Assignment #3 – Project assignment with double points!

This assignment is due on April 26th at 23:59:59 via email to `christian.wallraven+MLMIS2018@gmail.com`.

Important: You need to name your file properly. If you do not adhere to this naming convention, I may not be able to properly grade you!!!

If you are done with the assignment, make one zip-file of the **codes** in the `assignment3` directory and call this zip-file `STUDENTID1_STUDENTID2_STUDENTID3_A3.zip` (e.g.: `2016010000_2017010001_A3.zip` for a team consisting of two students or `2016010000_2017010001_2017010002_A3.zip` for a two-student team). The order of the IDs does not matter, but the correctness of the IDs does! **Please double-check that the name of the file is correct!!**

Please make sure to comment the code, so that I can understand what it does. Uncommented code will reduce your points!

Please read the assignment text carefully and make sure to implement EVERYTHING that is written here – if you forget to address something I wrote, this will also reduce your points! Precision is key ☺!

I would prefer ipynb-files for your submission, since you can comment more easily.

Part1 Heart rate analysis from mobile phone camera video (120 points):

In this assignment, you will use Python to build a system that can analyze your heart rate from videos taken by a mobile phone camera.

Setup

Take your mobile phone and switch its camera app to video mode. Switch on the flash. Hold your index finger such that it covers the camera lens and the flash (your phone should have a flash that's very close to the camera lens for this to work!).

You should see a rather homogenous, red picture on the preview – this is the light that is being diffused through your finger. Since the heart actually pumps blood through your finger, we would expect there to be some fluctuations in the picture as well – these are very hard to see with your naked eye, but let's hope we can pull them out using image processing.

Take 4-5 videos of your finger for 40s each (keep your body and the finger still, breathe regularly) so that you have data to work with. Include a video from each of your team members and label them with numbers (hrvid1.mp4, hrvid2.mp4 or hrvid1.mp4 - hrvid3.mp4 for a three-member team). Also, for one team member (whoever has number one), take a second video after 2 minutes (call this 1_2). Finally, for the same team member take one more video of your finger after you did some short exercise to get your heart rate up – call this 1_2

Tip: if you can, use a high frame rate setting >30fps [not too high though]. This may give more reliable data. The pixel resolution of the video is not that important. Keeping your finger still is important though!

Video and image processing libraries

In Python, you should find a suitable library for reading videos. I know that some of you have used OpenCV before, but for now I would like you to use

`Scikit-Video` from http://www.scikit-video.org/stable/io.html

together with the `Python Image Library (Pillow or PIL)` for image processing.

Import skvideo and load a video you created using `skvideo.io.vread`. This will give you a `numpy` array to work with.

Process images

Let's follow the basic pipeline from Marco Altini, which he explains in very nice detail on his blog (https://www.marcoaltini.com/blog/heart-rate-variability-using-the-phones-camera).

The first step is to average all pixels per frame and to visualize this value across time. Do this once for the red channel of all frames and once for the L (luminance-channel) of all frames. You can get the luminance channel by using, e.g., pil.convert("L")), but you are welcome to play around with other color-spaces (e.g., the HSV colorspace used by Marco Altini). Show two plots for your video.

The next step is to filter the signal – we have to get rid of slow drift as well as higher-frequency noise. We briefly talked about filter design in class, but here let's keep it simple and create a **fourth-order** Butterworth **band-pass** filter using scipy's `scipy.signal.butter` function. I'll leave it to you to figure out the proper stopping frequencies to put in – hint, think about what kinds of frequencies you want to preserve! Plot the signal after filtering.

Next, smooth the signal further with a **five-sample** moving average filter. If you recorded at very high frame rates, you may want to go a little higher with this. Pandas has a convenient function called `pd.rolling_mean(x, N)[N-1:]` which should do the trick very efficiently! Plot the signal after smoothing.

In Marco Altini's work, he next **upsamples** the signal, which in his case was recorded at 30fps, to 180fps using cubic spline **interpolation**. We can also use the `pandas` functions for Series called `resample` and `interpolate`. Try to hit a resampled frequency of an integer multiple of your original frame rate around 200Hz (e.g., he had 30fps and upsampled by factor of 6 to get 180Hz, which is close enough to 200). Call `interpolate` with the "cubic" method. Plot the first **2 seconds** of your signal BEFORE and AFTER the resampling and interpolation and check that it looks "nice" now.

Now for the real deal – peak detection of the R-peaks. This is an art in and of itself, but try to implement this using scipy's `scipy.signal.find_peaks_cwt` function. You may need to play around with the parameters to get this working properly – especially, since the second parameter to the function carries the expected widths of the peaks in the signal. Check carefully with your sample rate. Plot the first 20 seconds of your signal with the detected peak locations.

Just to check – perform the peak detection on the non-upsampled, non-interpolated signal and plot the first 20 seconds of your signal with the detected peak locations. Are there differences? Insert your comments in your script.

Now load the five videos, perform all steps in nicely commented and wrapped functions [except for the last check] and plot the heart rate of all five videos.

Comment on whether the heart rate correctly detects the higher heart rate after exercise and whether it is stable across sessions and across people (team members).