ECE1718H – Design Tradeoffs in Digital Systems
Assignment 1

This assignment (and other succeeding ones) is to help students to grasp the basic fundamentals of digital video compression. The exercises will help you to construct some building blocks and to integrate them forming an abstract video codec system

1. Chroma up-sampling, YUV pixel manipulation, YUV-RGB CSC
   a. Read the YUV 4:2:0 video sequence(s), and upscale it to 4:4:4
   b. Convert the 4:4:4 YUV content to corresponding sequence(s) of RGB .png files and visually compare – Notes:
      i. You may use the following formulas for CSC:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & 0.081312 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0 & 1.596 \\ 1.164 & 0.392 & 0.813 \\ 1.164 & 2.017 & 0 \end{bmatrix} \times \begin{bmatrix} Y-16 \\ U-128 \\ V-128 \end{bmatrix}$$

   c. Repeat (a) and (b) but adding random noise to the Y, U, V, R, G, and B components (one at a time) and observing the impact on subjective quality versus the original YUV

2. Basic block-based operations on video content and quality assessment metrics
   a. Read the Y-component of 4:2:0 video sequences, and dump it into corresponding *Y-only* files of each sequence
   b. Read every Y-only file, and apply to it the following operation(s):
      i. Split each frame into (*i*×*i*) blocks (where (*i*) takes the values 2, 8, and 64)
      ii. Use "padding" if the width and/or height of the frame is not divisible by (*i*). Pad with gray (128)
   c. Calculate the average* of the sample values within each (*i*×*i*) block
      * Use efficient rounded division by (*i*×*i*) while calculating the approximated average
   d. Replace every (*i*×*i*) block with another (*i*×*i*) block of identical elements of this average value to generate Y-only-block-averaged file(s)
   e. Subjectively compare* every original Y-only file with its corresponding Y-only-block-averaged one
      * In addition to the frame to frame comparison, you can use simple frame differencing (multiplied by an arbitrary factor to magnify the deltas)
   f. Using average PSNR and SSIM among frames, compare every original Y-only file with its corresponding Y-only-block-averaged one
   g. Plot graphs that relate objective quality, with (*i*) for the tested sequences

3. Basic Motion Estimation/Compensation:
    a. Repeat parts a and b of exercise 2
    b. Using integer pixel full search, find the best predicted block for every (*ixi*) current block; Except for frame boundary blocks that need special handling, use a search range of +/- *r* pixels in the horizontal and vertical dimensions (centered around the collocated block of ***reconstructed*** previous frame) . Use (*r* = 1, 4, and 8)

        i. While predicting the first frame, assume a hypothetical reconstructed reference frame with all samples having a value of 128

        ii. Use Sum of Absolute Difference (SAD) as the ME assessment metric. In case of a tie, chose the block with the smallest motion vector (L1 norm: |x|+|y|, not Euclidean). If there's still a tie, chose the block with smallest y. If there are more than one, chose the one with the smallest x.

    For boundary blocks, do not consider vectors that would make the block fall partially or totally outside the previous frame.

    c. Dump the x and y components of every successful MV into a text file using an arbitrary format that preserves the coordinates of the block

    d. An (*i*×*i*) residual block will be generated by subtracting the predicted block from the current block

    e. Generate an approximated residual block by rounding every element in the (*i*×*i*) residual block to the nearest multiple of 2^n (for *n* = 1, 2, 3)

    f. Dump the (*ixi*) approximated residual values into a text file using an arbitrary format that preserves the coordinates of the block

    g. Add every (*i*×*i*) block of approximated residuals to the corresponding (*i*×*i*) predictor block, generating an (*i*×*i*) reconstructed block, which should be used in the prediction process of blocks in the following frames

    h. Repeat for all (*ixi*) blocks in the frame (processed in raster order, starting from the top left block of each frame, ending with the bottom right block of the frame)

    i. A Y-only-reconstructed file will be constructed from the reconstructed (*i*×*i*) blocks

    j. Compare every original Y-only file with its corresponding Y-only-reconstructed file. Use subjective and objective quality metrics. Highlight cases to showcase the impact of content type, resolution, as well as the values of *i*, *r*, and *n*

    k. Decoder

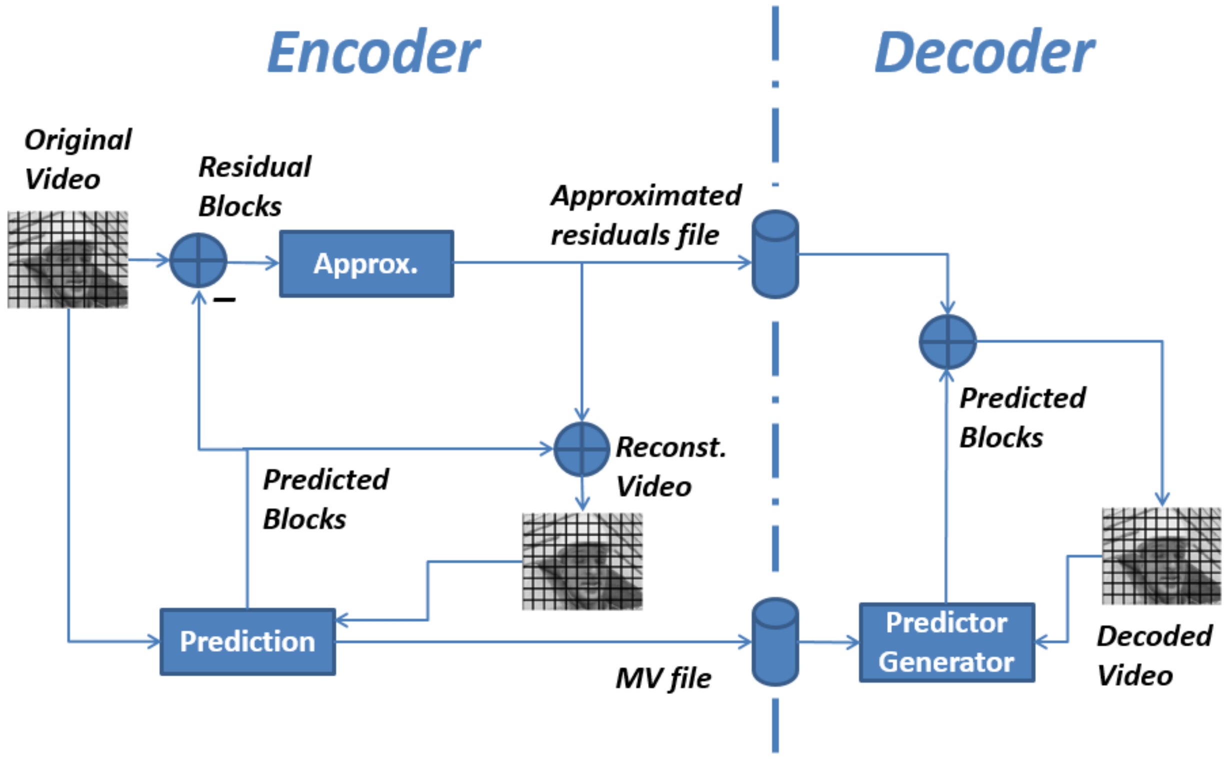
        i. Construct a decoder that uses the approximated residual file as well as the MV file (that were generated in the encoding process) to generate a Y-only-decoded file

        ii. The decoder should add the approximated (*i*×*i*) block to the predictor block that can be identified by the corresponding MV to generate the *Y-only-decoded* files

iii. All *Y-only-decoded* files should be compared against the *Y-only-reconstructed* files of Part (2). They should fully match

iv. The same as for the encoder, the blocks should be processed in raster order, starting from the top left block, ending with the bottom right block. Also, assume a hypothetical reference frame of all-128-values while decoding the first frame

v. The given figure shows a high-level view of the encoding/decoding process as described in this exercise



General Notes

- You can download 4:2:0 YUV sequences from this webpage. Make sure to select sequences with various resolutions and types of motion to showcase the impact of the encoder's settings. For practical execution time with 1080p content, use only a subset of the sequence
- To view/playback the raw YUV content, you can use this tool, or any similar one.
- A recommended practice is to parametrize your code, and to develop your own script to generate the config files, call the executables, collect results, and put them in presentable formats
- In future assignments, "tricks" will be applied to accelerate the encoder's throughput through parallelism. Make sure to develop your code in a platform that allows parallel processing (e.g. multithreading environment, or MATLAB with parallel computing toolbox)

Instructions and Deliverables (For exercise #3)

- Note: you only have to present exercise 3, but you will have to use parts of exercises 1 and 2 in order to complete exercise 3, so it is recommended you do all 3 exercises in order.
- This assignment is to be done in groups of 3 people. You might want to use the Discussion Board in the Portal to search for teammates. Please send an email to a.delmaslascorz@mail.utoronto.ca with the names and UTORids of the group's members by January 30. If you are not in a group by then, we might suggest one for you.
- You can use C, C++, C#, Java, or MATLAB. If you want to use another language, please ask beforehand. As mentioned in the General Notes, the language of your choice should support parallelism, so you should keep in mind if you choose a language such as Python, where some implementations might not expose shared-memory parallel processing in a straightforward manner.
- The deliverables are the source code you developed, two generated files described below, and a presentation (slides), introducing the problem, explaining how it was approached, showing evidence that the system is working, mentioning the observations/challenges, highlighting results to showcase the impact of different encoder's settings, and proposing ideas/thoughts.

  At a minimum, you will want to include a per-frame SAD graph for varying i with fixed r=4 and n=3, another for varying r with fixed i=8 and n=3, and another for varying n with fixed i=8 and r=4. You can use any sequences you want, but the first 10 frames of Foreman CIF (352x288) are a requirement, plus at least one other sequence of different dimensions (number of frames at your discretion).
- For the i=8, r=4 and n=3 case, include a Y-only reconstructed file of the first 10 frames of Foreman CIF (this file should be exactly 1,013,760 bytes in size). Include also a text file with the found motion vectors (the format of this file is arbitrary, but it should contain 15,840 x/y pairs).
- The due date is February 12, 23:59. At this point, the deliverables should be submitted in the Portal (only once per group). During the following week (Feb 13–17), you will have to present you assignment. As the date draws near, you will be requested to select a 15 minute slot for your presentation, where you should go through your slides and demonstrate your code working. It is recommended that you bring your own computer where you can **modify** and run your code (over the first 10 frames of Foreman). If this is a problem for you, please tell us ahead of time.
- If you have any questions about the assignment you can make use of the teaching assistant's office hours (Mondays 5PM-7PM, Engineering Annex room PT306). It is recommended that you email first to make an appointment.