Polytechnic University, Dept. Electrical and Computer Engineering
EL6123 --- Video Processing, S10 (Prof. Yao Wang)
Closed Book, 1 sheet of notes (double sided) allowed
No peeking into neighbors. Cheating will result in a failing grade.

---

1. (10 pt) As you may know, there are two types of HDTV formats known as 720p and 1080i. The 720p format is progressive video with a frame rate of 60 frames/s and frame resolution of 1280x720 pels/frame. The 1080i format is an interlaced video with a field rate of 60 fields/s and field resolution of 1920x540 pels/field. Assuming the underlying video can have arbitrary scene content, for the 720p video format, determine what is the maximum (a) temporal frequency (cycles/s), (b) vertical frequency (cycles/ picture-height), and (c) horizontal frequency (cycles / picture-width) that the video can render properly; (d, e, f) Repeat the above for the 1080i format. (g) Next assuming the underlying video content is fairly stationary (i.e., does not change in time), would any of your previous answers change and how? (h) If the video is stored in the 4:2:0 format (i.e. for every 2x2 Y pixels, there are 1 Cb pixel and 1 Cr pixel) and each color component is stored with 8 bits/pixel, what is the raw bit rate (bits/s) of the 720p video? (i) What is the raw bit rate of the 1080i video? (j) Based on you answers above, which video format you think is better and why?

2. (10 pt) We would like to do motion estimation using the DBMA method. We assume that the anchor frame is divided into square blocks and that each block in the anchor frame is mapped to a quadrangle in the target frame. We describe the motion of each block by the nodal movements of the four corner nodes, $\mathbf{d}_k = [d_{x,k}, d_{y,k}]^T$, using bilinear interpolation kernels $\phi_k(x, y)$. Specifically, the motion vector at each pixel (x,y) inside the block can be written as

$$d_x(x, y) = \sum_{k=1}^{4} \phi_k(x, y) d_{x,k}; \quad d_y(x, y) = \sum_{k=1}^{4} \phi_k(x, y) d_{y,k},$$

or in vector notation

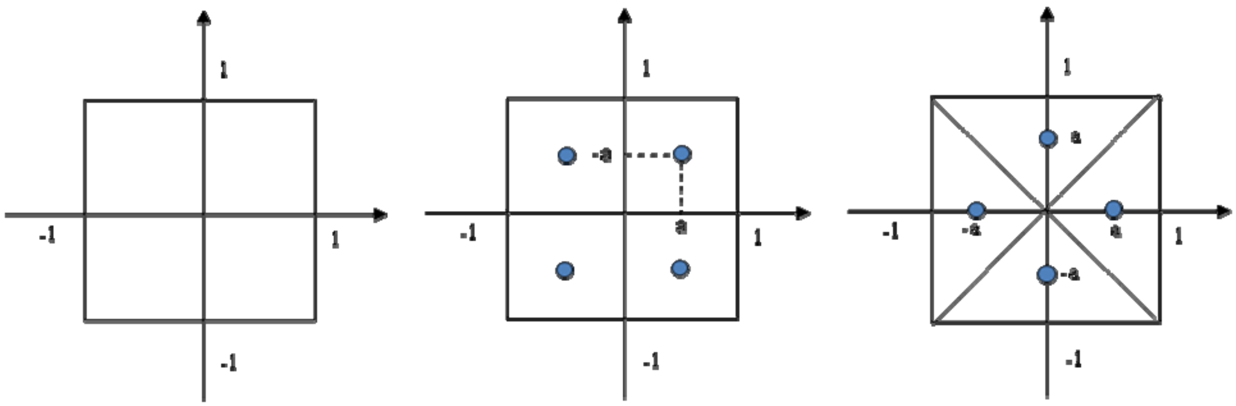
$$\mathbf{d}(\mathbf{x}) = \sum_{k=1}^{4} \phi_k(x, y) \mathbf{d}_k$$

One way to estimate the nodal displacements $\mathbf{d}_k$ or $d_{x,k}, d_{y,k}$ in each block is by minimizing the sum of squared errors between each pixel in the block and its corresponding pixel in the target. Set up your optimization problem using this approach and write clearly the iterative gradient descent algorithm for solving this optimization problem. You can assume you start with zero nodal displacements as initial conditions.

3. (10 pt) In conventional vector quantization, an input vector is compared to each one of *L* codewords, and the codeword that is closest to the input vector (in terms of the sum of squared errors in all vector components) is used to represent the input vector. The index of the chosen codeword is coded using $\lceil \log_2(L) \rceil$ bits. One alternative approach is known as gain-shape VQ. For a given input vector, you derive its gain factor (which is its l2-norm), and the shape vector, which is the input vector divided by the gain factor, so that the shape vector has a norm of 1. Then you quantize the gain factor using a pre-designed scalar quantizer with L_1 levels, and quantize the shape vector with L_2 levels. You represent the quantization level index for the gain factor using $\lceil \log_2(L_1) \rceil$bits, and shape vector codeword index using $\lceil \log_2(L_2) \rceil$ bits. Suppose vector dimension is *N*, and *L*, L_1 and L_2 are chosen such that $L_1 L_2 = L$. Furthermore, assume *L*, L_1 and L_2 are all integer numbers that can be represented as powers of 2. (a) What is the bit rate (bits/sample) using the conventional vector quantizer? (b) What is the bit rate using the gain-shape vector quantizer? How does it compare with the conventional vector quantizer? (c) Determine the complexity of quantizing an input vector using the conventional VQ and represent it in terms of *N* and *L*. You can consider one subtraction, and one multiplication and one addition together as one operation. (d) Determine the complexity using the gain-shape VQ, and represent it in terms of *N* and L_1 and L_2. Assume N is large, how does the two complexities compare? (e) Which method is likely to

yield less quantization distortion? (f) What are the pros and cons of the gain-shape VQ compared to conventional VQ? (List one pro and one con).

4. (10 pt) Consider coding a 2-D random vector that is uniformly distributed over the region illustrated in Fig. (a). Suppose you want to design a codebook with 4 codewords. Two possible codebook constructions (codeword locations and region partition) are illustrated in Figure 1(b) and 1(c), respectively. **For each codebook,**
    a. Determine whether the codebook satisfies the necessary conditions for minimizing the mean square error of the quantizer.
    b. Determine the value of *a* that will minimize the mean square error of the quantizer. Also determine the corresponding minimal mean square error.
    c. Which codebook is better? Are the necessary conditions that you stated sufficient to guarantee quantizer optimality? Why?



(a) (b) (c)

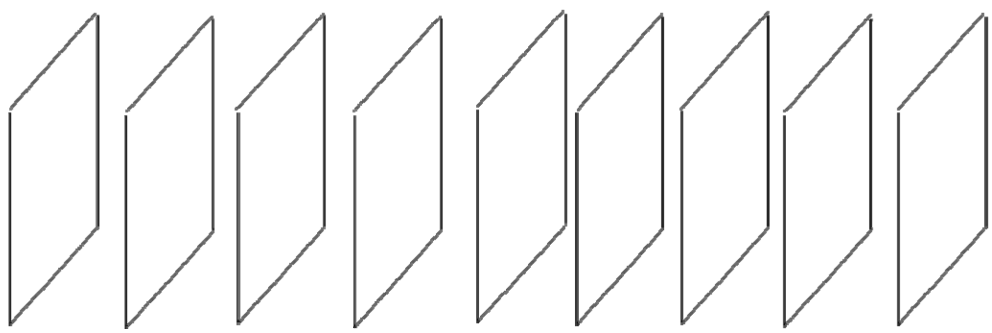
5. (10 pt) Consider the following method for coding a video, which codes two adjacent frames at a time. For each pixel location, it takes the pixel f1 in frame n and its corresponding pixel f2 in frame n-1, applies a 2-pt unitary transform, to generate two coefficients t1 and t2, quantize and code the coefficients. Assume that each pixel has zero mean and variance σ_s^2 and that the correlation coefficient between two corresponding pixels in two adjacent frames is ρ . Furthermore, assume that the operational rate-distortion function for coding each coefficient can be expressed as $D(R) = \varepsilon^2 \sigma^2 2^{-\alpha R}$ where σ^2 is the variance of the coefficient. (a) Determine the optimal transform basis vectors; (b) Determine the variances of the coefficients; (c) Suppose we use optimal bit allocation among the two coefficients, what would be the minimal coding distortion (in terms of mean squared error) for the reconstructed pixels when the average bit rate is R (bits/pixel).

6. (10 pt) Now consider another coding method, which predicts a pixel *f1* in the current frame from the corresponding pixel *f2* in the previous frame using *fp= a f2*, and quantize and code the prediction error *e= f1 –fp*. Assume the video signal follows the same statistics as stated in Prob. 5 and that the rate-distortion function for coding the prediction error can also be expressed as $D(R) = \varepsilon^2 \sigma^2 2^{-\alpha R}$ where σ^2 is the variance of the prediction error. (a) Determine the optimal value for *a* that minimizes the mean squared prediction error; (b) Determine the variances of the prediction error; (c) What would be the minimal coding distortion (in terms of mean squared error) for the reconstructed pixels when the average bit rate is R (bits/pixel); (d) How does this method compare with the one in Prob. 5 in terms of coding efficiency? Specifically, when the bit rate is the same, which one gives lower reconstruction error ?

7. (10pt) Be brief in your answers to the following questions.

   a. In video coding, a frame is often coded as either an I-frame, a P-frame, or a B-frame. Explain what does each mean. Also rank these modes in terms of coding efficiency and complexity.

   b. A video coder often divides successive frames into separate Groups of Pictures, and each GOP contains an I-frame, some P-frames and some B-frames. (a) For the GOP structure illustrated below, what is the encoding order? (c) What are some of the benefits of using the GOP structure? (list 2) What are some of the problems? (d) Describe how do you achieve temporal scalability with this GOP structure. How many layers you could generate? Which frames constitute the base layer, and which frames constitute each of the subsequent layers?



Choose either Prob. 8 or Prob. 9. Only one will be graded.

8. (15 pt) Consider the following 2 level HBMA algorithm. At the top level, you do integer-pel search with a search range so that the equivalent search range at the lowest level (same as original image) is equal to R. At the bottom level, you do half pel search over a search range of S (S<R), centered at the solution from the top level. You use the same block size of BxB at both levels. Write a matlab code for performing this algorithm. Your program should have the following syntax:

   [mvx,mvy,pimg]=HBMA(img1,img2,R,S,B,width,height)

   The input variables are

   img1: the anchor image;
   img2: the target image;
   R: desired search range at the original resolution.
   S: search range at the bottom level.
   B: block size is BxB;
   Width, height: the horizontal and vertical dimension of img1 and img2. For simplicity, assume width, width/2, height, height/2 are all integer multiples of the block size B.

   The output variables are:

   mvx,mvy: the images storing the horizontal and vertical components of the estimated motion field, respectively;
   pimg: the predicted image for the anchor frame using the estimated motion field.

   Note that if you need to use some additional image arrays within the program, you should properly allocate its memory. You may also define any function that you can call from your program.

9. (15pt) Consider the coding of a B-frame. Each 8x8 block in the frame is either coded directly (i.e. predicted block is a constant block with values equal to 128), predicted from the corresponding block in the past frame using motion compensation, or predicted from the average of the corresponding block in the past frame and the corresponding block in the next previously coded frame, using bidirectional motion compensation . The coder chooses the mode that has the lowest prediction error (in terms of sum of absolute difference). Then the prediction error is coded using a transform coder, which transforms the error block using 8x8 DCT, uniformly quantizes each DCT coefficient using a specified quantization stepsize (assuming each coefficient value has a symmetric distribution around 0). The coding mode (mode=0 if coded directly, mode=1 if predicted from the previous frame, mode=2 if predicted using both previous and next frames), the motion vector, and the quantized DCT coefficient indices are coded using an entropy coding method. Write a Matlab code that implements the coding of a frame. Assume that the programs for motion estimation and entropy coding are given (as explained below).
   Your program should have the following syntax:

function [QF]=Encode(F, F1, F2, width,height, mhmax,mvmax, QS, outfile)

where
F: the frame to be coded.
F1: the previous frame, previously decoded.
F2: the future frame, previously decoded.
QF: the reconstructed frame for the current frame.
Width and Height: the width and height of a frame and assume both the width and height are dividable by 8.
mhmax,mvmax: the search range in the horizontal and vertical motions, respectively. That is, you search over a range of [-mhmax, mhmax] for the horizontal displacement, and [-mvmax, mvmax] for the vertical displacement.
Outfile: the file pointer to which to write the bitstream.
QS: the quantization stepsize for DCT coefficients

Your program can call the following functions as well as other MATLAB functions and functions defined by yourself.

function [mvh, mvv,PredictedBlock]=MotionEstimation(Block,RefFrame,h,v,mhmax,mvmax):
find the best matching block for a given 8x8 block (Block) with the top left pixel at position (h,v) in RefFrame, [mvh, mvv] are the returned motion vector components, and PredictedBlock is the best matching block. mhmax and mvmax indicate the search range for horizontal and vertical motion. Note that you don't need to write this function yourself.

function EntropyCoding(mode, mvh1,mvv1, mvh2,mvv2, quantDCTblock, outfile)
Code mode info given by "mode", the motion vector given by "mvh1,mvv1" and "mvh2,mvv2", and the indices of the quantized DCT coefficients of the prediction error block, given by a matrix "quantDCTblock", and write the resulting bits into a file (outfile). When mode=0, the motion vectors will be ignored. When mode=1, ""mvh1,mvv1" represent the motion vector for the previous frame. Note that you don't need to write this function yourself.