SCHOOL OF ENGINEERING SCIENCE
SIMON FRASER UNIVERSITY
ENSC 424 Multimedia Communications Engineering – Fall 2015
Homework Assignment # 5 – Due Date: Tuesday **Nov. 17**, 11:30 am

---

A test video sequence **glas40.qcif** (40 frames, 176x144 pixels / frame) and a sequence player **YUVPlayer.exe** for Windows are included in the homework package. You can use YUVPlayer to load up to two raw video sequences and play them. Remember to choose the correct image size (QCIF, CIF etc) in YUVPlayer. Note that the video is played behind the control window, so it is better if you maximize the YUVPlayer window, and move the control window to the right side of your screen, so that you can see the video behind it.

Some email servers do not allow exe files in zip files. Therefore we the YUVPlayer.exe is renamed as YUVPlayer.exe.rename in the zip file. Please rename it to exe file to run it.

1. (20 points) Motion Estimation Complexity

(CEAB indicators: 1.1: Mathematical Knowledge, 1.4: Discipline - specific Knowledge, 2.3: Problem Solution)

Consider motion estimation with integer pixel accuracy and the following parameters:

Frame size:	16M x 16N pixels,
Macroblock size:	16 x 16 pixels,
Search window range:	[-7, 7].

Assume that the Sum of Absolute Differences (SAD) is used, and the computation of the SAD between two blocks, i.e.,

$$\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\left|x(i,j)-y(i,j)\right|,$$

requires L^2 operations. Also assume that no early termination scheme is used, and the ME complexity of a block at the boundary is the same as that of a block in the center of the image.

a) (10 points) Determine the number of operations (in terms of M, N) required to perform the motion estimation of one frame when the **full search** method is used.

b) (10 points) Determine the number of operations (in terms of M, N) required to perform the motion estimation of one frame when the **3-step search** method is used.

2. (30 points) Programming: Compute the PSNR between two video sequences
(CEAB indicators: 1.1: Mathematical Knowledge, 1.4: Discipline - specific Knowledge, 2.3: Problem Solution, 2.4: Solution Verification and Evaluation, 3.2: Investigation Design, 4.1: Requirement and Constraint Identification, 4.3: Design Candidate Generation, 4.5: Detailed Design)

Complete the program **yuvpsnr.cpp** to compute the Y-PSNR, U-PSNR, and V-PSNR between two YUV 4:2:0 video sequences. The raw YUV file format used here does not have any file header. The data are saved frame by frame. Within each frame, it stores all Y samples (row by row), followed by all U samples and all V samples. The frame size of both U and V components are 1/4 of the Y component. Therefore the average bits used for each pixel is 12 bits, if each sample is represented by 8 bits.

The program is executed as follows:

```
yuvpsnr seq1name seq2name col row frames
```

where col and row specify the size of the luminance frame, and frames specifies the number of frames to be evaluated.

For each frame index, the program will print out the PSNRs of Y, U, and V components of the two sequences. It will also print out the average Y-PSNR, U-PSNR, and V-PSNR of all frames at the end.

In the hard copy of your report, you only need to print the code you wrote. For soft copy submission, please submit the entire file that you modified.

A test sequence **glas40_test.qcif** is included in the homework package. It has 10 frames. Report its printed PSNRs with respect to the first 10 frames of glas40.qcif in your report.

3. (50 points) Programming: Video Coding

(CEAB indicators: 1.1: Mathematical Knowledge, 1.4: Discipline - specific Knowledge, 2.3: Problem Solution, 2.4: Solution Verification and Evaluation, 3.2: Investigation Design, 4.1: Requirement and Constraint Identification, 4.3: Design Candidate Generation, 4.5: Detailed Design)

A simple video codec with motion estimation/compensation, DCT, quantization, and binary arithmetic coding is provided. The usages are:

```
videnc infile outfile col row frames qstep
viddec infile outfile
```

For example:

```
videnc glas40.qcif z.out 176 144 40 10
viddec z.out dec.qcif
```

Both 4-point DCT and 8-point DCT are provided, and the default setting uses 8-point DCT. You don't have to change this for this homework.

In this question, we will work on the motion estimation part. **All changes are within the file vidcodeclib.cpp, and you only need to change the encoder side.**

codeImage( ) and decodeImage( ) are the main routines at the encoder and decoder, respectively. Two frame buffers, m_pfFrameBuf[0] and m_pfFrameBuf[1], are required for video coding, one for the current frame and one for the reference frame. Each frame buffer contains the Y, U and V components of a frame. At the end of codeImage() and decodeImage(), we need to switch the pointers to these buffers, so that the current frame becomes the reference frame of the next frame.

You need to complete the following motion estimation routines at the encoder:

MBMotionEst( )
GetSAD( )

In the hard copy of your report, you only need to print the code you wrote. For soft copy submission, please submit the entire file that you modified.

Note that the initial code you are given simply sets all MVs to be 0, so the encoder and decoder still work, although not efficient. Before you change anything, **please first record the average bit rate (bits/pixel) and average Y-component PSNRs of the sequence glas40.qcif at the quantization step sizes 5, 10, 20, 30, and 50.**

For each 16 x 16 macro-block, the function MBMotionEst( ) finds the best motion vector within a search window range of [-18, 18]. To prevent random MV, we only update MV when its SAD is less than 0.925 times of the previous minimal SAD.

You need to check the location of the reference block so that all pixels in it are within the frame boundary. Only Y component is used in motion estimation, and the MVs of UV components are chosen to be half of the MVs in the Y component.

GetSAD( ) is called by MBMotionEst( ), and is used to compute the sum of absolute difference (SAD) of two macro-blocks for a particular motion vector. One block is in the current frame and another one is in the reference frame. Your code should use early termination. That is, if the current SAD is already greater than the input parameter iSAD0, we can stop adding the remaining absolute difference.

Once you have completed the programming, **record the average bit rate (bits/pixel) and average PSNRs of the sequence glas40.qcif at the quantization step sizes 5, 10, 20, 30, and 50. Use Matlab or Excel to plot the Y-component PSNR vs Rate curves of the initial codec and your completed codec. That is, PSNR is plotted in the Y axis and bit rate is in the X axis. Which method gives better result?**

Appendix: A brief description of the entropy coding in the simple video codec

For each block, we first send one bit to signal whether the entropy coding for this block can be skipped. If all coefficients are 0, we send a 1 to signal the skipping, otherwise we send a 0, followed by all unary-coded coefficients in this block (this is not optimal, as we have not used any zigzag scanning and efficient signaling of the last nonzero coefficient of a block, but further improvement is beyond the scope of this homework).

At the decoder side, we first decode this skip bit. If it is 1, we set the block of prediction error to be 0, and skip the inverse quantization and inverse DCT. Otherwise, we continue to decode all coefficients in this block, and perform inverse quantization and inverse DCT.

This skip bit is encoded by the binary arithmetic coder with its own context.