COMP.CE.470 Exercise 2

On Linux and macOS the ffmpeg can be installed using your systems package manager. For windows it can be downloaded from here: https://www.gyan.dev/ffmpeg/builds/ffmpeg-git-essentials.7z You can run `ffmpeg -encoders` to verify that your ffmpeg has libx264 support, if for some reason that is not the case you can use some other encoder.

In this exercise, you will compare the course encoder (TAU encoder) in its initial state with the x264. To calculate the BD-rates you need to perform four encoding with different QP values (for example, 20, 25, 30, 35). For the TAU encoder be sure to compile a release version, in Linux and MacOS commandline by using `cmake -DCMAKE_BUILD_TYPE=Release ..` and in VSCode by selecting the build type from the CMake menu. As the input sequence you can use https://ultravideo.fi/video_compression/foreman_352x288.yuv or any sequence you can find on the internet, for example, https://media.xiph.org/video/derf/ , low resolution sequences are recommended to save time.

Examples on how to use the course encoder:

```
.\VideoCourseEncoder.exe -i foreman_352x288.yuv -r 352x288 -n 1 -o output.tau --qp=20

Where:

    -i <input>

    -r <resolution>

    -n <number of frames to encode, if not defined whole sequence>

    -o <output_file>

    --qp <used qp value>

.\DecoderApp.exe -b output.tau -o decoder_recon.yuv
```

Take also note of the encoding times. Calculate the PSNR and SSIM distortions for the encoded videos using FFmpeg. Note that FFmpeg does not have a decoder for the course encoder, so you'll have to decode the video first using the course decoder. The filter documentation can be found from https://ffmpeg.org/ffmpeg-filters.html

Then you should encode the same sequence with four QPs using libx264 for in FFmpeg.

Then you should either use the attached excel document or https://github.com/FAU-LMS/bjontegaard python library to calculate the BD-rate between the libx264 and your encoder. Also compare the encoding times. Include

It is recommended that you create some kind of script that helps you automate this process, because it will be repeated in other exercises.

You should return tabulated the bitrates, distortions, and encoding speeds for both encoders, the BD-rate value and the used FFmpeg commands, and short analysis about the obtained results. Everything returned in a single .txt file.

The important thing to note about FFmpeg is that the order of arguments matters. In general the format is as follows:

```
ffmpeg <input 1 parameters> -i input1 <input 2 parameters> -i input2 <operation> <output 1 parameters> output1
```

For example, the following command will blend together two 1920x1080 raw yuv videos and then encode the blended video with libx264. When using the psnr and ssim filters, they do not produce any output stream, and thus you should use `-f null –` (the last dash is important) as output. In the filter_complex command the [0:v] and [1:v] are the video streams from the first and second output, and the [out:v] is the output stream, which then could be used in further filters or as here, mapped as the output. Overall, ChatGPT is excellent at explaining how to use FFmpeg.

```
ffmpeg -s:v 1920x1080 -f rawvideo -pix_fmt yuv420p -i input1.yuv -s:v 1920x1080 -f rawvideo -pix_fmt yuv420p -i input2.yuv -filter_complex “[0:v][1:v]blend=average[out:v]” -map “[out:v]” -c:v libx264 out.mp4
```