Lab 1 – Background modeling

General information:

- Run files in Anaconda Powershell Prompt e.g. `python example1.py`
- Remember that when you create your own source code, the indentation in loops, whether in conditions should be equal to one tab, or 4 characters
- Follow the instructions in the examples one by one and in the case of the tasks set out, implement them (detailed task is always presented at the end of the instructions), answer the questions and include it in the report. Send the report to the lecturer with the title of e-mail **IPbVSLAB1 RAPORT**

Example 1 - Reading image data from a file and writing to a file.

- Download the `lena.png` test image from Moodle to your working folder
- Run the `example1.py` file downloaded from Moodle
- Note the conversion of the RGB color space to HSV and the example of color selection used
- Note the color spaces in the OpenCV documentation https://docs.opencv.org/3.4.8/df/d9d/tutorial_py_colorspaces.html
- Note the saving of images in the OpenCV documentation https://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html

```
import cv2
import numpy as np


img = cv2.imread('lena.png',cv2.IMREAD_UNCHANGED)
cv2.imshow('input image',img)

hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# define the color range in HSV
lower_value = np.array([170,10,50])
upper_value = np.array([205,255,255])

# Threshold HSV image to get only selected colors
mask = cv2.inRange(hsv, lower_value, upper_value)

# AND operation of the input image and mask
res = cv2.bitwise_and(img,img, mask= mask)

cv2.imshow('thresholded components',res)

cv2.imwrite('modified_lena.jpg', res, [cv2.IMWRITE_JPEG_QUALITY, 100])


cv2.waitKey(0)
cv2.destroyAllWindows()
```

Example 2 - Reading data from a video sequence and from a camera and saving data to a file.

- Download `test.mp4` (RAR packed) with Moodle to your working folder
- Run the `example2.py` file downloaded from Moodle
- Change the data source to the camera, change `cv2.VideoCapture` parameter to 0
- Make an example change in the resolution of the video sequence (proposed solutions in the yellow frame, and convert to monochrome the saved frame)

```
import cv2
import numpy as np

# Create a VideoCapture object and read the data from the input file
# If the input is a camera, enter 0 instead of the video file name

cap = cv2.VideoCapture('test.mp4')
#cap = cv2.VideoCapture(0)

# Check if the source opened successfully

if (cap.isOpened()==False):
    print ("Error opening video stream or file")

# We get the default frame resolutions.
# We convert resolutions from floating point numbers to integers.

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

# Define the codec and create the VideoWriter object. The output is stored in the "outpy.avi" file
out = cv2.VideoWriter('outpy.avi',cv2.VideoWriter_fourcc('M','J','P','G'), 10,
(frame_width,frame_height))

# Read until the sequence is completed

while(cap.isOpened()):

    # Read until the sequence is completed
    ret, frame = cap.read()

    if ret == True:

         # Save the output frame
         out.write(frame)

         # Display the resulting frame
         cv2.imshow('Frame',frame)


         # Press Q on the keyboard to exit
         if cv2.waitKey(25) & 0xFF == ord('q'):
             break

    # Break the loop
    else:
        break

# When everything is done, release the video capture object

cap.release()
out.release()

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

```
# We define a new variable, GRAY, as a frame transformed into a monochrome frame
# We implement this with use the function BGR2GRAY. It should be noted that OpenCV reads colors as BGR
# (blue green red) while most computer applications read as RGB
# (red green blue). Remember that.

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

#scale
scale_percent = 50/100

calculation of the new dimensions
width = int(frame_width * scale_percent)
height = int(frame_height * scale_percent)

dsize = (width, height)

# resolution change, dsize must contain values of type int
frame2 = cv2.resize(frame, dsize)
```

Example 3 – Estimation of the background

- Download highway.mp4 (RAR packed) with Moodle to your working folder
- Run the `example3.py` file downloaded from Moodle
- Note the implementation of the loop that generates `frameIds` vector elements.
- Note the method for determining the mean and median in the image vector `numpy.mean` and `numpy.median` object functions.
- Note the difference between the mean and median results, discuss the complexity and computation time of the both algorithm.

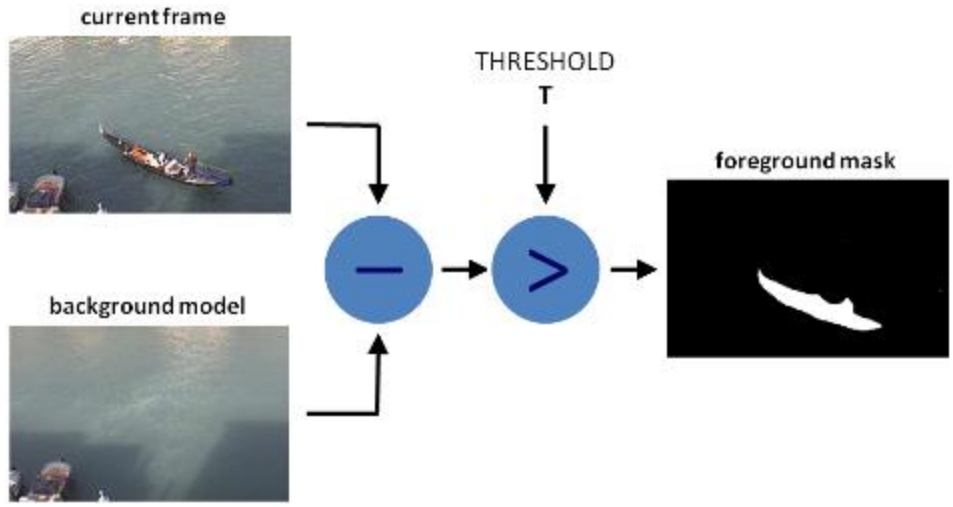
```
import numpy as np
import cv2

# Open Video

cap = cv2.VideoCapture('highway.mp4')

number_of_frame=0
frameIds = []



while number_of_frame < 101:
    frameIds.append(number_of_frame)
    number_of_frame=number_of_frame+1


# Store selected frames in an array
frames = []

for fid in frameIds:
    cap.set(cv2.CAP_PROP_POS_FRAMES, fid)
    ret, frame = cap.read()
    frames.append(frame)

# Calculate the mean along the time axis
meanFrame = np.mean(frames, axis=0).astype(dtype=np.uint8)

# Calculate the median along the time axis
medianFrame = np.median(frames, axis=0).astype(dtype=np.uint8)
```

```
# Display median frame
cv2.imshow('median frame', medianFrame)
cv2.imwrite('medianFrame.jpg', medianFrame, [cv2.IMWRITE_JPEG_QUALITY, 100])

# Display mean frame
cv2.imshow('mean frame', meanFrame)
cv2.imwrite('meanFrame.jpg', meanFrame, [cv2.IMWRITE_JPEG_QUALITY, 100])



cv2.waitKey(0)

# When everything is done, release the video capture object

cap.release()
# Closes all frames
cv2.destroyAllWindows()
```

Example 4 – Foreground mask generation

Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras.

As the name suggests, BS calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene.



Background modeling consists of two main steps:

1. Background Initialization;

2. Background Update.

In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.

- Download `currentFrame.jpg` with Moodle to your working folder
- Use the `medianFrame.jpg` from the output of the previous example.
- Run the `example4.py` file downloaded from Moodle
- Note the *`absdiff`* and *`threshold`* functions.

```
import cv2
import numpy as np


currentFrame = cv2.imread('currentFrame.jpg',cv2.IMREAD_UNCHANGED)
backgroundModel = cv2.imread('medianFrame.jpg',cv2.IMREAD_UNCHANGED)


first_gray = cv2.cvtColor(currentFrame, cv2.COLOR_BGR2GRAY)
gray_frame = cv2.cvtColor(backgroundModel, cv2.COLOR_BGR2GRAY)

difference = cv2.absdiff(first_gray, gray_frame)
_, difference = cv2.threshold(difference, 25, 255, cv2.THRESH_BINARY)

cv2.imshow("Current frame", currentFrame)
cv2.imshow("Background Model", backgroundModel)
cv2.imshow("thresholded difference", difference)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Example 5 – Background subtractor – Mixture of Gaussians, KNN

- Run the example5.py file downloaded from Moodle
- Note the definition of subtractor: `createBackgroundSubtractorMOG2()` and `createBackgroundSubtractorKNN`

    - Inside the parenthesis we can change the value of the subtractor. **History** is the number of the last frame that are taken into consideration (by default 120).
      The **threshold** value is the value used when computing the difference to extract the background. A lower threshold will find more differences with the advantage of a more noisy image.
      Detectshadows is a function of the algorithm that can remove the shadows if enabled.
      There are no right or wrong values, you need to try different settings to see what best fits your need.

- subtractor = cv2.createBackgroundSubtractorMOG2(history=120, varThreshold=50, detectShadows=True)

- *https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html*
-

```
import cv2 as cv


# MOG2
backSub = cv.createBackgroundSubtractorMOG2()

# KNN
#backSub = cv.createBackgroundSubtractorKNN()


cap = cv.VideoCapture('highway.mp4')
if not cap.isOpened:
    print('Unable to open file')
    exit(0)

while True:
    ret, frame = cap.read()

    if ret is True:


        fgMask = backSub.apply(frame)


        cv.imshow('Frame', frame)
        cv.imshow('FG Mask', fgMask)

        if cv.waitKey(25) & 0xFF == ord('q'):
            break


    else:
        break

capture.release()
# Closes all frames
cv.destroyAllWindows()
```

Homework

- **Prepare the source code of the program – background subtractor, in which the goal is to determine the mask of foreground elements, as is done in example 5. The average background should be used as the model of the estimated background instead the MOG or KNN (the option should also be the median filtration as an estimation of the background).**
- **Reporting on the implementation of the task:**
    - **program code in the `.py` file to run**

- **Answers to the questions contained in this manual, plus any hints as to the conditions of the experiment if they are quite specific when running the uploaded, written by you code should be in a separate PDF file.**