HW1: Video Processing, Harris Corner Detection

Due date: 7/4/2020 @ 23:50

Part 1 – Python Basic (0 points)

This part includes several simple assignments and its purpose is to make sure you grasp the basic operations in Python.

You won't be graded on this part, but you are expected to understand it. Failing to understand this part might lead to mistakes in the parts that are graded.

1. How can we select a portion of code and run it without running the rest of the code?
2. How is it possible to stop the code execution in a certain point? For example, before execution of a line of code that may raise an exception. Can we see the call stack and run commands with existing variables?
3. Write a short Python code to randomly generate a 5x6 matrix with values between 0 and 9. Add extra code lines to replace the 3rd smallest value in row #5 with the value 10.

Part 2 – Video Manipulation Basics (30 points)

In this part you will open the attached video file (atrium.avi), conduct several operations on it and save the altered videos to new video files.

Load the video file using VideoCapture and save the new video files using VideoWriter. Read about those functions and classes in the OpenCV help or look it up in the internet.

Write your solution to each question as a python function where the only input is the path to the original video file. The functions do not return any output variables.

All three functions should be declared in the file ex1_part2_functions.py.

All functions read the same input video and save the output video with the same frame rate/ resolution/video size of the input video (don’t write the values hard coded, read the values from the input video).

1. Load the original video, convert the video to black and white (not grayscale) and save it to a new avi file. The conversion from color RGB to black and white has to be done by:
   Compute the global threshold level (Hint: *THRESH_OTSU)* and then set that threshold in the function cv2.threshold. Repeat for each frame.
   The saved filename must be Vid1_Binary.avi. The function name is Q2A.

2. Load the original video, convert each frame to a grayscale image and save the new video file.
   The saved filename must be Vid2_Grayscale.avi. The function name is Q2B.

3. Describe the Sobel operator (you can find it in Wikipedia) and explain how it may be used to obtain the edge map of each frame of the video.
   Apply the Sobel operator on each frame as mentioned (use cv2.Sobel).
   The saved filename must be Vid3_Sobel.avi. The function name is Q2C.

Part 3 – Harris Corner Detection (70 points)

In the first part, answer (**add your answer in a PDF file**) :

1. Is Harris corner detector invariant to translation? Yes/No? Explain.
2. Is Harris corner detector invariant to rotation? Yes/No? Explain.
3. Is Harris corner detector invariant to constant illumination ($I_{out} = a * I_{in} + b$) ? Yes/No? Explain.

The idea to answer is if we take an image I1 and apply translation/rotation/illumination (and get a new image I2), will the two images have the same corners?

No need for a long answer, just the general direction.

In this following part you will implement the Harris corner detection and run it on 2 images.

You'll create 2 functions: myHarrisCornerDetector and createCornerPlots (you can add additional functions for your needs).

myHarrisCornerDetector accepts an image input (color or B&W), a value K and a Threshold, finds all the corners using a 5x5 neighborhood for each pixels, and returns a binary matrix as its output, the same dimensions as the dimensions of the original image where pixels that are corners have the value '1' and all other pixels are of value '0'.

createCornerPlots accepts the original image and corner detection results and creates a plot showing the original image with red circles on the corners.

This plot *has* to contain 2 subplots: the left one shows image I1 with red circles on the corners and the right side does the same for image I2.

CLARIFICATION (Calling both functions):

OUT = **myHarrisCornerDetector**(IN,K,Threshold)

- IN = input image (can be grayscale or color)
- OUT = output binary matrix (same size as input image)
- K = K value in the formula (see instruction 2 for this question)
- Threshold = Threshold value in the formula (see instruction 2 for this question)
- use_grid = Boolean (True/False) (see instruction 3 for this question)

createCornerPlots(I1 , I1_CORNERS , I2 , I2_CORNERS)

Instructions:

1. You may ignore pixels on the very edge of the images (the 'frame' pixels).
2. Use the formula you saw in class with a 5x5 neighborhood:
    - Compute all derivatives in x and in y in the image.
    - Compute the M matrix (defined in class) for every neighborhood.
    - $Response\_Image = det(M) - k * \left(trace(M)\right)^2$
    - For each pixel check if:

    $$Response\_Image \geq threshold$$

   If it is – this can be a corner.
   This is similar to the equation you saw in class. It's the equation that appears in the original paper.
3. As usually we'll find multiple corners near each other, most of the time we want to keep only the maximal corners in each block of the image. To do that, we'll divide the response image (defined in '2'), into a grid of square blocks of size 25X25. In each block we'll return no more than one corner.
4. You may NOT use ready-made functions off the internet (or library functions that implement Harris corner detection algorithm).
5. You may transform the color image to grayscale before computing the corners, but at the end you have to show the circles superimposed on the *original* images.
   To draw red circles on grayscale image, you will have to transform the image to three channels (cv2.COLOR_GRAY2RGB).
6. For finding the derivatives in x and y you can apply a filter on the image or create a shifted copy of the image file (after converting to grayscale) and simply subtract in x and in y. Such that:

$$I_x = I - I_{shifted\ right} \quad , \quad I_y = I - I_{shifted\ down}$$

Summary of files to submit

Part 1 – submit NOTHING. As I wrote – this part is for your own benefit.

Part 2 – Submit only your Python code for all 3 questions (ex1_part2_functions.py file) .

If your Python code fails to read my video files you will lose all the points for this question!

Part 3 – Submit the following:

- ex1_Q3_functions.py that contains:
    - myHarrisCornerDetector
    - createCornerPlots
- Save the output image as: **ex1_ID1_ID2.jpg**

You will also need to submit **a single PDF** (**ex1_ID1_ID2.pdf)** containing the answers to the questions in 2.3 and 3.1.

Please add the file **HW1.py** to your submission and fill there your IDs.

Running HW1.py should create a figure with the detected corners.

Any compilation errors of any sort equal zero points for that respective question. Please double check your solutions before submitting them!

ALL FILES WILL BE PLACED IN ONE ZIP FILE CALLED: **vp2020_ex1_ID1_ID2.zip** , where ID1&ID2 should be your ID's.

Only one of each student pair should submit the HW.

Good luck! ☺