ECE1783H – Design Tradeoffs in Digital Systems
Assignment 3

Exercise 1

In this exercise, you will implement a basic rate control algorithm on the top of the encoder that you have implemented in assignment 2. You will need to do the following:

a. For every specific target set of configs, run the encoder at all different QP values for adequately different sequences. Create a table for average bit-count per row of (ixi) blocks at each QP value. You will need a table for P-blocks, and another for I-blocks. You will have a separate set of tables per tested resolution. The following is an example of how the P-frame table will look like for one possible 4x4 block encoder (numbers are assumed; you will need to obtain more realistic numbers by experiments and average measurements)

QP Value	0	1	2	3	4	5	6	7	8	9
Average bit-count per (4x4) row of blocks for P-frames	2,112	1,936	1,760	1,584	1,408	1,232	1,056	880	704	528

b. Add an "RCflag" to the encoder configs, which indicates whether RC is ON or OFF (if OFF, then constant QP mode is used). Add a "targetBR" parameter to the encoder (unit is bps, kbps, or mbps), which is used only if (RCflag==1). For simplicity, assume a very restrictive and latency sensitive algorithm, which splits the bit-budget evenly among frames and rows of blocks, regardless of the content or prediction type. For example, if we assume that the targetBR is 1,140,480 bps ("~1 mbps"), then
   - Every second of video is allocated a bit-budget of 1,140,480 bits
   - Assuming 30 fps video, then every frame is allocated a bit-budget of 38,016 bits
   - Assuming a (352x288) CIF frame, split into (4x4) blocks, then every row of blocks is allocated a bit-budget of 528 bits

c. Your RC algorithm should operate as follows
   i. Given the encoder configs, the target resolution, and the RC parameters, find the bit-budget per row of blocks (as described in part (b))
   ii. Using the table(s) obtained in part (a), find the QP value to use in order to meet (not exceed) the target per-row bit-budget
   iii. After encoding the row of blocks, if entire frame is encoded, go to step (i) and start encoding the following frame (if any); else, subtract the actual bit-spent from the remaining budget per frame, to obtain the new remaining budget per frame. Assume that this new remaining budget will be split evenly across all remaining rows of blocks, and calculate the new target per-row bit-budget; then go to step (ii)

d. Note that you need to formulate a way to communicate the QP values (per row in our case) to the decoder. For bitrate efficiency, encode the per-row QP differentially (assume any hypothetical reference QP value for the first frame), and embed its Exponential Golomb code into the bitstream at the start of every row. Show evidence that the decoder is able to extract the correct QP values from the bitstream, and to generate bit-accurate decoded video, which fully matches the encoder's reconstructed video

Deliverables of Exercise 1

Configure your encoder to operate with (nRefFrames=1, VBSEnable=1, FMEEnable=1, and FastME=1), using block size = 16x16, bounding the MVs to be within a range of +/-16 pixels around the collocated block, and other optimization modes at your discretion. Obtain the corresponding tables for part (a), based on statistics collected from the provided artificially-generated CIF and QCIF sequences (a 21-frame artificial sequence composed of the first 7 frames of Foreman CIF/QCIF, followed by the first 7 frames of Akiyo CIF/QCIF, followed by the second 7 frames of Foreman CIF/QCIF). In order to fill the tables, you will be averaging bitcount throughout the 21 frames. The table(s) that you use in part (c) should include all QP values from 0 to 11. Ideally, the average bitcount that corresponds to every QP value in the table should be measured statistically. However, practically, if you believe that the stats-collection phase would be too time-consuming, you can interpolate every other QP bit-count by averaging the two direct neighboring measured bit-counts (e.g., bitcount of QP1 = [bitcount of QP0+ bitcount of QP2]/2). You will be producing 4 tables in total: CIF and QCIF, I-Frame and P-Frame for each. To get I-Frame statistics, you can encode with an I_Period of 1 (only I-frames) and take the average. To get P-Frame statistics, use an I_Period of 21 and average the bitcount of the corresponding 20 P-Frames. Include these tables in your written report. Note: The requirement is to collect stats and obtain results based on the provided sequence. However, for a more realistic solution, feel free to additionally collect statistics based on a set of sequences, then applying it to compose tables that are used while encoding another set of sequences. If you choose to do so, please keep the results separate from the main requirement

Using the algorithm in part (c) and the same encoder configs you used for stats-collection, encode the sequences targeting a bitrate of 2.4 mbps for the CIF sequence, and 960 kbps for the QCIF one, with I_Period = 1, 4 and 21. Assume sequences have 30 frames per second. In your report, include:

1. Pictures of the decoded 4th frames (one for each I_Period and sequence – six in total)
2. Per-frame bit cost and PSNR graphs (for each I_Period and sequence)

For the CIF sequence with I_Period = 21, try to match the total size as closely as possible using a fixed QP, with no RC. To do so, encode it using (say) a QP of 6. If file size comes too big, try QP = 8 [(6+11)/2]; otherwise try QP = 3 [(0+6)/2], until you find the QP that produces the closest total size. In your report, include a per-frame bit cost and PSNR graphs of this sequence.

Answer the following questions:

- Why do table(s) in part (a) vary according to the video's resolution? Would linearly scaling the bitcount in the table of some specific resolution be sufficient/accurate to obtain the table for another resolution?
- Using the algorithm in (c), do you see any negative behavior at I-frames and/or scene-changes? Any thoughts on how to avoid this?
- How does the global quality compare between the suggested RC encoded sequence and the one encoded with fixed QP. Give examples of scenarios where we have to use RC, and others where encoding with fixed QP is possible/advisable.

Something to explore (but does NOT contribute to the grade)

- Now that the encoder can operate in RC mode (unlike Assignments 1 & 2), you can play with other encoder parameters to see the impact of prediction accuracy on per-frame quality and bitcount.

Exercise 2

This exercise introduces the students to the concept of multiple-pass encoding. The idea is as simple as encoding every frame in two passes; during the first pass, encoding is done with a reasonable constant QP value, and statistics are collected to reflect further info about the content. These statistics are used to improve the decisions on the second encoding pass, targeting better quality at a specific bitrate. The following are examples of improvements that can be introduced when 2-pass encoding is enabled (i.e., RCflag==2):

a. If total bit-count for a P-frame is higher than a reasonable threshold, the frame is considered a scene change. You will need to come up with this threshold as a function of QP. A scene change is encoded as an I-frame in the second pass, with RC ON, as described in exercise 1(c). Notice that in this specific case, the prediction method changes between the two encoding passes, and hence, no other info from the first pass can be leveraged for the second pass (i.e., 2nd pass will be as blind as the process described in exercise 1(c))
b. Exercise 1(c) assumed that the frame bit-budget is split evenly across all (remaining) block rows in the frame. In exercise 2, during the first pass, you will collect the per-block-row bit-count at constant QP. Assume that during the second pass, the per-block-row bit-budget are distributed in proportion to the bit-count statistics that were collected at first pass. For example, at first pass, if 5% of the frame's bitcount is spent on the first row of blocks, then during the second pass, the first row of blocks will be allocated 5% of the frame's bit-budget
c. Exercise 1(a) assumed a fixed table to refer to for average bitcount per QP value. One enhancement that you should consider in exercise 2 is to leverage info that is obtained in the first pass, to make the QP/Bitcount table more realistic/reflective. For example, if first pass used QP=3, then a scaling factor can be obtained by dividing the *real* average per row bitcount obtained from the first pass by the *statistically obtained* QP3 bitcount in the table. Accordingly, for a more reflective bit-budget estimation, all elements in the table can be scaled using this same factor before use
d. The constant QP value that is used in the first pass of every frame is equal to the average QP value that is used in the second pass of the preceding frame. For the first frame, use any reasonable QP value for the first pass.
e. Obviously, when operating at (RCflag=2), the encoder will suffer considerable performance drop due to the extra pass. Allow the encoder to operate in (RCflag=3) mode, in which throughput drop introduced by 2-pass encoding can be minimized by leveraging info that are obtained during the first pass. For example, during the second pass, ME may take place around the corresponding MVs of the first pass, with a smaller search range to refine. Also, the VBS partitioning scheme of every block can be propagated as is from the first pass to the second pass, etc.

Deliverables of Exercise 2

For this exercise, use the same 21-frame artificial sequence composed of the first 7 frames of Foreman CIF, followed by the first 7 frames of Akiyo CIF, followed by the second 7 frames of Foreman CIF. Use similar encoder configs defined in the deliverables of Exercise 1 (and the same bit count estimation tables), but consider only the case I_Period = 21 (note that insertion of I-frames at scene changes may add more I-Frames).

Consider the following 12 experiments:

- Using RCflag = 0 (constant QP), encode the sequence using QP = 3, QP = 6 and QP = 9
- Using RCflag = 1,2,3, encode the sequence using bitrates of 7 mbps, 2.4 mbps and 360 kpbs

Plot an RD-graph comparing the R-D performance of these experiments (4 curves – one per RCflag value – with 3 points on each). Include a table reflecting encoding times.

Additionally, plot a per-frame PSNR graph comparing RCflag = 1,2,3 for the 2 mbps case (3 curves with 21 points each).

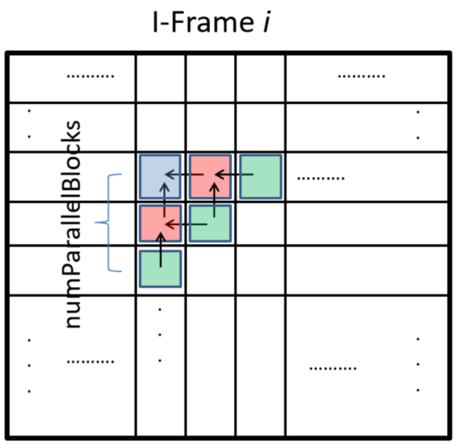
What are your thoughts about the performance/quality tradeoffs introduced by this 2-pass rate control algorithm (RCflag=2) and its low-complexity variation (RCflag=3)?

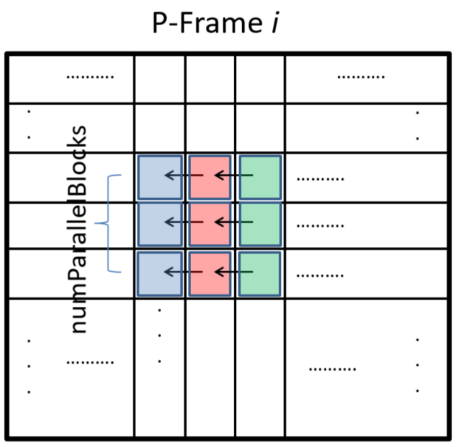
Exercise 3

In this exercise, you will leverage existing data-level parallelism to considerably improve the throughput of your encoder. The following are three different types of parallelism that you are asked to leverage using a programming environment that allows parallel processing on a platform with multiple cores:

a. Type 1: Extreme block-level parallelism (ParallelMode = 1). This is a hypothetical approach where the bitstream format is changed in order to enable the desired high degree of parallelism. This comes at the cost of coding efficiency since motion vectors are stored without differential encoding to break the dependency between blocks, and intra prediction is disabled entirely (reference blocks are assumed to be 128 gray, and no prediction information is transmitted, similar to what you did with the first frame in Ex 3 of Assignment 1). Therefore, all blocks of a frame can be encoded in parallel, and then their bitstreams are appended to each other (in order).

b. Type 2: Block-level parallelism (ParallelMode = 2) – dependency between blocks is preserved and hence no need to sacrifice differential encoding nor intra prediction

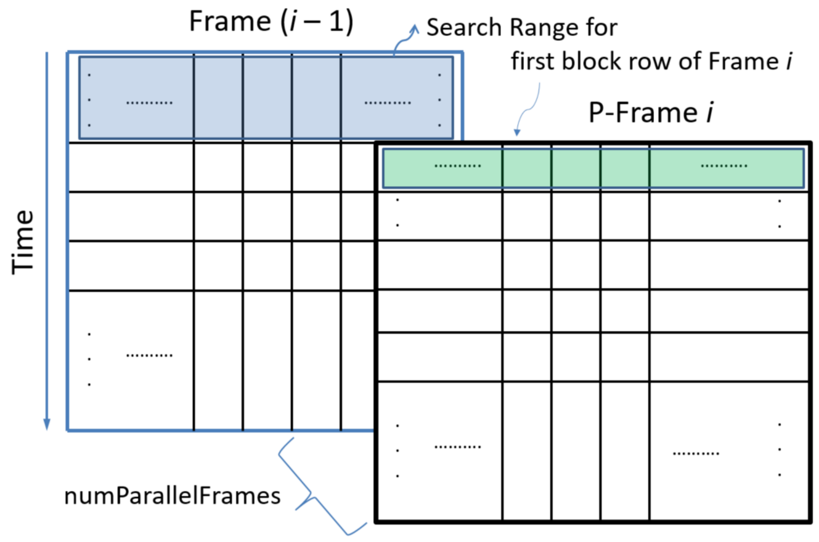




- Blocks with same color can be processed in parallel
- In an I-frame, a block can be intra-encoded/decoded once its top and left neighbors reconstructed pixels are available

- Blocks with same color can be processed in parallel
- In a P-frame, due to differential encoding/decoding of MVs, a block can be encoded/decoded once the MV of its left neighbor is available

c. <u>Type 3:</u> Frame-level parallelism (wave-front parallel processing) (ParallelMode = 3) – dependency between blocks is preserved and hence no need to sacrifice differential encoding differential encoding nor intra prediction



- For a P-frame, a row of blocks can start processing, once all the blocks within its search range are reconstructed at the reference frame
- An I-frame can be fully processed in parallel with its preceding frame

Deliverables of Exercise 3

Implement all three parallelization methods in your encoder, supporting two threads for each method. For Types 2 and 3, the generated compressed stream should not differ from the serially generated one (ParallelMode = 0), while for Type 1, the generated bitstream is not conformant. In all cases, you will need to synchronize the bitstream writing process for both threads (they will have to internally buffer their output and emit it in the correct order – an example is provided below).

In order to simplify your implementation, the following assumptions can be made:

- For Block-level parallelism (Type 2), you do not need to handle P-frames differently from I-frames. You can always use the most simple schedule, where the first thread encodes its blocks in order, and the second thread waits for thread 1 to encode block *N+1* before encoding block *N*, where *N* and *N+1* refer to the horizontal indices of the respective blocks on each row
- For Block-level parallelism (Type 2), you do not need to overlap several rows: once thread 1 finishes, it can write its portion of the bitstream and then wait for thread 2 to finish. Once thread 2 finishes it will write its buffered bitstream, then both threads proceed to the next pair of rows
- For Frame-level parallelism (Type 3), the concept is similar, but in this case the second thread will have to always wait for the (*N+2*)th row of blocks to be encoded before it starts encoding its *N*th row of blocks (that is, when the first thread has completed two rows of its frame, the second one can start with its first row of the following frame)
- For simplicity, you can apply the *N+2*-row delay for every frame, regardless of its type
- You will again need to handle the order of entropy encoded data in the bitstream
- Again, you do not have to overlap frames

You are encouraged to (optionally) use more flexible (efficient) scheduling than these guidelines.

Test your encoder using the same configs as in Exercise 1, on the same provided CIF sequence with an I_Period of 10 and a constant QP of 5 (RCflag = 0). Check that the encoded stream is identical in Types 2 and 3 (compared to Type 0), and compare encoding times. **<u>Note:</u>** For this exercise, correctness of the implementation is more important than performance gains. It is understood that, especially for MATLAB implementations, the communication overhead may sometimes outweigh the parallelization gains. For Type 1, show how disabling intra prediction and differential encoding affects the R-D performance of the encoder compared to Type 0.

Optionally, you can allow your design to support more than 2 threads per parallel mode, as well as combinations of block-level and frame-level parallelism.

What are the pros and cons of each of the three types of parallelism?

Where does the suggested *N+2* delay for Frame-level parallelism come from? Does it depend on some encoder configuration?

What would be the implications of using per-row RC (RCflag=1) in each of the three types? Would the encoded bistream change in both cases?

What about the two-pass RC (RCflag=2 or 3)? In this case, what changes would you apply to Frame-level parallelism? Recall that the final encoding of every frame can only start after the first pass for that frame has finished.

Exercise 4 (BONUS)

Feel free to impress us with any improvements to your encoder's quality/bitrate and/or throughput/power. Use your own approach(es) to showcase the benefits of your ideas. Bonus marks may be awarded for this.

The due date for this assignment is **Sunday December 15**, end of the day. If you have any questions about the assignment you can make use of the teaching assistant's office hours (Wednesdays 4PM-6PM, Engineering Annex room PT306). It is recommended that you email first to make an appointment.