Project 4 – Video Codec

This project will need the background of Projects 2 and 3. Please revise your implementations of those projects before you start.

This document will help you put the already implemented code of Projects #2 and 3 in such a way that it will create a complete video codec for you. Remember that you need not write any significant code for the implementation. However, what you will need is all the sub- routines and temp storage to keep the previous frame information and the routines to use this set of data to predict the next frame and so on.

The problem statement (uploaded project file) shows the basic block diagram of the video codec. Approach this project as a modular design for the implementation of each block shown in the block diagram.

Please note that this codec needs you to keep a history of (*i*-1) frame every time, which is used to estimate the parameters for the current frame. (Please understand the codec block diagram in depth before you begin to avoid mistakes).

Please take a logical approach to implement this project since it requires you to break up the entire problem in parts, which are then put together to give you the whole codec.

It is easy to create functions and call them whenever needed in the main program.

NOTES:
1. Keep the Project 2 and 3 help files for ready reference.

2. Show images wherever and whenever possible to score more points in grading.

3. DCT is computed on 8x8 block but the motion prediction is done only on 16x16 Luminance macroblocks. Do not get confused with this.

Step 1: Understand the logical flow of the video codec from the block diagram.

Step 2: Define all the global variables and constants like the global matrix to store the coefficients etc. (see Project 2 help file) and the Quantization matrices.

Now keep in mind that at first only do it for the 1st two frames and then loop it for 5 frames like in Project 3. However, in this project, you will need to do the DCT in a loop too, for every frame.

Step 3:

a. Do 4:2:0 or 4:2:2 sampling on the frame.

b. Have a set of temp matrices here for storing the previously subsampled frames (Y, Cb and Cr) (these matrices are 0 for the 1st frame since there is not previous frame.). Subtract these temp matrices from the currently computed matrices.

c. Compute the DCT for the frame.

d. Apply quantization on the frame.

e. For 1st I-frame we will have approximately
[(174/8 * 144/8) DC coefficients + (174/8 * 144/8) * 63 AC coeffs] for Y component. You can similarly find coefficients for the color components.

f. Go ahead and do the Differential Coding for DC coeffs and ZigZag coding for AC coefficients. Store all the values in a set of matrices for this frame.

g. Immediately apply de-quantization followed by inverse DCT on this frame and store it in another set of temp matrices.

h. You can use the code and functions developed in HW2.

i. The *Sum block* in the block diagram should be ignored and the *Frame storage block* should be used to store the I-frame which has been reconstructed by Dequatization and Inverse DCT.

j. For each subsequent frame (i.e., 2nd frames and later), the 4:2:2 or 4:2:0 subsampled Y-component matrix of *step a* is now used along with the previous frame data (reconstructed I frame) to compute the motion estimation.

k. Please note that **the 1st frame is I frame and the next 4 frames are predicted from it.** It is better to store the 1st frame data permanently and use it every time in the motion estimation code to predict the other frames.

l. The *Predicted Frame Storage block* will temporarily store the current predicted frame.

m. You will compute the difference of each P frame (i.e., original – predicted pixel values) and then perform DCT, Quantization etc. on it.

n. In order to obtain motion vectors, please use the idea from HW3 and create a function, which gives you output motion vectors using I and P frames and MB size.

o. You can use any search criteria to obtain motion vectors.

p. Thus finally you have the following data for each frame:
DC and ZigZag coded data + Motion Vectors. Store this in a set for matrices for each frame.

q. Use the motion vectors to generate motion compensated image.

r. For decoding you need to perform inverse operations such as IDCT, up-sampling.

This is for the first frame. So this way we will have 5 frames, 1 of which will be I frames and the rest P frames.

Please ignore the Entropy coding, MUX and Buffer block. You are also not expected to do byte stream coding in the project. Also ignore the pseudo code given in the Project 4 problem file.

Decoder: You will write a separate code for decoding operation.