ECE1783H – Design Tradeoffs in Digital Systems

Assignment 2

This assignment (and other succeeding ones) is to help students to grasp the 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.

Beside the report, you should submit the produced source code. As in Assignment 1, the clarity and conciseness of your report will be assessed. You should try to make a convincing case that your encoder is working as it should, explaining and reasoning about the results.

This assignment is to be done in **groups of 3 people**. You can use **Python or MATLAB**, but please use the one that you are most comfortable with. 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 Python, where some implementations might not expose shared-memory parallel processing in a straightforward manner. Therefore, MATLAB is recommended.

The **due date** will be **the evening of Sunday, November 17 (11:59pm)**. By the deadline, the deliverables should be submitted in Quercus (only once per group).

The clarity and conciseness of your report will be assessed. You should try to make a convincing case that your encoder is working as it should, explaining and reasoning about the results.

If you have any questions about this assignment, feel free to post on Piazza. You can also email the TAs:

- Shichen (Stephen) Ji (shichen.ji@mail.utoronto.ca),
- Saleh Tabatabaei (s.mirzatabatabaei@mail.utornoto.ca)
- Jiahong Dong (jay.dong@mail.utoronto.ca)

Contents of this Assignment

Exercise 1: Implementing Optimizations

Apply the following changes to the encoder you developed in Exercise 4 of Assignment 1. Make sure that each of them can be toggled on and off.

1. Multiple Reference Frames

a. Add to config file a parameter nRefFrames, which takes any value from 1 to 4

b. Your encoder should be able to find the best match in any of the reference frames. Your MV will need a third component to represent the used reference frame. Different blocks inside the frame may refer to different reference frames

c. Your decoder should be able to reverse this process. Hence the reference frame index $(0..3)$ needs to exist in the bitstream. Coding it as the difference to the previous one will lead to the range $(-3..3)$

d. Intra frames clear all previous references. Larger number means older frame. It is not possible to use reference 3 unless the encoder has previously encoded/reconstructed three P-frames after an Intra frame

2. Variable Block Size

a. Add to your config file a parameter/switch called VBSEnable

b. When VBSEnable is set, for all I and P frames, allow every $2^j \times 2^j$ block (where j = 2, 3, or 4, depending on the coding block size) to be encoded with a single MV/mode or with four MVs/modes (each corresponding to a $2^{j-1} \times 2^{j-1}$ sub-block). The encoder should choose the right block size depending on the cost that corresponds to it (i.e., comparing the single RD_Cost that corresponds to the larger block size, with the sum of the four RD_Costs that correspond to the smaller block sizes). At least basic implementation of RDO is required for this part to make sense. If you rely on distortion (measured by comparing the final reconstructed pixels of a block with its original ones, using Sum of Absolute Differences (SAD) for example) only, ignoring bit costs for MVs/modes/residuals, the encoder will almost always tend to choose smaller block sizes.

c. You are encouraged to find a suitable relation between lambda and QP that serves the purpose of improving the overall R-D performance across the range of QPs for your encoder. For example, pick q candidate QP values that approximately cover the range of possible values of QP; Creatively try different candidate sets of corresponding lambda values, and pick the one that leads to an overall better R-D performance (higher curve). You can use interpolation to estimate the lambda values that correspond to the remaining QP values that were not in the candidate q QP values (if any). You can try small values of q (e.g., $q = 2$); Although it is far from being accurate, the number of tested combinations of the corresponding lambda values would be manageable.

d. To simplify the implementation, assume that if a block is split into four $2^{j-1} \times 2^{j-1}$ sub-blocks, every sub-block will be later transformed using its own $2^{j-1} \times 2^{j-1}$ DCT. In this case, the quantization process will use a matrix corresponding to QP-1, unless QP was already zero. Note that varying QP per block size would lead to comparing RD costs that are calculated using different values of lambda, which is unpreferred. It is preferable to use the same value of lambda when

comparing R-D costs. Hence, consider using lambda that corresponds to the larger block when comparing R-D costs.

e. Your decoder should be able to reverse this process. Hence a flag needs to exist in the bitstream to indicate whether a block is split or not (0=not split, 1=split; if split, then encoded date of four $2^{j-1} \times 2^{j-1}$ sub-blocks follow, ordered in "Z" shape)

f. Except for the first block in a row, while differentially encoding/decoding the MV/mode of a block (or sub-block), use the MV/mode of the last encoded/decoded block (or sub-block) as the predictor

g. While encoding intra sub-blocks, reference to reconstructed pixels of neighboring sub-blocks within the same block may sometimes be necessary

3. Fractional Motion Estimation

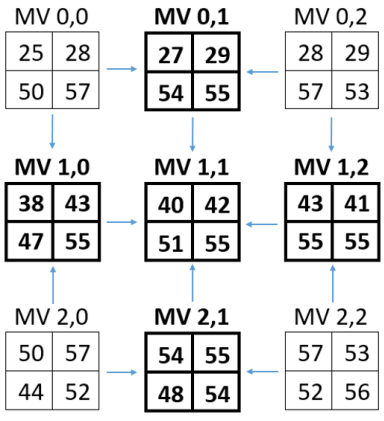
a. Add to your config file a parameter/switch called FMEEnable

b. When FMEEnable is set for a P-frame, the motion magnitudes will be scaled by 2 in both directions. The effective motion vector length corresponds to the values divided by 2. During ME/MC, if both components of the scaled MV are multiples of 2 (even), MC works as it did before; if any of the two components is odd, the reference samples have to be interpolated. This is done by an averaging process as explained below. Whenever any direct neighbors (horizontal or vertical) represent *actual* reference pixels, they should be used during interpolation.

Example – Assume that the shaded pixels in the following diagram form the 2×2 co-located block (MV 0,0) in the reference frame, while the non-shaded pixels represent the neighboring *full* pixels in the reference frame (3×3 region).

25	28	29
50	57	53
44	52	56

With FMEEnable set, the following are possible MVs pointing inside this region (interpolated references are in **bold**, and arrows point to the averaging process):



Implementation Note: In the (MV 1,1) case, each interpolated pixel is the average of 4 other interpolated pixels. Hence, interpolated values are used several times. In order to implement this, you can interpolate on demand (reading $i + 1 \times i + 1$ samples), or build 4 buffers with all 4 possibilities (original and displaced half a sample in one, the other, or both dimensions), or build a large $2X \times 2Y$ buffer (where $X \times Y$ are the picture dimensions). For a decoder, interpolating on demand is likely to be faster. For an encoder, since every position is checked several times, building a pre-interpolated buffer is likely to be faster.

c. Your decoder should be able to reverse this process

Note: for this part, assumes the search range remains unchanged in terms of source pixels. You will therefore have to search more candidate positions.

4. Fast Motion Estimation

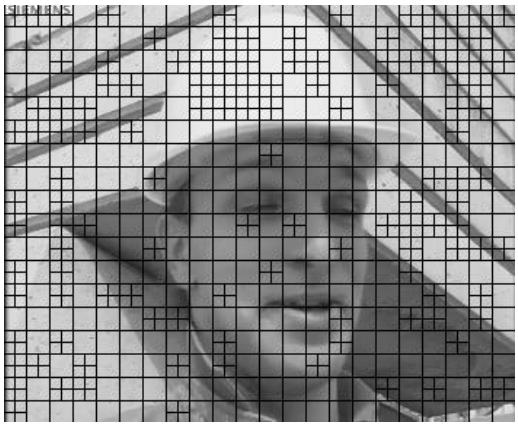
a. Add to your config file a parameter called FastME

b. When FastME is enabled, the full-range ME search is replaced with a fast ME algorithm. Nearest Neighbors search is a requirement, with MVP being the MV of the latest encoded block (MVP = (0,0) for first block in every row of ($i \times i$) blocks). Note: any candidate block that partially or fully exists outside of the frame is not searched.

Deliverables of Exercise 1

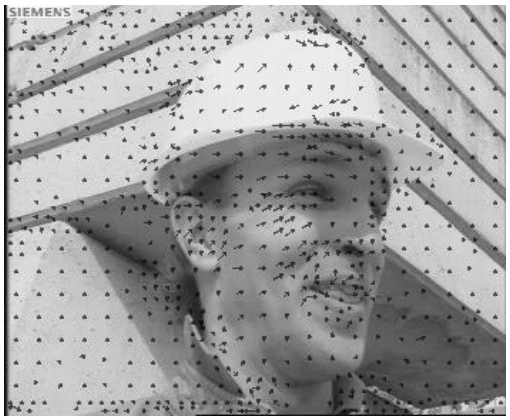
In your written report, include the following:

- Create RD plots for a fixed set of parameters (block size = 16, search range = 4, I_Period = 8). There should be 6 curves drawn: one for the encoder of part1, one for each feature added in this assignment (by itself), and one with all four features. Like in Assignment 1, use the first 10 frames of Foreman for testing, and build your curves using at least QPs 1, 4, 7 and 10. It is important to include execution times as well in the comparisons.
- While only enabling the Variable Block Size feature, report the percentage of blocks that were chosen by mode decision to be split into four sub-blocks for every tested QP value. Basically, you need to produce a curve whose x-axis is the tested QP value, and the y-axis is the percentage of blocks within the sequence that were chosen to be split.
- Replace the x-axis (QP value) with bitsteam size and produce another curve.
- For the test file "synthetic.yuv" provided in Quercus, and using only QP=4 with no other experiments, plot a per-frame distortion and encoded bitstream size graphs for each varying setting of nRefFrames, from 1 to 4. Explain your results.
- Visualizations: include a simple tool that helps visualizing a P-frame (or an I-frame) of your choosing overlaying the variable block size decisions on top. Suggestion (left):



Do the same for the multiple reference frame support, where you mark each block with the index of the reference frame it is using. It does not have to be a number (can be a color, such as in the example on the right), and variable block size does not have to be turned on for this.

Do the same for MVs, where a map of MVs can be overlaid on the top of any arbitrary P-frame. For example: (feel free to get creative with the representation/ visualization)



For I-frames, a layer of the selected intra mode should be overlaid on top.

Answer the following questions in your report:

- What effect do Multiple Reference Frames, Variable Block Size, and Fractional Motion Estimation have on encoding speed and quality? How do they compare to using a smaller block size and/or a larger search range?
- Could the effect of Multiple Reference Frames depend on content type? Can you think about what kind of non-artificial video could benefit the most from it?
- What trade-offs are involved in using a feature like Fractional Motion Estimation, other than performance? Some video standards support up to 1/8-pixel ME, would it be a good idea to use such a feature unconditionally? Why?
- For Variable Block Size, what kind of areas get larger block sizes in Intra frames? Are they the same in Inter frames? Why?