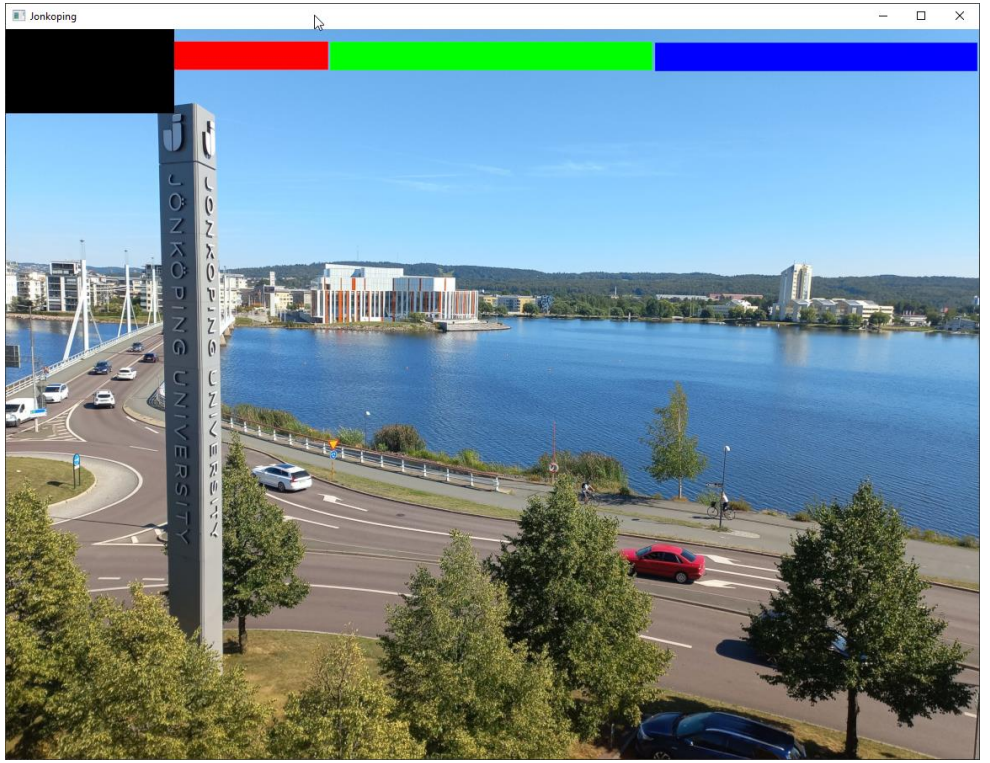

```
C:\Users\lanjer\Desktop\box\ju2023\teaching\courses\semester2\intelligent-mobile-platform\cours-jerome\code>python improc-0000.py
---> Shape:  (867, 1156, 3)
---> Size:  3006756
---> Number of dimensions 3
---> Image nd-array:  [[[244 183 119]
  [244 183 119]
  [244 181 120]
  ...
  [247 189 124]
  [247 189 124]
  [247 189 124]]

 [[244 183 119]
  [244 183 119]
  [244 181 120]
  ...
  [247 189 124]
```

Be careful: OpenCV uses the **BGR** pixel representation! For instance the first pixel of the image is [244 183 169], It is blue (upper left part of the sky) because of the high blue value: 244.

CHANGING THE PIXELS VALUES (1)

improc-0001.py > ...

```python
# we import all libraries used in our program
import cv2

# we read an image from disk to memory
img = cv2.imread('img/jonkoping-01.jpg')

# we change some pixels values
img[0:100, 0:200, :] = 0

# we print the information concerning this image
print('Shape: ', img.shape)
print(img)

# we show the image and wait for a key to be pressed
cv2.imshow('Jonkoping', img)
key = cv2.waitKey(0)

cv2.destroyAllWindows()

```

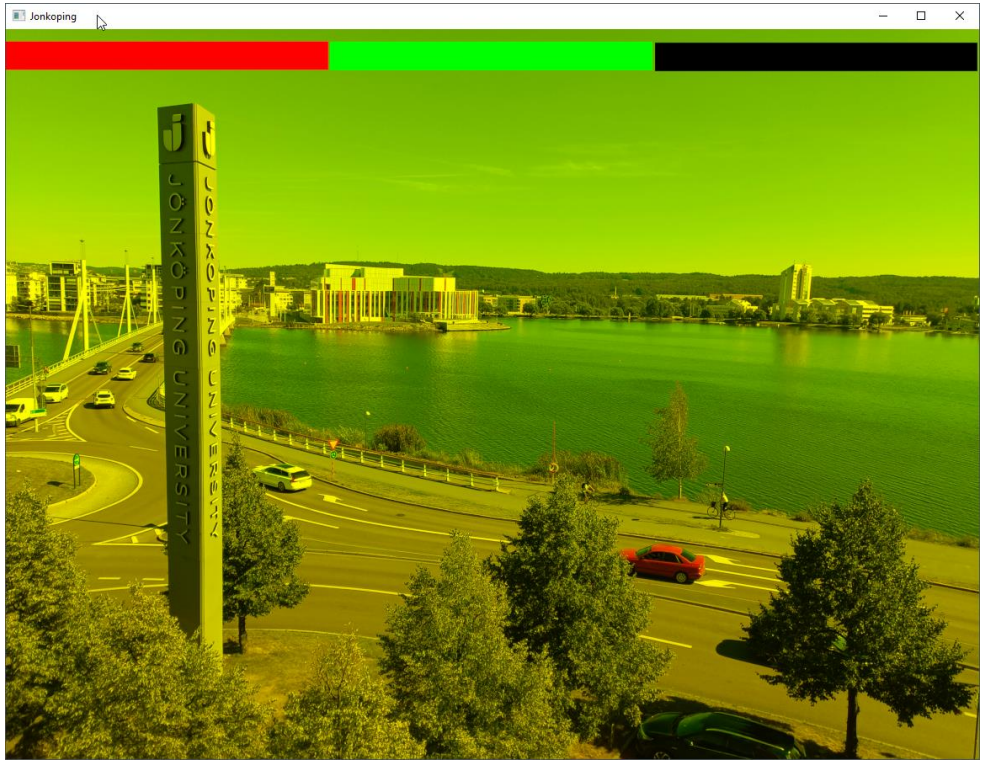



```
PS C:\Users\lanjer\Desktop\ju2022\teaching\COURSES\semester2\intelligent-mobile-platform\cours-jerome\code> python .\improc-0001.py
Shape:  (867, 1156, 3)
[[[  0   0   0]
  [  0   0   0]
  [  0   0   0]
  ...
  [247 189 124]
  [247 189 124]
  [247 189 124]]

 [[  0   0   0]
```

CHANGING THE PIXELS VALUES (2)

improc-0002.py > ...

```python
# we import all libraries used in our program
import cv2

# we read an image from disk to memory
img = cv2.imread('img/jonkoping-01.jpg')

# we change some pixels values
img[:, :, 0] = 0

# we print the information concerning this image
print('Shape: ', img.shape)
print(img)

# we show the image and wait for a key to be pressed
cv2.imshow('Jonkoping', img)
key = cv2.waitKey(0)

cv2.destroyAllWindows()

```




```
PS C:\Users\lanjer\Desktop\ju2022\teaching\COURSES\semester2\intelligent-mobile-platform\cours-jerome\code> python .\improc-0002.py
Shape:  (867, 1156, 3)
[[[  0 183 119]
  [  0 183 119]
  [  0 181 120]
  ...
  [  0 189 124]
  [  0 189 124]
  [  0 189 124]]

 [[  0 183 119]
```

3. Methodology for Image Processing

METHODOLOGY FOR IMAGE PROCESSING

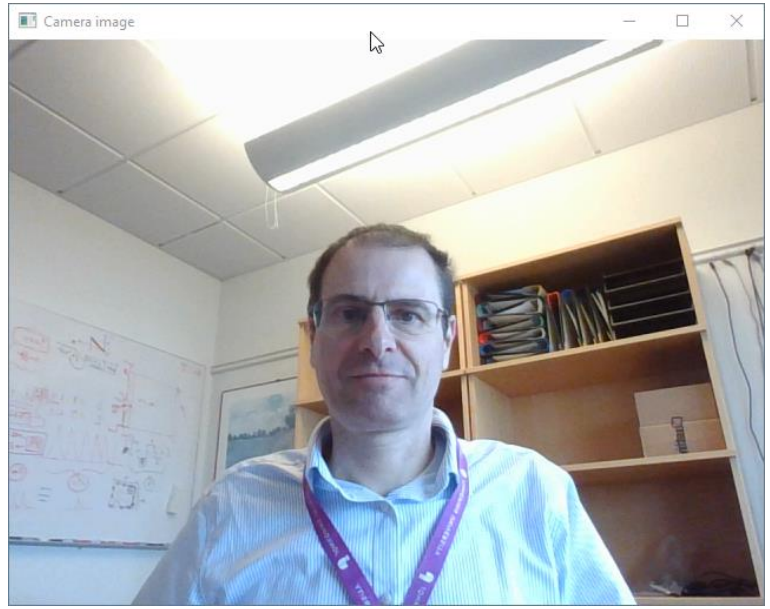
- Where to start?
  - 1) How many image sources do I have/need?
  - 2) Is the camera is fixed or moving?
  - 3) What kind of images do I need?
  - 4) Do I need color images?
  - 5) Do I need to remove the background of my images?
  - 6) Do I need to find edges or regions in my images?
  - 7) Will I work on the whole image or on a region?
  - 8) What information do I want to extract from my images? How?
  - 9) Do I need Machine Learning to solve my problem?
  - 10) Can I find useful tools included in OpenCV?
  - …

1) IMAGE SOURCES (WEBCAM)

- How many image sources do I have/need?
  - It depends on the application you want to build.
  - In general, we have an infinite loop to capture images from the camera and we process the images in the heart of this loop

improc-0003.py > ...

```python
# we import all libraries used in our program
import cv2

# Initialize Camera
cap = cv2.VideoCapture(0)

# infinite loop
while True:
    # we read an image from the camera
    ret, frame = cap.read()
    # there is an image
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        # we display it
        cv2.imshow('Camera image', frame)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

# we close all windows
cv2.destroyAllWindows()

```



1) IMAGE SOURCES (INTEL REALSENSE)

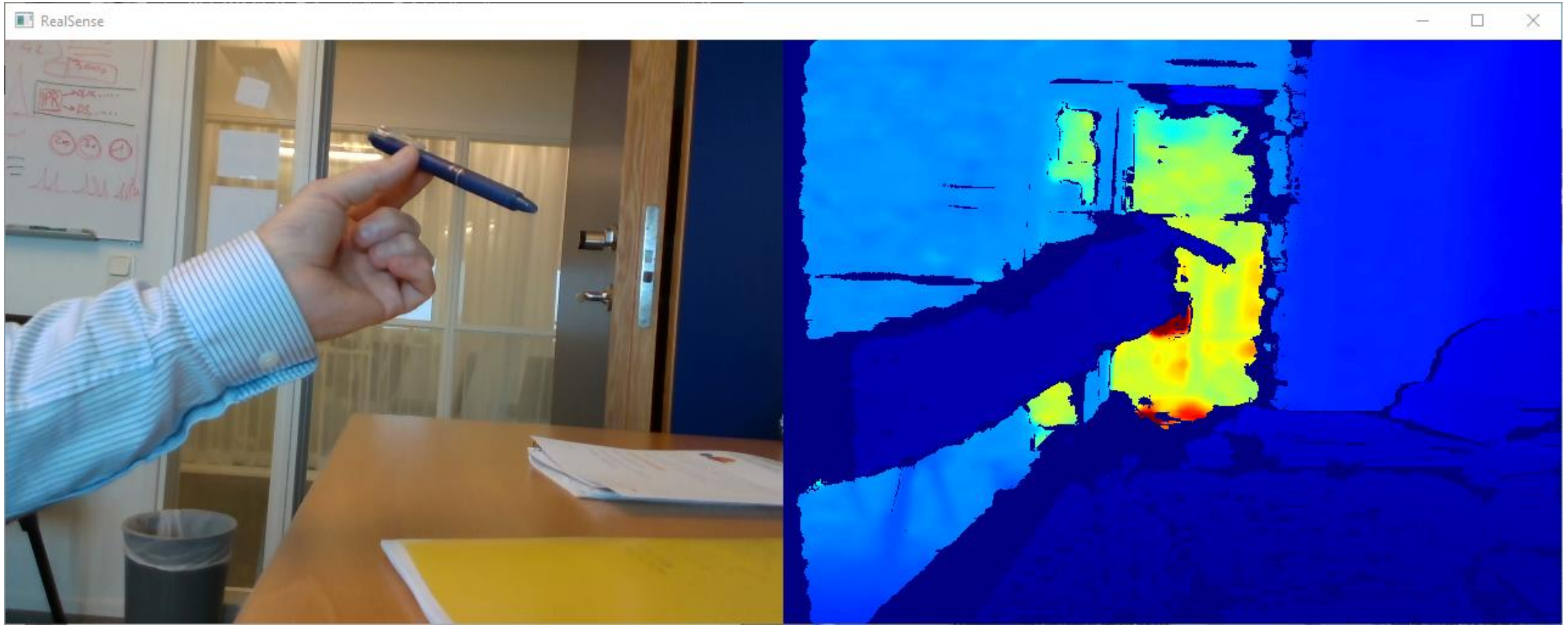
improc-0020.py > ...

```python
import pyrealsense2 as rs
import numpy as np
import cv2
# Configure depth and color streams
pipeline = rs.pipeline()
config = rs.config()
# Get device product line for setting a supporting resolution
pipeline_wrapper = rs.pipeline_wrapper(pipeline)
pipeline_profile = config.resolve(pipeline_wrapper)
device = pipeline_profile.get_device()
device_product_line = str(device.get_info(rs.camera_info.product_line))

found_rgb = False
for s in device.sensors:
    if s.get_info(rs.camera_info.name) == 'RGB Camera':
        found_rgb = True
        break
if not found_rgb:
    print("The demo requires Depth camera with Color sensor")
    exit(0)

config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
if device_product_line == 'L500':
    config.enable_stream(rs.stream.color, 960, 540, rs.format.bgr8, 30)
else:
    config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
# Start streaming
pipeline.start(config)
```

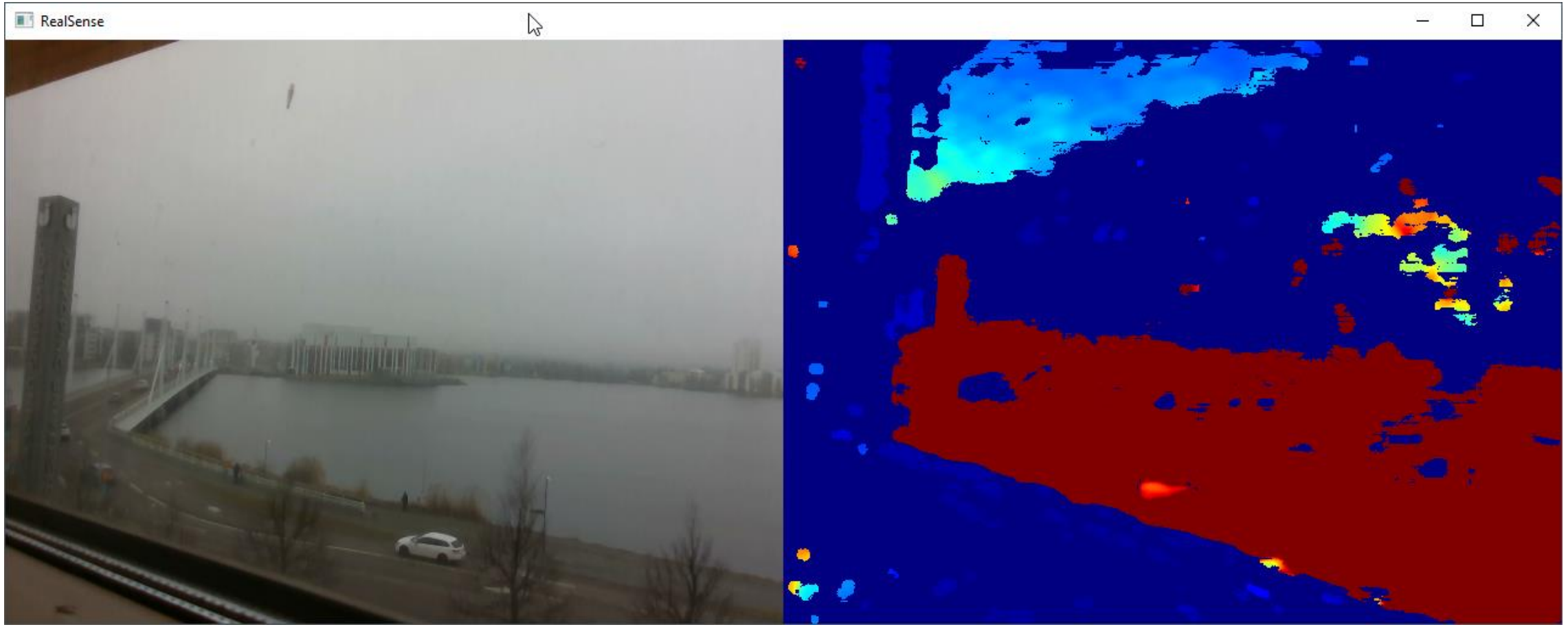
```python
try:
    while True:
        # Wait for a coherent pair of frames: depth and color
        frames = pipeline.wait_for_frames()
        depth_frame = frames.get_depth_frame()
        color_frame = frames.get_color_frame()
        if not depth_frame or not color_frame:
            continue

        # Convert images to numpy arrays
        depth_image = np.asanyarray(depth_frame.get_data())
        color_image = np.asanyarray(color_frame.get_data())

        # Apply colormap on depth image (image must be converted to 8-bit per pixel first)
        depth_colormap = cv2.applyColorMap(cv2.convertScaleAbs(depth_image, alpha=0.03), cv2.COLORMAP_JET)

        depth_colormap_dim = depth_colormap.shape
        color_colormap_dim = color_image.shape

        # If depth and color resolutions are different, resize color image to match depth image for display
        if depth_colormap_dim != color_colormap_dim:
            resized_color_image = cv2.resize(color_image, dsize=(depth_colormap_dim[1],
                                                                  depth_colormap_dim[0]), interpolation=cv2.INTER_AREA)
            images = np.hstack((resized_color_image, depth_colormap))
        else:
            images = np.hstack((color_image, depth_colormap))
        # Show images
        cv2.namedWindow('RealSense', cv2.WINDOW_AUTOSIZE)
        cv2.imshow('RealSense', images)
        k = cv2.waitKey(1)
        if k==ord('q'):
            break
finally:
    # Stop streaming
    pipeline.stop()
    cv2.destroyAllWindows()

```

Be careful: the code works with an Intel Realsense camera **only**,
If you have another camera, you must change this code.

1) IMAGE SOURCES (INTEL REALSENSE)



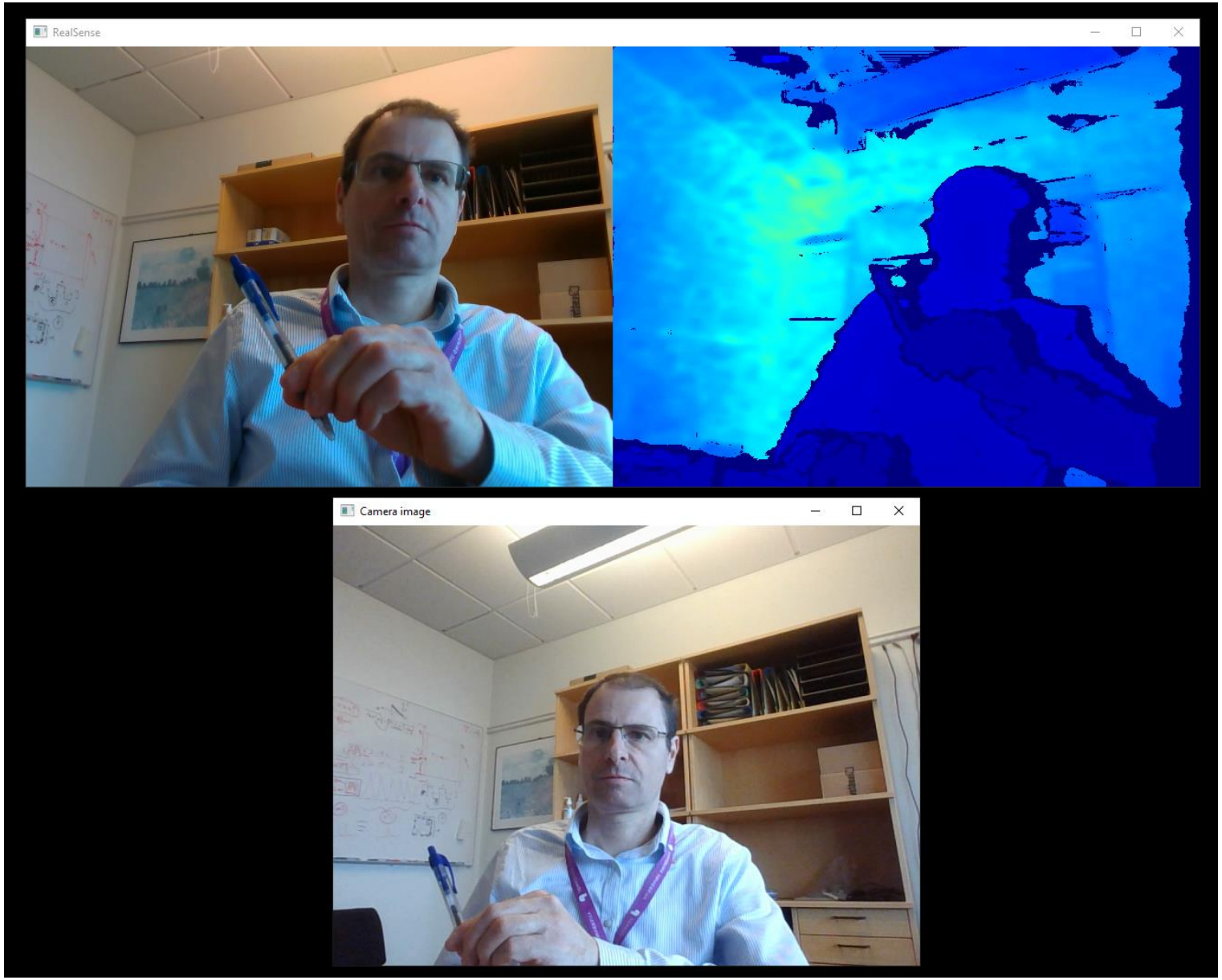
1) IMAGE SOURCES (INTEL REALSENSE)



1) IMAGE SOURCES (BOTH)

improc-0030.py > ...

```python
import pyrealsense2 as rs
import numpy as np
import cv2
# Configure depth and color streams
pipeline = rs.pipeline()
config = rs.config()
# Get device product line for setting a supporting resolution
pipeline_wrapper = rs.pipeline_wrapper(pipeline)
pipeline_profile = config.resolve(pipeline_wrapper)
device = pipeline_profile.get_device()
device_product_line = str(device.get_info(rs.camera_info.product_line))

found_rgb = False
for s in device.sensors:
    if s.get_info(rs.camera_info.name) == 'RGB Camera':
        found_rgb = True
        break
if not found_rgb:
    print("The demo requires Depth camera with Color sensor")
    exit(0)

config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
if device_product_line == 'L500':
    config.enable_stream(rs.stream.color, 960, 540, rs.format.bgr8, 30)
else:
    config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
# Start streaming
pipeline.start(config)

# Initialize Camera
cap = cv2.VideoCapture(0)
try:
    while True:
        # we read an image from the camera
        ret, frame = cap.read()
        # there is an image
        if ret:
            frame = cv2.flip(frame, flipCode = 1)
            # we display it
            cv2.imshow('Camera image', frame)
        # Wait for a coherent pair of frames: depth and color
        frames = pipeline.wait_for_frames()
        depth_frame = frames.get_depth_frame()
        color_frame = frames.get_color_frame()
        if not depth_frame or not color_frame:
            continue
        # Convert images to numpy arrays
        depth_image = np.asanyarray(depth_frame.get_data())
        depth_image = cv2.flip(depth_image, flipCode = 1)
        color_image = np.asanyarray(color_frame.get_data())
        color_image = cv2.flip(color_image, flipCode = 1)

        # Apply colormap on depth image (image must be converted to 8-bit per pixel first)
        depth_colormap = cv2.applyColorMap(cv2.convertScaleAbs(depth_image, alpha=0.03), cv2.COLORMAP_JET)

        depth_colormap_dim = depth_colormap.shape
        color_colormap_dim = color_image.shape

        # If depth and color resolutions are different, resize color image to match depth image for display
        if depth_colormap_dim != color_colormap_dim:
            resized_color_image = cv2.resize(color_image, dsize=(depth_colormap_dim[1],
                                                                  depth_colormap_dim[0]), interpolation=cv2.INTER_AREA)
            images = np.hstack((resized_color_image, depth_colormap))
        else:
            images = np.hstack((color_image, depth_colormap))
        # Show images
        cv2.namedWindow('RealSense', cv2.WINDOW_AUTOSIZE)
        cv2.imshow('RealSense', images)
        k = cv2.waitKey(10)
        if k==ord('q'):
            break
finally:
    # Stop streaming
    pipeline.stop()
    cv2.destroyAllWindows()

```


RealSense
Camera image

2) FIXED OR MOVING CAMERA

- The problem is not exactly the same with a fixed or moving camera
- **But** there is always **NOISE** on your images in both case
- Let's do an **experiment**

improc-0031.py > depth_colormap_dim

```
29  pipeline.start(config)
30
31  # Initialize Camera
32  cap = cv2.VideoCapture(0)
33  ret, frame = cap.read()
34  x = datetime.datetime.now()
35  fileRGB = 'outRGB'+str(x.year)+str(x.month)+str(x.day)+str(x.hour)+str(x.minute)+str(x.second)+'.avi'
36  outRGB = cv2.VideoWriter(fileRGB,cv2.VideoWriter_fourcc(*'MJPG'), 30, (frame.shape[1],frame.shape[0]))
37  numberOfImagesInVideo = 300
38  nb = 0
39
40  try:
41      while True:
42          # we read an image from the camera
43          ret, frame = cap.read()
44          # there is an image
45          if ret:
46              frame = cv2.flip(frame, flipCode = 1)
47              if nb<numberOfImagesInVideo:
48                  outRGB.write(frame)
49                  nb += 1
50              # we display it
51              cv2.imshow('Camera image', frame)
52          # Wait for a coherent pair of frames: depth and color
53          frames = pipeline.wait_for_frames()
54          depth_frame = frames.get_depth_frame()
```

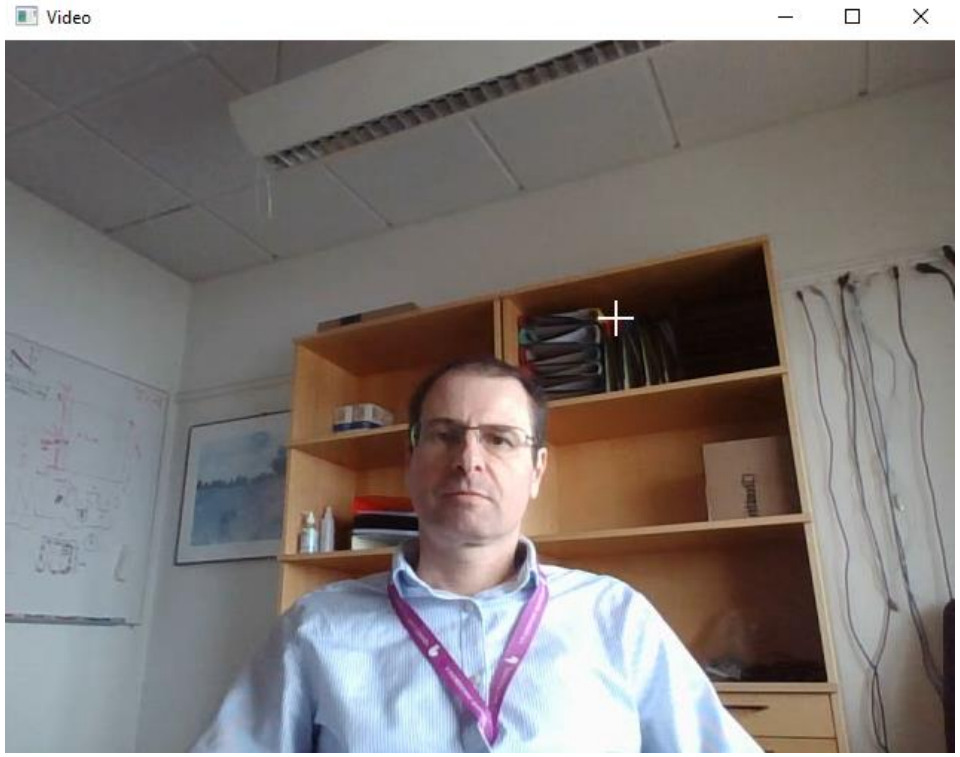
2) FIXED OR MOVING CAMERA

- We obtain a **video file**
- Let’s try to see if the image pixels are ”stable” in time!

2) FIXED OR MOVING CAMERA

improc-0032.py > ...

```python
import cv2
import numpy as np

#source: https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/
cap = cv2.VideoCapture('outRGB2023414131958.avi')

# Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, frame = cap.read()
  if ret == True:
    # Display the resulting frame
    cv2.imshow('Video', frame)
    # Press Q on keyboard to  exit
    k = cv2.waitKey(25)
    if k == ord('q'):
      break
  else:
    break

# When everything done, release the video capture object
cap.release()

# Closes all the frames
cv2.destroyAllWindows()
```



2) FIXED OR MOVING CAMERA

improc-0034.py > ...

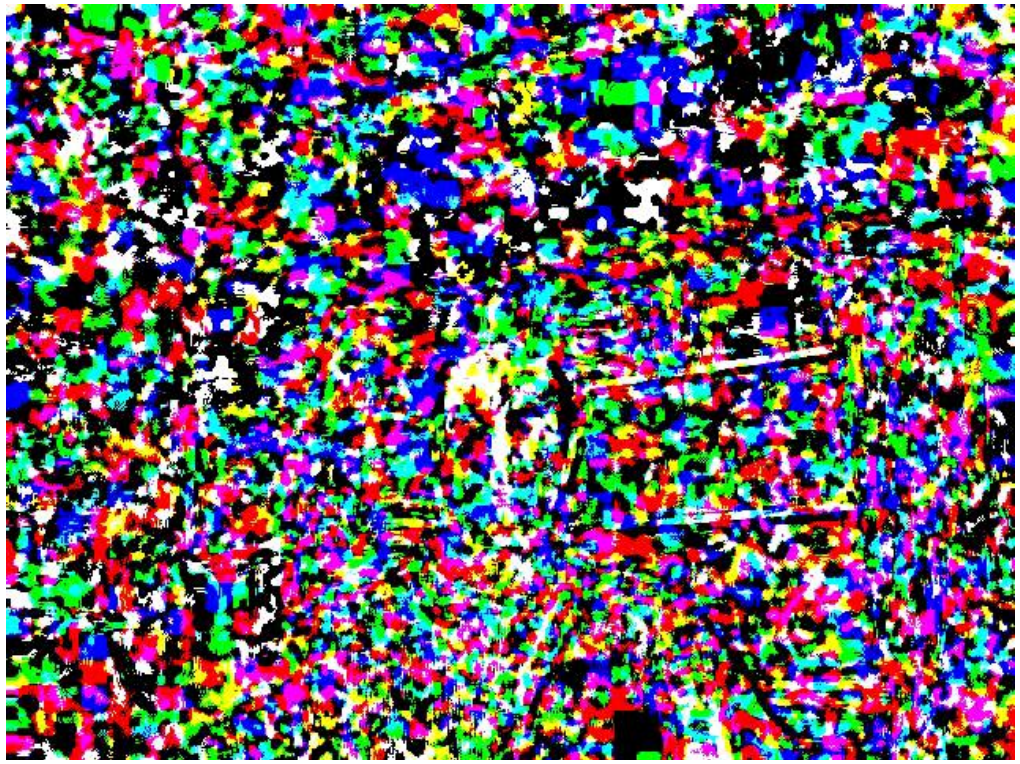
```python
import cv2
import datetime

#source: https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/
cap = cv2.VideoCapture('outRGB2023414131958.avi')

# Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")

x = datetime.datetime.now()
ret, frame = cap.read()
fileRGB = 'noiseRGB'+str(x.year)+str(x.month)+str(x.day)+str(x.hour)+str(x.minute)+str(x.second)+'.avi'
outRGB = cv2.VideoWriter(fileRGB,cv2.VideoWriter_fourcc(*'MJPG'), 30, (frame.shape[1],frame.shape[0]))

ret, oldFrame = cap.read()

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, newFrame = cap.read()
  if ret == True:
    # Display the resulting frame
    cv2.imshow('Video', newFrame-oldFrame)
    outRGB.write(newFrame-oldFrame)

    oldFrame = newFrame.copy()
    # Press Q on keyboard to  exit
    k = cv2.waitKey(25)
    if k == ord('q'):
      break
  else:
    break

# When everything done, release the video capture object
cap.release()
outRGB.release()

# Closes all the frames
cv2.destroyAllWindows()

```

2) FIXED OR MOVING CAMERA

improc-0035.py > ...

```python
import cv2
import datetime

#source: https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/
cap = cv2.VideoCapture('outRGB2023414131958.avi')

# Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")

x = datetime.datetime.now()
ret, frame = cap.read()
fileRGB = 'noiseRGB'+str(x.year)+str(x.month)+str(x.day)+str(x.hour)+str(x.minute)+str(x.second)+'.avi'
outRGB = cv2.VideoWriter(fileRGB,cv2.VideoWriter_fourcc(*'MJPG'), 30, (frame.shape[1],frame.shape[0]))

ret, oldFrame = cap.read()
oldFrame = cv2.blur(oldFrame, (5, 5))

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, newFrame = cap.read()

  if ret == True:
    # Display the resulting frame
    newFrame = cv2.blur(newFrame, (5, 5))
    cv2.imshow('Video', newFrame-oldFrame)
    outRGB.write(newFrame-oldFrame)

    oldFrame = newFrame.copy()
    # Press Q on keyboard to  exit
    k = cv2.waitKey(25)
    if k == ord('q'):
      break
  else:
    break

# When everything done, release the video capture object
cap.release()
outRGB.release()

# Closes all the frames
cv2.destroyAllWindows()
```

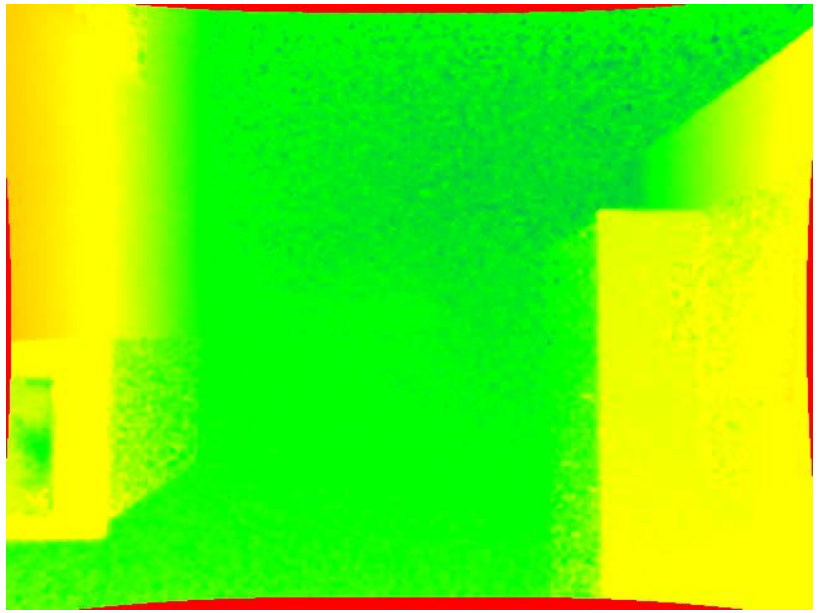
- GaussianBlur and medianBlur can also help…



3) TYPES OF IMAGES

- Depending on your camera sensors and needs, you can use many types of cameras:

DEPTH



INFRARED

RGB

LIDAR, MEDICAL, MULTISPECTRAL, AERIAL, SATELITE IMAGES…

4) COLOR IMAGES OR NOT…

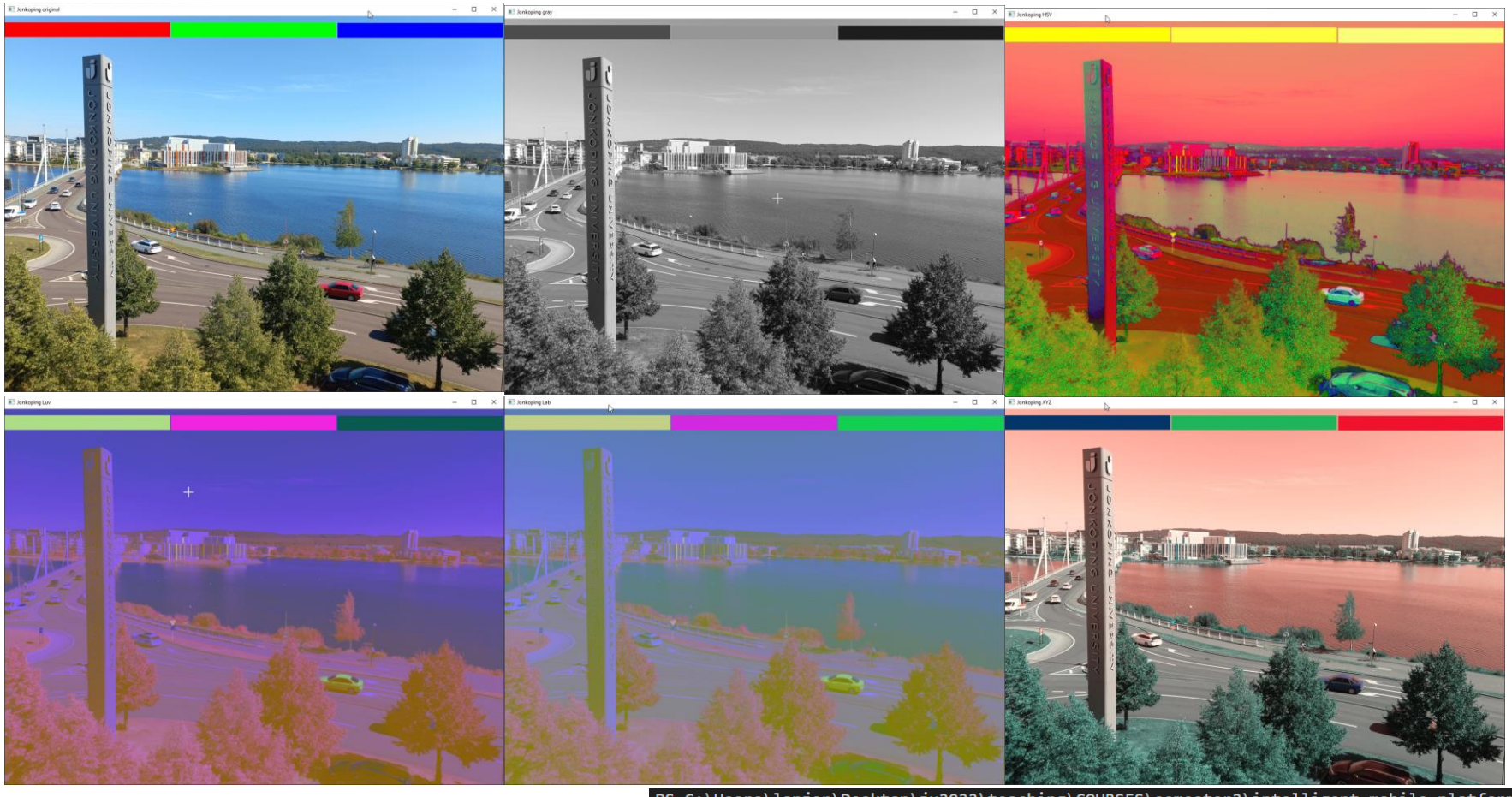
- RGB, Grayscale, HSV, HSL, XYZ, Lab, Luv… colorspaces

improc-0040.py > ...

```python
# we import all libraries used in our program
import cv2
import numpy

# we read an image from disk to memory
img = cv2.imread('img/jonkoping-01.jpg')
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
imgLab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
imgLuv = cv2.cvtColor(img, cv2.COLOR_BGR2LUV)
imgXYZ = cv2.cvtColor(img, cv2.COLOR_BGR2XYZ)
print("---> Stats:")
print("shape, (min, max, avg) RGB: ", img.shape, ', ', numpy.min(img), numpy.max(img), numpy.mean(img))
print("shape, (min, max, avg) Gray: ", imgGray.shape, ', ', numpy.min(imgGray), numpy.max(imgGray), numpy.mean(imgGray))
print("shape, (min, max, avg) HSV: ", imgHSV.shape, ', ', numpy.min(imgHSV), numpy.max(imgHSV), numpy.mean(imgHSV))
print("shape, (min, max, avg) Lab: ", imgLab.shape, ', ', numpy.min(imgLab), numpy.max(imgLab), numpy.mean(imgLab))
print("shape, (min, max, avg) Luv: ", imgLuv.shape, ', ', numpy.min(imgLuv), numpy.max(imgLuv), numpy.mean(imgLuv))
print("shape, (min, max, avg) XYZ: ", imgXYZ.shape, ', ', numpy.min(imgXYZ), numpy.max(imgXYZ), numpy.mean(imgXYZ))
# we show the image and wait for a key to be pressed
cv2.imshow('Jonkoping original', img)
cv2.imshow('Jonkoping gray', imgGray)
cv2.imshow('Jonkoping Lab', imgLab)
cv2.imshow('Jonkoping HSV', imgHSV)
cv2.imshow('Jonkoping Luv', imgLuv)
cv2.imshow('Jonkoping XYZ', imgXYZ)
key = cv2.waitKey(0)

cv2.destroyAllWindows()

```

OpenCV docs: https://docs.opencv.org/4.7.0/d8/d01/group__imgproc__color__conversions.html

4) COLOR IMAGES OR NOT…



```
PS C:\Users\lanjer\Desktop\ju2022\teaching\COURSES\semester2\intelligent-mobile-platform\cours-jerome\code> python .\improc-0040.py
---> Stats:
shape, (min, max, avg) RGB:  (867, 1156, 3) ,  0 255 133.69111128405498
shape, (min, max, avg) Gray:  (867, 1156) ,  4 255 132.29723762087778
shape, (min, max, avg) HSV:  (867, 1156, 3) ,  0 255 114.5795405413675
shape, (min, max, avg) Lab:  (867, 1156, 3) ,  2 255 130.53138565284314
shape, (min, max, avg) Luv:  (867, 1156, 3) ,  3 255 119.25149164082487
shape, (min, max, avg) XYZ:  (867, 1156, 3) ,  1 255 137.87846236941076
```

5) BACKGROUND REMOVAL

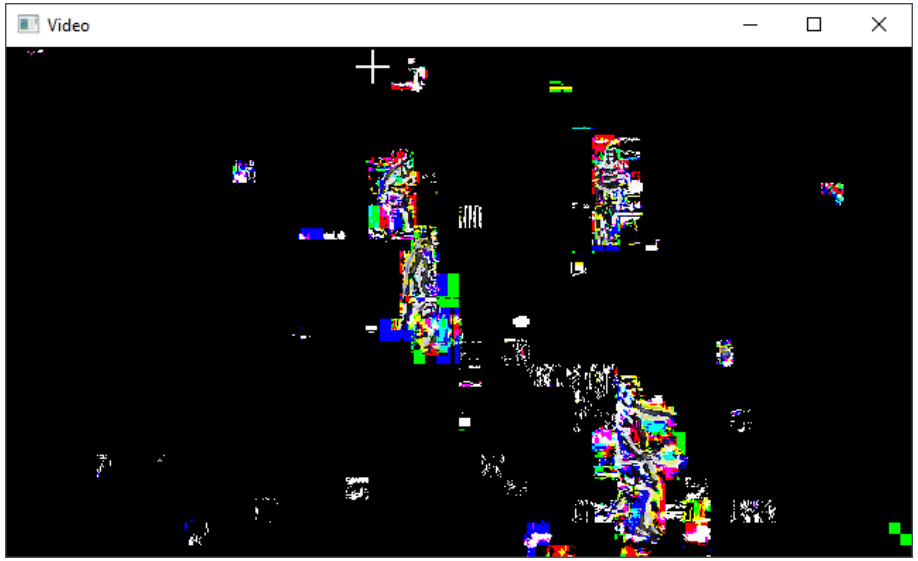
- In order to study the moving parts in a video (here th players and the ball), we can try to guess the background of the image and see what have moved by a simple difference (bgImg-CurrentImg)
- How to get the background of a video?

Source video: https://www.youtube.com/watch?v=P6yfa7rfc6o
YT2MP4 Converter: https://wave.video/convert/youtube-to-mp4-85

5) BACKGROUND REMOVAL

improc-0050.py > ...

```python
import cv2
import numpy as np

#source: https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/
cap = cv2.VideoCapture('padelpoint.mp4')
# Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")
ret, oldFrame = cap.read()
# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, newFrame = cap.read()
  if ret == True:
    # Display the resulting frame
    cv2.imshow('Video', oldFrame-newFrame)
    oldFrame = newFrame.copy()
    # Press Q on keyboard to  exit
    k = cv2.waitKey(25)
    if k == ord('q'):
      break
  else:
    break

# When everything done, release the video capture object
cap.release()
# Closes all the frames
cv2.destroyAllWindows()

```



5) BACKGROUND REMOVAL (1ST TRY)

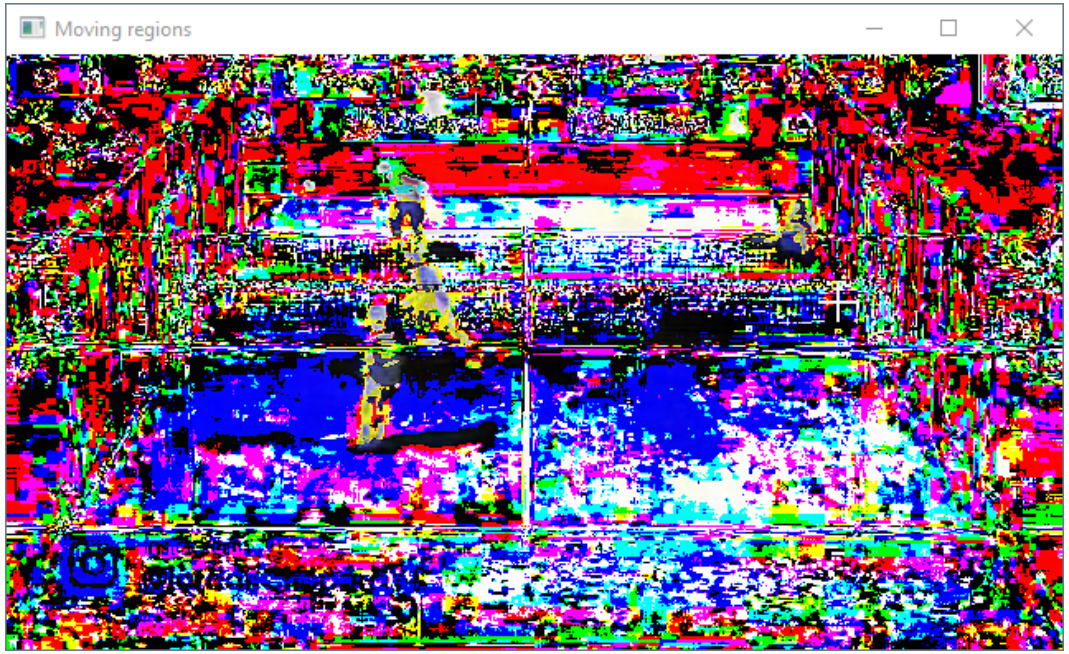
improc-0051.py > ...

```python
import cv2
import numpy

#source: https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/
cap = cv2.VideoCapture('padelpoint.mp4')
# Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")
ret, frame = cap.read()
sumImages = numpy.zeros(frame.shape)
nb = 0

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, frame = cap.read()
  if ret == True:
    # add the image to the sum
    sumImages += frame
    nb += 1
  else:
    break

avgImages = sumImages / nb
avgImages = avgImages.astype(numpy.uint8)
cv2.imshow('sumImages', avgImages)
key = cv2.waitKey(0)

# When everything done, release the video capture object
cap.release()
# Closes all the frames
cv2.destroyAllWindows()
```

5) BACKGROUND REMOVAL (2ND TRY)

improc-0052.py > ...

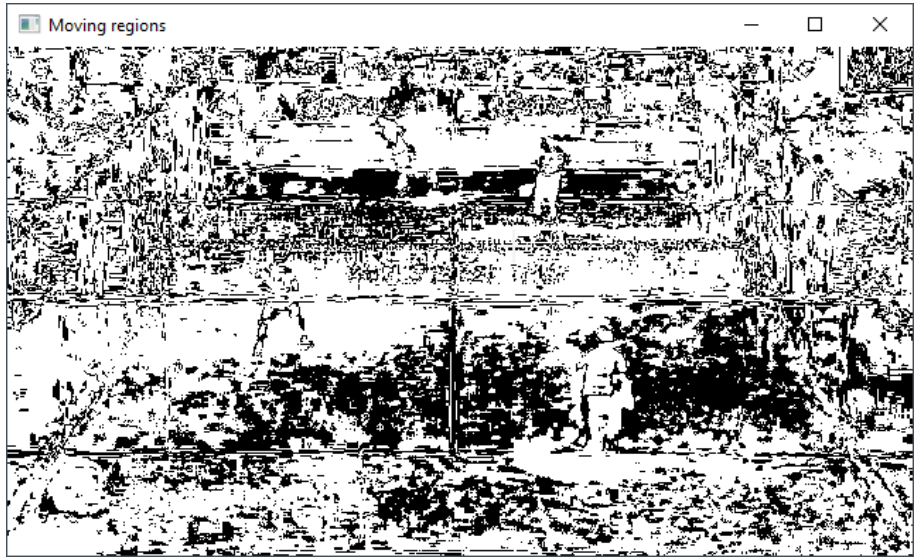
```
        # add the image to the sum
        sumImages += frame
        nb += 1
    else:
        break

avgImages = sumImages / nb
avgImages = avgImages.astype(numpy.uint8)
cv2.imshow('sumImages', avgImages)
key = cv2.waitKey(0)

backgroundImage = avgImages.copy()
# When everything done, release the video capture object
cap.release()

cap = cv2.VideoCapture('padelpoint.mp4')
# Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")
ret, frame = cap.read()
sumImages = numpy.zeros(frame.shape)
nb = 0

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, frame = cap.read()
  cv2.imshow('Moving regions', backgroundImage-frame)
  key = cv2.waitKey(10)
  if key == ord('q'):
    break

# When everything done, release the video capture object
cap.release()
# Closes all the frames
cv2.destroyAllWindows()
```



5) BACKGROUND REMOVAL (3RD TRY)

improc-0053.py > ...

```
      sumImages += cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
      nb += 1
    else:
      break

avgImages = sumImages / nb
avgImages = avgImages.astype(numpy.uint8)
cv2.imshow('sumImages', avgImages)
key = cv2.waitKey(0)

backgroundImage = avgImages.copy()
# When everything done, release the video capture object
cap.release()

cap = cv2.VideoCapture('padelpoint.mp4')
# Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")
ret, frame = cap.read()
sumImages = numpy.zeros(frame.shape)
nb = 0

# Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, frame = cap.read()
  diff = (((backgroundImage-cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY))<200)*255).astype(numpy.uint8)
  print(diff)
  cv2.imshow('Moving regions', diff)
  key = cv2.waitKey(10)
  if key == ord('q'):
    break

# When everything done, release the video capture object
cap.release()
# Closes all the frames
cv2.destroyAllWindows()
```



5) BACKGROUND REMOVAL

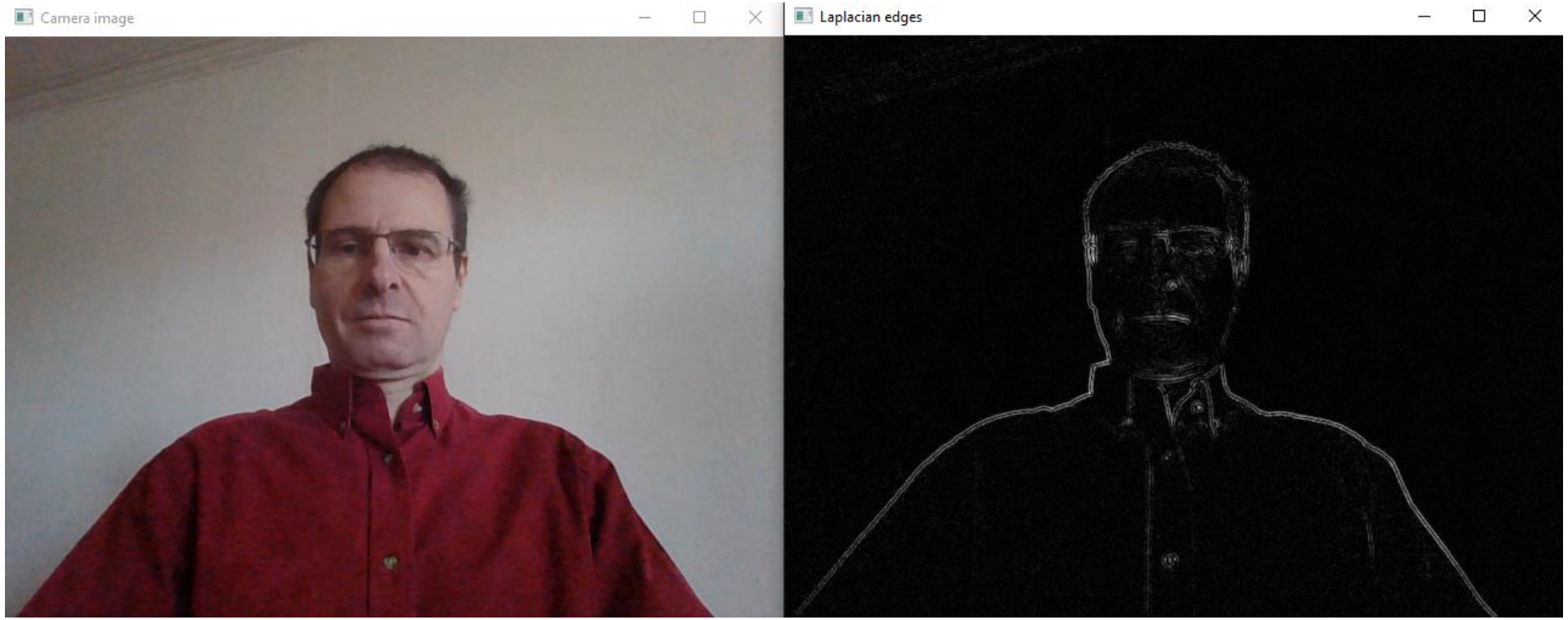
5) BACKGROUND REMOVAL

improc-0054.py > ...

```
23      break
24
25  avgImages = sumImages / nb
26  avgImages = avgImages.astype(numpy.uint8)
27  cv2.imshow('sumImages', avgImages)
28  key = cv2.waitKey(0)
29
30  backgroundImage = avgImages.copy()
31  # When everything done, release the video capture object
32  cap.release()
33
34  cap = cv2.VideoCapture('bridgeRGB2023414161111.avi')
35  # Check if camera opened successfully
36  if (cap.isOpened()== False):
37    print("Error opening video stream or file")
38  ret, frame = cap.read()
39  sumImages = numpy.zeros(frame.shape)
40  nb = 0
41
42  # Read until video is completed
43  while(cap.isOpened()):
44    # Capture frame-by-frame
45    ret, frame = cap.read()
46    diff = (((backgroundImage-cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY))<200)*255).astype(numpy.uint8)
47    print(diff)
48    cv2.imshow('Moving regions', diff)
49    key = cv2.waitKey(10)
50    if key == ord('q'):
51      break
52
53  # When everything done, release the video capture object
54  cap.release()
55  # Closes all the frames
56  cv2.destroyAllWindows()
57
```

6) EDGES AND REGIONS

- In many applications of Image Processing, you need to find **edges** and **regions** in an image.



6) EDGES AND REGIONS

improc-0060.py > ...

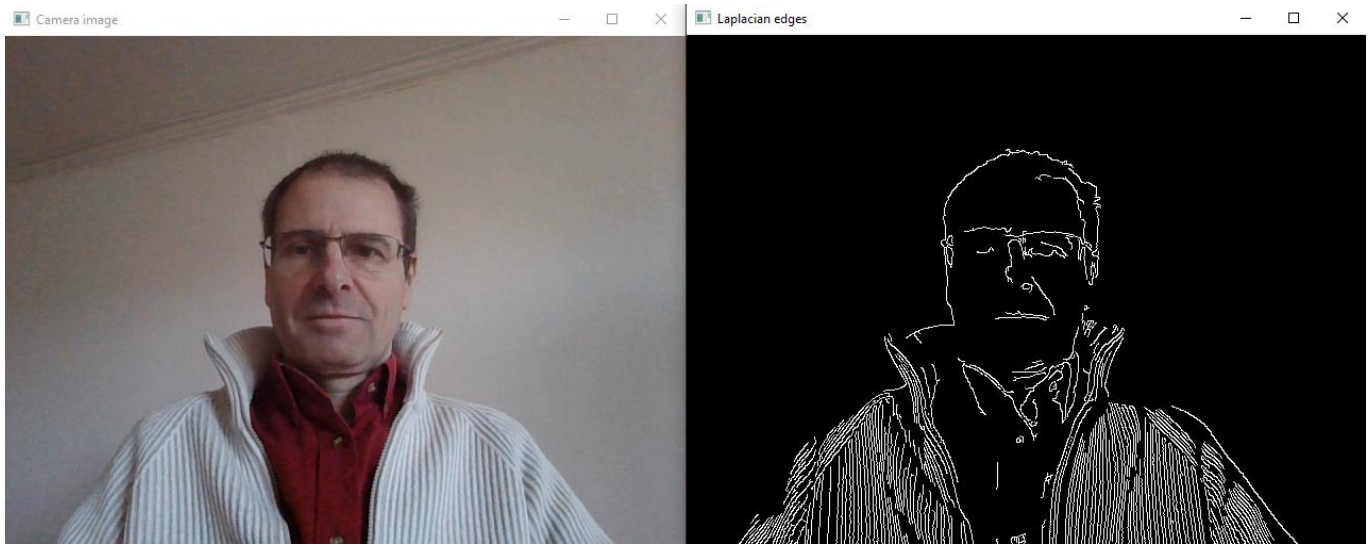
```python
# we import all libraries used in our program
import cv2

# Initialize Camera
cap = cv2.VideoCapture(0)

# infinite loop
while True:
    # we read an image from the camera
    ret, frame = cap.read()
    # there is an image
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        frameBlur = cv2.GaussianBlur(frame, (3, 3), 0)
        frameGray = cv2.cvtColor(frameBlur, cv2.COLOR_BGR2GRAY)
        frameEdges = cv2.Laplacian(frameGray, cv2.CV_16S, ksize=3)
        frameFinalEdges = cv2.convertScaleAbs(frameEdges)
        # we display it
        cv2.imshow('Camera image', frame)
        cv2.imshow('Laplacian edges', frameFinalEdges)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

# we close all windows
cv2.destroyAllWindows()
```

Laplacian filter

6) EDGES AND REGIONS

improc-0061.py > ...

```python
# we import all libraries used in our program
import cv2

# Initialize Camera
cap = cv2.VideoCapture(0)

# infinite loop
while True:
    # we read an image from the camera
    ret, frame = cap.read()
    # there is an image
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        frameEdges = cv2.Canny(frame, 100, 200)
        # we display it
        cv2.imshow('Camera image', frame)
        cv2.imshow('Laplacian edges', frameEdges)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

# we close all windows
cv2.destroyAllWindows()
```

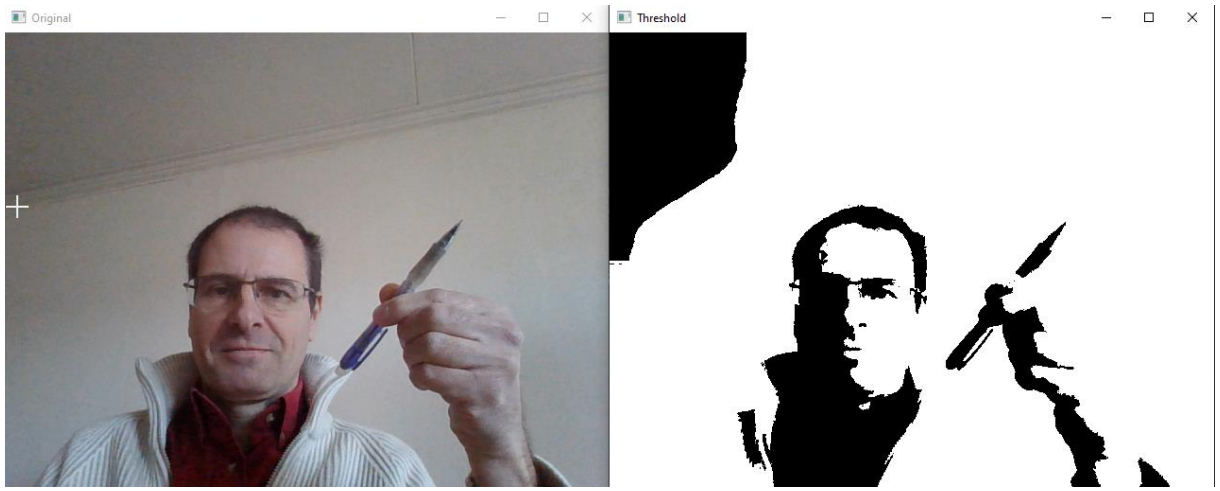


Canny Edge Detector

6) EDGES AND REGIONS

improc-0062.py > ...

```python
# we import all libraries used in our program
import cv2
# Initialize Camera
cap = cv2.VideoCapture(0)
# infinite loop
while True:
    # we read an image from the camera
    ret, frame = cap.read()
    # there is an image
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        #Edge-based segmentation
        #source: https://pyimagesearch.com/2015/11/02/watershed-opencv/
        shifted = cv2.pyrMeanShiftFiltering(frame, 21, 51)
        # convert the mean shift image to grayscale, then apply
        # Otsu's thresholding
        gray = cv2.cvtColor(shifted, cv2.COLOR_BGR2GRAY)
        thresh = cv2.threshold(gray, 0, 255,
            cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
        # we display it
        cv2.imshow("Original", frame)
        cv2.imshow("Threshold", thresh)

        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break

# we close all windows
cv2.destroyAllWindows()

```



Watershed algorithm

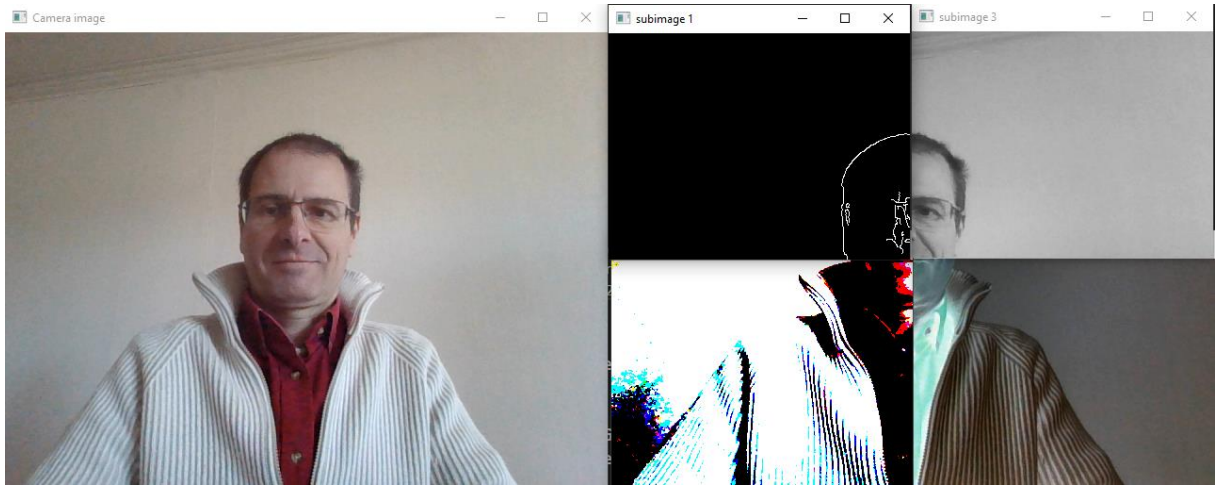
7) REGION OF INTEREST

- When working on an image, we can choose to work only in a sub-image:
  - The interest thing happens only in a small part of the image so it is unnecessary to work on the whole image
  - It saves computing time to work only on a small part of the image

7) REGION OF INTEREST

improc-0070.py > ...

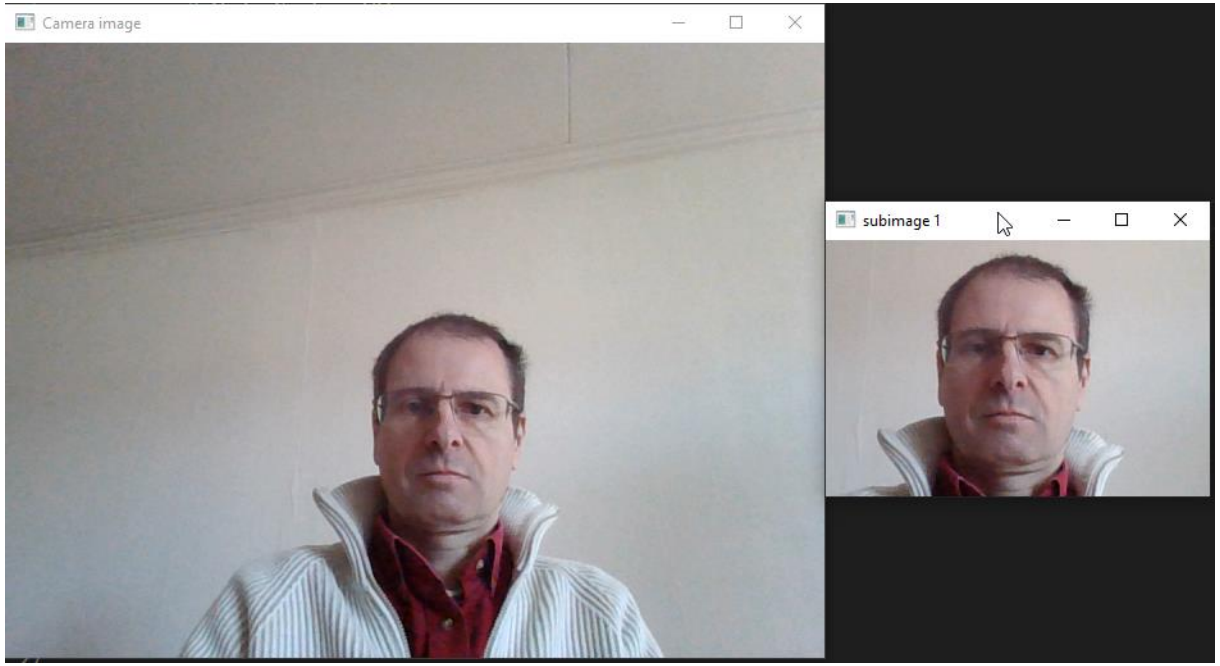
```python
# we import all libraries used in our program
import cv2
import numpy
# Initialize Camera
cap = cv2.VideoCapture(0)
ret, frame = cap.read()
print("Shape: ", frame.shape)
nb = 0
# infinite loop
while True:
    ret, frame = cap.read()
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        frame1 = frame[0:frame.shape[0]//2, 0:frame.shape[1]//2]
        frame2 = frame[frame.shape[0]//2:frame.shape[0], 0:frame.shape[1]//2]
        frame3 = frame[0:frame.shape[0]//2, frame.shape[1]//2:frame.shape[1]]
        frame4 = frame[frame.shape[0]//2:frame.shape[0], frame.shape[1]//2:frame.shape[1]]
        if nb==0:
            print("shapes 1,2,3,4: ", frame1.shape, frame2.shape, frame3.shape, frame4.shape)
        nb += 1
        frame1 = cv2.Canny(frame1, 100, 200)
        frame2 = ((frame2>127)*255).astype(numpy.uint8)
        frame3 = cv2.cvtColor(frame3, cv2.COLOR_BGR2GRAY)
        frame4 = 255 - frame4
        # we display it
        cv2.imshow('Camera image', frame)
        cv2.imshow('subimage 1', frame1)
        cv2.imshow('subimage 2', frame2)
        cv2.imshow('subimage 3', frame3)
        cv2.imshow('subimage 4', frame4)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

```



7) REGION OF INTEREST

improc-0071.py > ...

```python
# we import all libraries used in our program
import cv2
import numpy
# Initialize Camera
cap = cv2.VideoCapture(0)
ret, frame = cap.read()
# infinite loop
while True:
    ret, frame = cap.read()
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        frame1 = frame[200:400, 200:500]
        # we display it
        cv2.imshow('Camera image', frame)
        cv2.imshow('subimage 1', frame1)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

# we close all windows
cv2.destroyAllWindows()
```



8) EXTRACT INFORMATION

- You can extract statistics from images to get information about the whole image and/or regions, …
- Among all possible tools, the histogram of the image gives information about the distribution of colors in the image

8) EXTRACT INFORMATION

improc-0080.py > ...

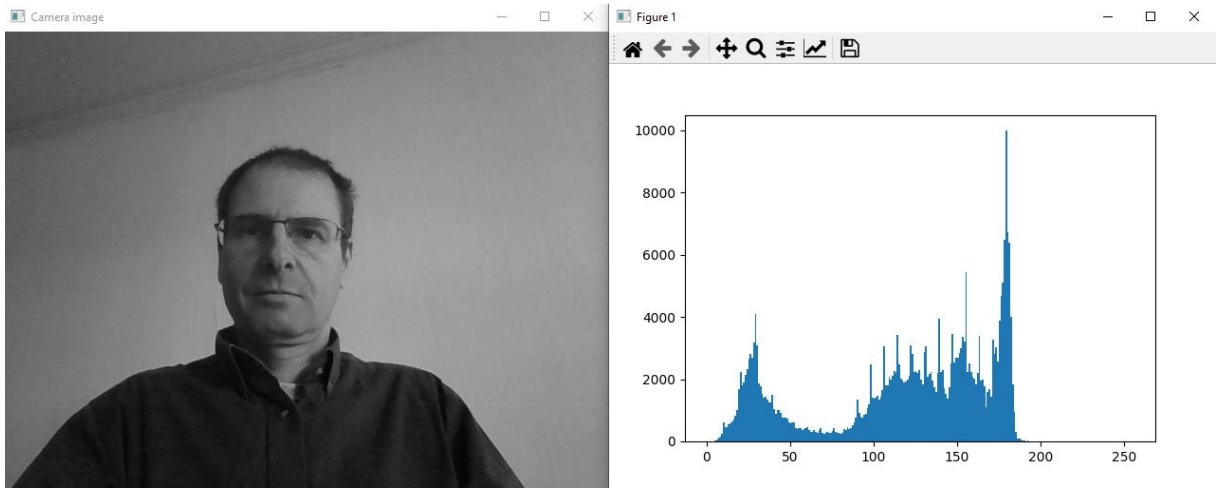
```python
# we import all libraries used in our program
import cv2
import numpy
# Initialize Camera
cap = cv2.VideoCapture(0)
ret, frame = cap.read()
# infinite loop
while True:
    ret, frame = cap.read()
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        mean = numpy.mean(frame)
        min = numpy.min(frame)
        max = numpy.max(frame)
        stddev = numpy.std(frame)
        print('Statistics: min, max, mean, stddev ---> ', min, max, mean, stddev)
        # we display it
        cv2.imshow('Camera image', frame)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

# we close all windows
cv2.destroyAllWindows()
```

```
Statistics: min, max, mean, stddev --->  0 221 121.81957139756945 33.391712692708815
Statistics: min, max, mean, stddev --->  0 217 121.92159505208333 33.3514742402933
Statistics: min, max, mean, stddev --->  0 221 121.96024848090278 33.34492241090537
Statistics: min, max, mean, stddev --->  0 217 121.95812825520834 33.36773861152294
Statistics: min, max, mean, stddev --->  0 223 121.9390603298611 33.327133106727736
```

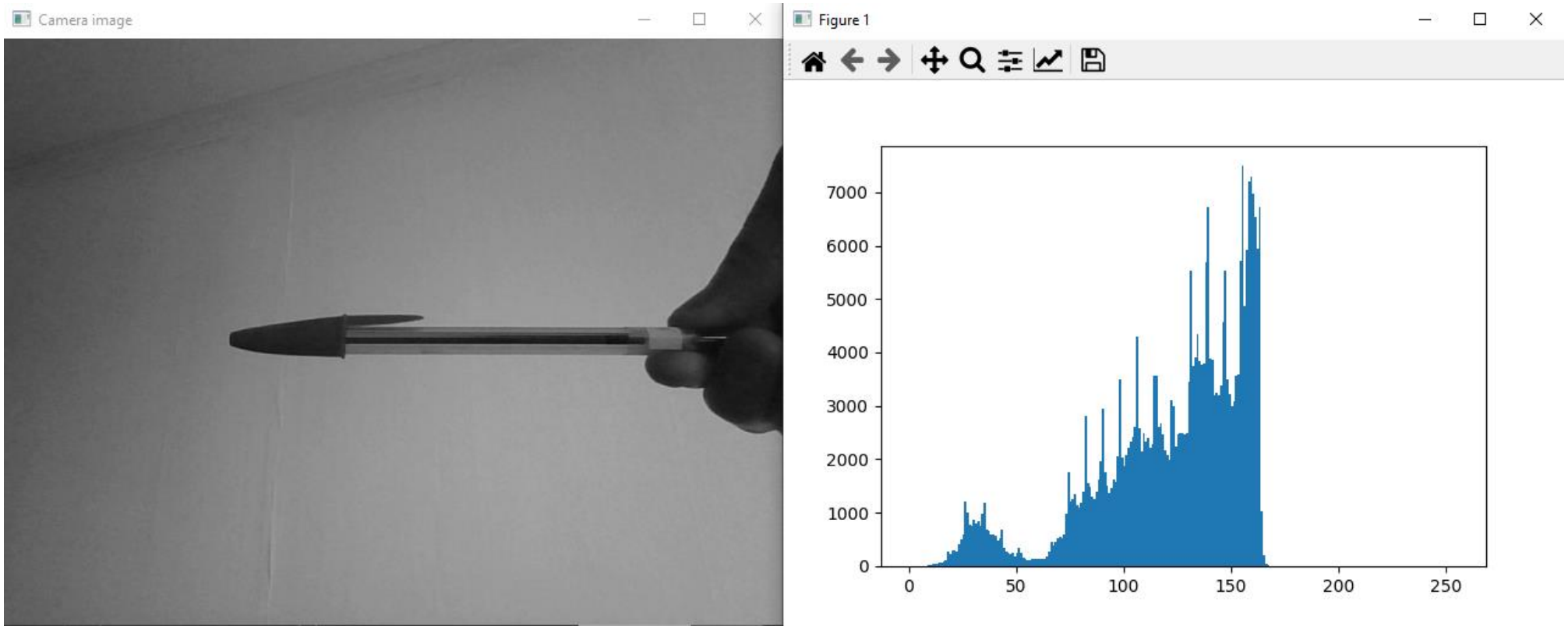
8) EXTRACT INFORMATION

improc-0081.py > ...

```python
# we import all libraries used in our program
import cv2
import numpy
from matplotlib import pyplot as plt
# Initialize Camera
cap = cv2.VideoCapture(0)
ret, frame = cap.read()
# infinite loop
while True:
    ret, frame = cap.read()
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        plt.hist(frame.ravel(),256,[0,256])
        plt.show()
        # we display it
        cv2.imshow('Camera image', frame)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

# we close all windows
cv2.destroyAllWindows()
```



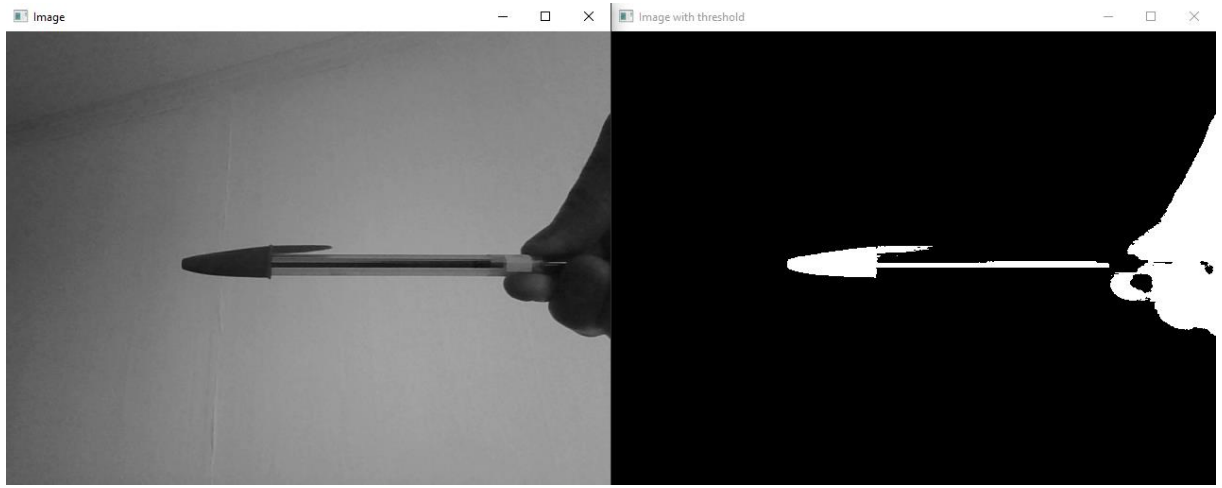
8) EXTRACT INFORMATION



8) EXTRACT INFORMATION

improc-0082.py > ...

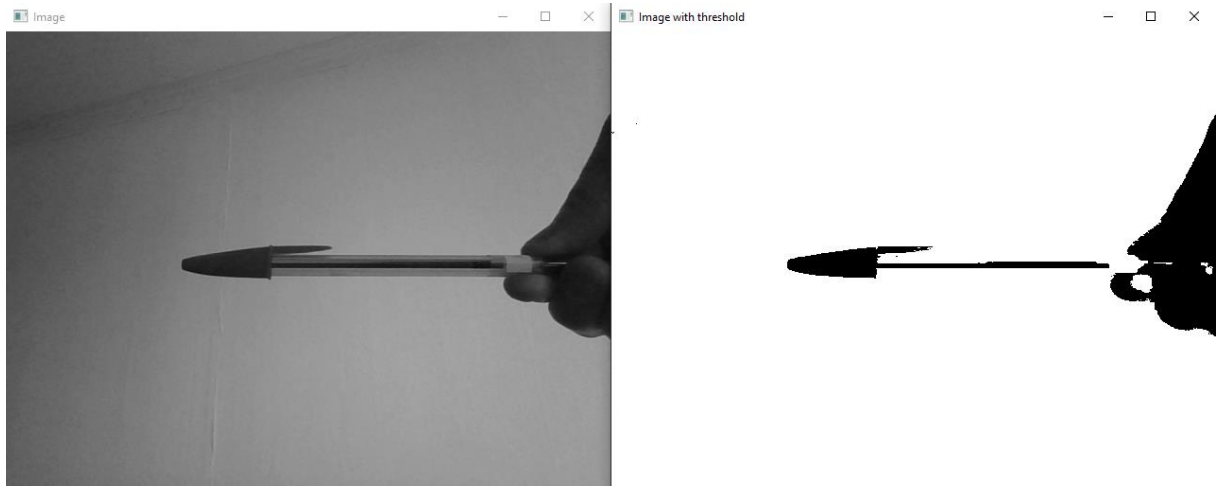
```
5    # we load the image
6    img = cv2.imread('penna.bmp')
7    print('Shape: ', img.shape)
8    # infinite loop
9    frame = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
10   plt.hist(frame.ravel(),256,[0,256])
11   plt.show()
12
13   frameThreshold = ((frame<56).astype(numpy.uint8))*255
14   print(frameThreshold)
15   # we display it
16   cv2.imshow('Image', frame)
17   cv2.imshow('Image with threshold', frameThreshold)
18
19   # we wait for a key pressed
20   k = cv2.waitKey(0)
21
22   # we close all windows
23   cv2.destroyAllWindows()
24
```



8) EXTRACT INFORMATION

improc-0083.py > ...

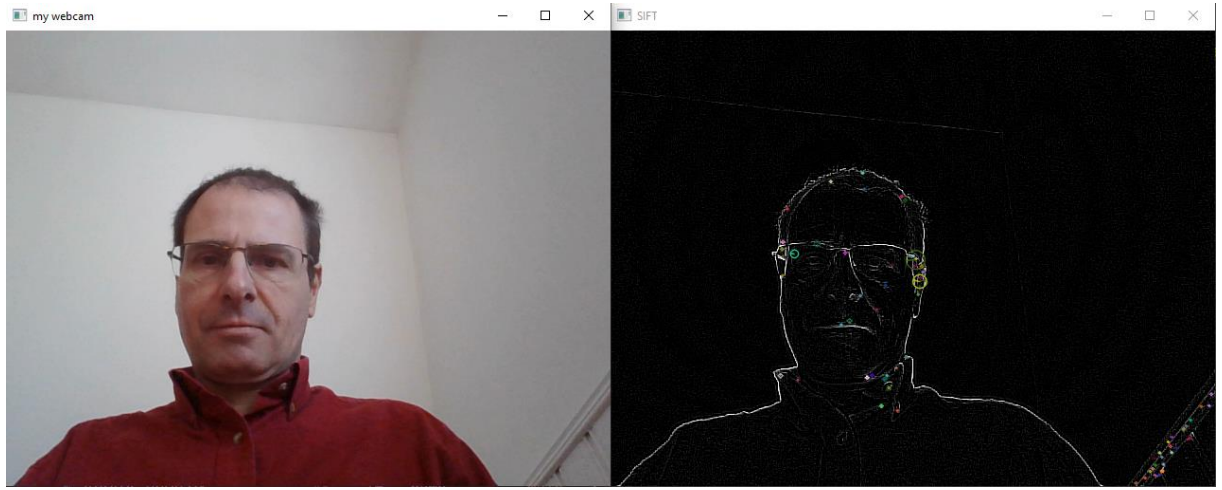
```python
# we import all libraries used in our program
import cv2
import numpy
from matplotlib import pyplot as plt
# we load the image
img = cv2.imread('penna.bmp')
print('Shape: ', img.shape)
# infinite loop
frame = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.hist(frame.ravel(),256,[0,256])
plt.show()

frameThreshold = ((frame>56).astype(numpy.uint8))*255
print(frameThreshold)
# we display it
cv2.imshow('Image', frame)
cv2.imshow('Image with threshold', frameThreshold)

# we wait for a key pressed
k = cv2.waitKey(0)

# we close all windows
cv2.destroyAllWindows()

```



8) EXTRACT INFORMATION

improc-0110.py > show_webcam

```python
import cv2
import numpy
def show_webcam(mirror=False):
    ddepth = cv2.CV_8U
    kernel_size = 3
    cam = cv2.VideoCapture(0)
    while True:
        ret_val, img = cam.read()
        imgSift = img.copy()
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        grayLaplace = cv2.Laplacian(gray, ddepth, ksize=kernel_size)

        # Applying SIFT detector
        sift = cv2.SIFT_create()
        kp, descriptor = sift.detectAndCompute(grayLaplace, None)

        # Marking the keypoint on the image using circles
        imgSift=cv2.drawKeypoints(grayLaplace,
                                  kp ,
                                  imgSift,
                                  flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
        if mirror:
            img = cv2.flip(img, 1)
            imgSift = cv2.flip(imgSift, 1)
        cv2.imshow('my webcam', img)
        cv2.imshow('SIFT', imgSift)
        if cv2.waitKey(1) == 27:
            break  # esc to quit
    cv2.destroyAllWindows()
def main():
    show_webcam(mirror=True)
if __name__ == '__main__':
    main()
```



8) EXTRACT INFORMATION

improc-0111.py > ...

```python
import cv2
def show_webcam(mirror=False):
    cam = cv2.VideoCapture(0)
    while True:
        ret_val, img = cam.read()
        imgSift = img.copy()
        gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        # Applying SIFT detector
        sift = cv2.SIFT_create()
        kp = sift.detect(gray, None)
        # Marking the keypoint on the image using circles
        imgSift=cv2.drawKeypoints(gray,
                                  kp,
                                  imgSift,
                                  flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
        if mirror:
            img = cv2.flip(img, 1)
            imgSift = cv2.flip(imgSift, 1)
        cv2.imshow('my webcam', img)
        cv2.imshow('SIFT', imgSift)
        if cv2.waitKey(1) == 27:
            break  # esc to quit
    cv2.destroyAllWindows()
def main():
    show_webcam(mirror=True)
if __name__ == '__main__':
    main()

```

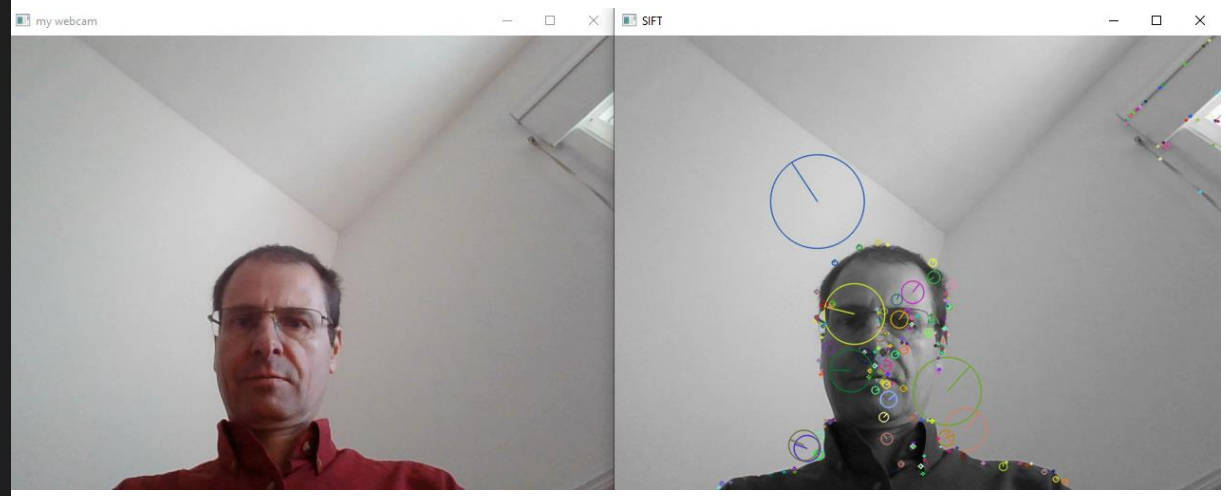


Image registration:
https://learnopencv.com/image-alignment-feature-based-using-opencv-c-python/

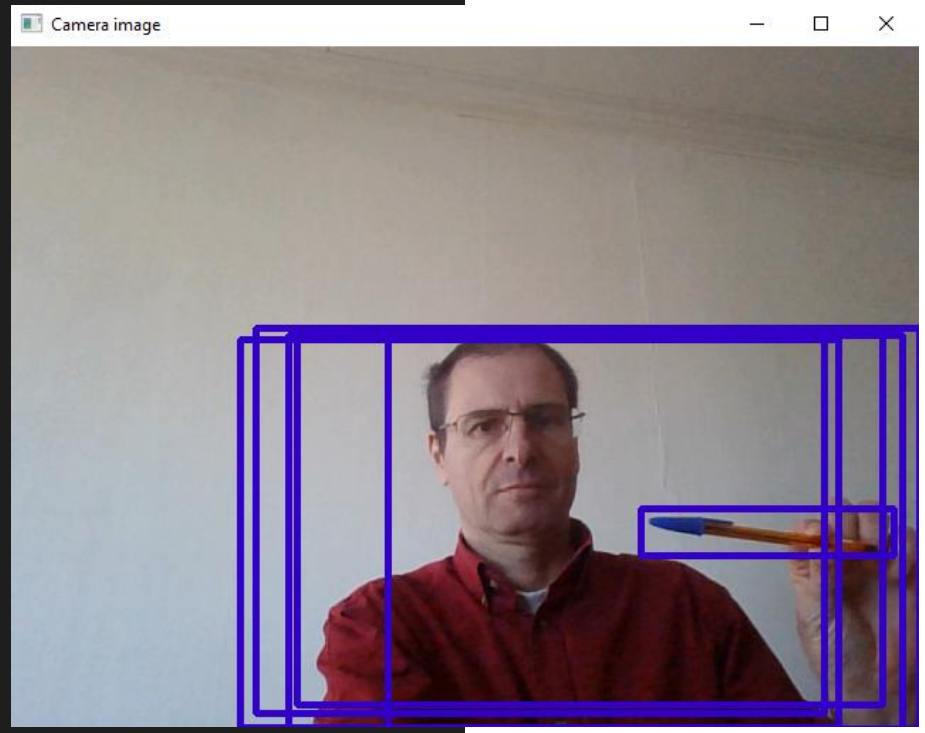
9) MACHINE LEARNING

- In order to find objects in images, you can use pre-trained machine learning algorithms (or train them yourself)
- To use these tools, you must have a powerful computer (GPU processing)
- In order to boost performances, you must program in C++ instead of Python

9) MACHINE LEARNING

improc-0099-model.py > ...

```python
import cv2
# OpenCV DNN
#source: https://github.com/patrick013/Object-Detection---Yolov3/blob/master/model/yolov3.weights
#another source: https://learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/
net = cv2.dnn.readNet('model/yolov3.weights', 'model/yolov3.cfg')
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA_FP16)
model = cv2.dnn_DetectionModel(net)
model.setInputParams(size=(320, 320), scale=1/255, swapRB=True)
# Initialize Camera
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if ret:
        (class_ids, scores, bboxes) = model.detect(frame)
        for class_id, score, bbox in zip(class_ids, scores, bboxes):
            (x, y, w, h) = bbox
            cv2.rectangle(frame, (x, y), (x+w, y+h), (200, 0, 50), 3)
        cv2.imshow('Camera image', frame)
        k = cv2.waitKey(100)
        if k==ord('q'):
            break
    else:
        print('ERROR ---> Camera not found!')

cv2.destroyAllWindows()
```



10) USEFUL TOOLS

- You can get important information about your OpenCV version
- You can also use keys and mouse events onto your images

10) USEFUL TOOLS

improc-0100.py > ...

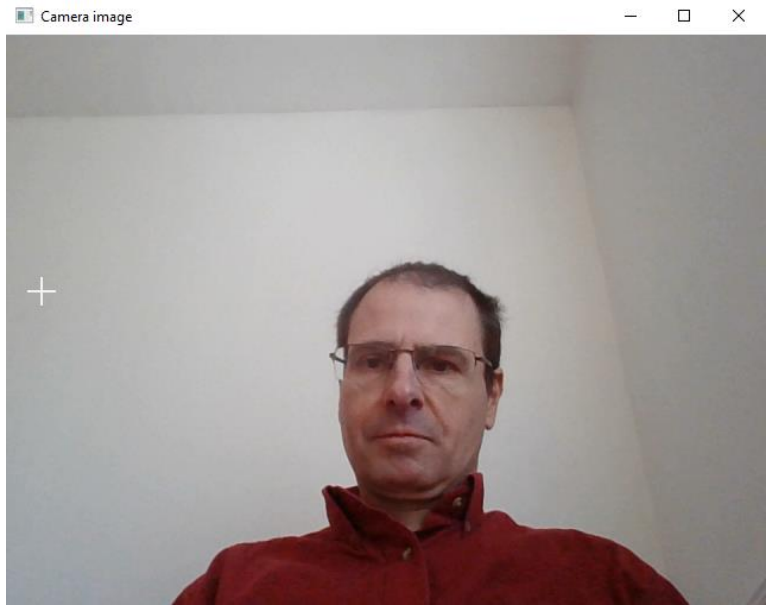
```python
# we import all libraries used in our program
import cv2
import datetime
print("---> OpenCV version : {0}".format(cv2.__version__))
# Initialize Camera
cap = cv2.VideoCapture(0)
# infinite loop
while True:
    # we read an image from the camera
    ret, frame = cap.read()
    # there is an image
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        # we display it
        cv2.imshow('Camera image', frame)
        # we wait for a key pressed
        k = cv2.waitKey(10)
        # if it is 'q', we quit
        if k == ord('q'):
            break
        if k == ord('s'):
            x = datetime.datetime.now()
            filenameRGB = 'outRGB'+str(x.year)+str(x.month)+
            str(x.day)+str(x.hour)+str(x.minute)+str(x.second)+'.png'
            cv2.imwrite(filenameRGB, frame)
            print('Image ', filenameRGB, ' saved...')
    # there is no image
    else:
        print('ERROR ---> Camera not found!')

# we close all windows
cv2.destroyAllWindows()
```

```
PS C:\Users\lanjer\Desktop\ju2022\teaching\COURSES\semester2\intelligent-mobile-platform\cours-jerome\code> python .\improc
-0100.py
---> OpenCV version : 4.7.0
Image  outRGB202341714158.png  saved...
Image  outRGB20234171420.png  saved...
Image  outRGB20234171422.png  saved...
PS C:\Users\lanjer\Desktop\ju2022\teaching\COURSES\semester2\intelligent-mobile-platform\cours-jerome\code>
```

10) USEFUL TOOLS

improc-0101.py > ...

```python
import cv2
import datetime
print("---> OpenCV version : {0}".format(cv2.__version__))
#source: https://thinkinfi.com/detect-mouse-click-events-with-python-opencv/
listPoints = []
def mousePoints(event,x,y,flags,params):
    global listPoints
    # Left button mouse click event opencv
    if event == cv2.EVENT_LBUTTONDOWN:
        listPoints.append([x, y])
        print(listPoints)
# Initialize Camera
cap = cv2.VideoCapture(0)
# infinite loop
while True:
    ret, frame = cap.read()
    if ret:
        frame = cv2.flip(frame, flipCode = 1)
        cv2.imshow('Camera image', frame)
        cv2.setMouseCallback("Camera image", mousePoints)
        k = cv2.waitKey(10)
        if k == ord('q'):
            break
        if k == ord('s'):
            x = datetime.datetime.now()
            filenameRGB = 'outRGB'+str(x.year)+str(x.month)+str(x.day)+str(x.hour)+str(x.minute)+str(x.second)+'.png'
            cv2.imwrite(filenameRGB, frame)
            print('Image ', filenameRGB, ' saved...')
    # there is no image
    else:
        print('ERROR ---> Camera not found!')
# we close all windows
cv2.destroyAllWindows()
```




```
---> OpenCV version : 4.7.0
[[217, 160]]
[[217, 160], [476, 229]]
[[217, 160], [476, 229], [515, 337]]
[[217, 160], [476, 229], [515, 337], [187, 395]]
[[217, 160], [476, 229], [515, 337], [187, 395], [30, 216]]
```

MORE INFORMATION

- Official OpenCV documentation: https://docs.opencv.org/
- GeeksforGeeks OpenCV Python tutorial: https://www.geeksforgeeks.org/opencv-python-tutorial/
- Learn OpenCV: https://learnopencv.com
- PyImageSearch: https://pyimagesearch.com/

JÖNKÖPING UNIVERSITY

School of Engineering