Polytechnic University, Dept. Electrical and Computer Engineering
EL612 --- Video Processing, S06 (Prof. Yao Wang)
Final Exam, 4/18/2007, 3:35-5:50
One sheet of Notes (double sided) Allowed

---

1. (15 pt) Consider coding a 2-D random vector that is uniformly distributed over the region illustrated in Fig. 1(a). Suppose you want to design a codebook with 4 codewords. Figure 1(b),1(c),1(d) illustrate three possible codebooks with their corresponding region partitions.
   a. Which codebook will minimize the mean square quantization error?
   b. Determine the x- and y- coordinate of each codeword in your chosen codebook that will minimize the mean square error.
   c. Do the other two codebooks satisfy the necessary condition for minimizing the mean square error?

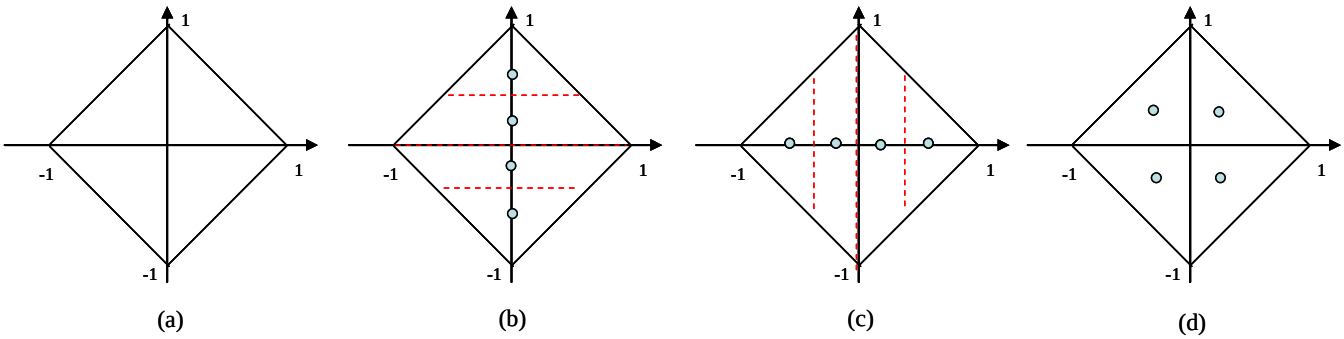


Figure 1

Hint: you should make use of symmetry in deriving your solution.

Solution:

a) (d) will have the minimal distortion, as the largest distance of a point in each partition to its codeword is smaller in (d).
b) The codeword position of (d) can be determined using the centroid condition, which says the codeword position in one partition is the center of the mass of the partition. Let us call the codeword in the upper left triangle (B1) (x1,y1), then

$$x1 = \int_0^1 \int_0^{1-y} x^* \, f(x, y \mid B1)\,dxdy = \int_0^1 \int_0^{1-y} x^* (2)\,dxdy = \int_0^1 (1-y)^2\,dy = 1/3$$

By symmetry, we know y1=1/3. The other codewords are (-1/3,1/3), (1/3,-1/3), (-1/3,-1/3).

Alternatively, you can determine x1 and y1 by minimizing the MSE in the upper left triangle:

$$D1 = \int_0^1 \int_0^{1-y} \left((x-x1)^2 + (y-y1)^2\right) f(x, y \mid B1)\,dxdy$$

$$\frac{\partial D1}{\partial x1} = 0, \frac{\partial D1}{\partial y1} = 0$$

This will give you the same answer.

c) The other two codewords satisfy both the nearest neighbor and centroid conditions, but they are only locally optimal.

2. (10 pt) Consider the following predictive coding method (see Fig. 2). A sample in frame n, $F = f_n(x, y)$ is predicted from its two neighboring pixels in the same frame $A = f_n(x-1, y)$ and $B = f_n(x, y-1)$ and a pixel in a previous frame $C = f_{n-1}(x, y)$ and following frame $D = f_{n+1}(x, y)$, using the linear predictor: $F = aA + bB + cC + dD$.
Suppose the correlations between these samples
are $E\{FA\} = E\{FB\} = \rho_s \sigma^2$, $E\{FC\} = E\{FD\} = \rho_t \sigma^2$, $E\{CD\} = \rho_t{}^2 \sigma^2$ and all other samples are uncorrelated. Find

the optimal values for predictor coefficients a, b, c, d that will minimize the mean square prediction error, and the corresponding minimal prediction error. What is the coding gain compared to code each sample directly?

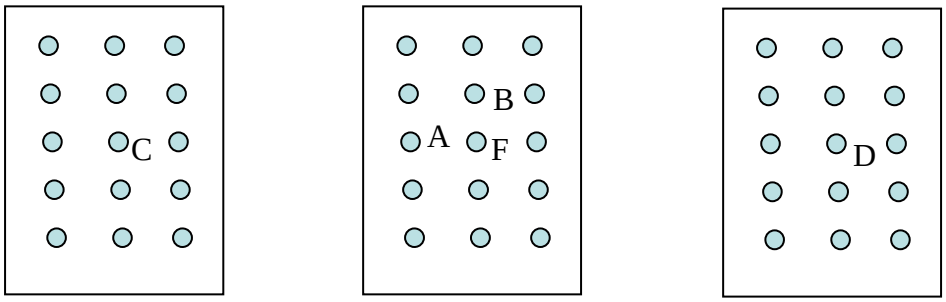


Figure 2

The coefficients satisfy the following equation:

$$\begin{bmatrix} R(A,A) & R(A,B) & R(A,C) & R(A,D) \\ R(B,A) & R(B,B) & R(B,C) & R(B,D) \\ R(C,A) & R(C,B) & R(C,C) & R(C,D) \\ R(D,A) & R(D,B) & R(D,C) & R(D,D) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} R(A,F) \\ R(B,F) \\ R(C,F) \\ R(D,F) \end{bmatrix}$$

Using the provided correlations, we get

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \rho_t^2 \\ 0 & 0 & \rho_t^2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \rho_s \\ \rho_s \\ \rho_t \\ \rho_t \end{bmatrix}$$

The first two equations give $a = b = \rho_s$.

The last two equation is a simple 2x2 equation. Although you may try to find the inverse matrix, a simpler way is to make use of symmetry and recognizing that $c = d$. Then you can get immediatly, $c = d = \rho_t / (1+\rho_t^2)$.

The minimal prediction error with the optimal predictor is

$$\sigma_p^2 = R(F,F) - [R(F,A), R(F,B), R(F,C), R(F,D)] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \sigma^2 - \sigma^2 [\rho_s, \rho_s, \rho_t, \rho_t] \begin{bmatrix} \rho_s \\ \rho_s \\ \rho_t / (1+\rho_t^2) \\ \rho_t / (1+\rho_t^2) \end{bmatrix}$$

$$= \sigma^2 \left(1 - 2\rho_s^2 - \frac{2\rho_t^2}{1+\rho_t^2} \right)$$

The coding gain is

$$G_{DPCM} = \frac{\sigma^2}{\sigma_p^2} = \frac{1}{\sigma^2 \left(1 - 2\rho_s^2 - \frac{2\rho_t^2}{1+\rho_t^2} \right)}$$

3. (15 pt) Consider applying transform coding to every two horizontally adjacent pixels in an image. Suppose every pixel has the same variance σ^2 and the correlation between two adjacent pixels is ρ. (a) Determine the Karhunen Loeve transform basis vectors. (b) Determine the variance of the transformed coefficients. (c) If the total number of bits for the two coefficients is *R*, what is the optimal bit allocation to each coefficient? (d) What is the coding gain over direct coding of individual pixels? Express your results in terms of given variables.

Solution:

(a) the covariance matrix of the two samples is

$$C = \sigma^2 \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

To determine the eigen values, we set

$$\det(C - \lambda I) = \det\begin{bmatrix} \sigma^2 - \lambda & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 - \lambda \end{bmatrix} = \left(\sigma^2 - \lambda\right)^2 - \left(\rho\sigma^2\right) = 0$$

$$\left(\sigma^2 - \lambda\right) = \pm\left(\rho\sigma^2\right)$$

$$\lambda_1 = \sigma^2(1+\rho); \lambda_2 = \sigma^2(1-\rho);$$

To determine the basis vector associated with each eigen value, we solve

$$\left(C - \lambda_k I\right)\phi_k = 0$$

With first eigenvalue, we get

$$\begin{bmatrix} -\rho\sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & -\rho\sigma^2 \end{bmatrix}\phi = 0 \Rightarrow \phi = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Similarly, for the second eigenvalute, we get

$$\begin{bmatrix} \rho\sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \rho\sigma^2 \end{bmatrix}\phi = 0 \Rightarrow \phi = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

(b) the coefficient variances are the same as the eigenvalues:

$$\sigma_{t,1}{}^2 = \sigma^2(1+\rho); \sigma_{t,2}{}^2 = \sigma^2(1-\rho);$$

(c) The optimal bit allocation is

$$R_k = R + \frac{1}{2}\log\frac{\varepsilon_k^2\sigma_{t,k}^2}{\left(\prod_k \varepsilon_k^2\sigma_{t,k}^2\right)^{1/2}}$$

$$\left(\prod_k \varepsilon_k^2\sigma_{t,k}^2\right)^{1/2} = \sigma^2\sqrt{\varepsilon_1^2\varepsilon_2^2(1-\rho)(1+\rho)}$$

$$R_1 = R + \frac{1}{2}\log\frac{\varepsilon_1^2\sigma^2(1+\rho)}{\sigma^2\sqrt{\varepsilon_1^2\varepsilon_2^2(1-\rho)(1+\rho)}} = R + \frac{1}{2}\log\frac{\varepsilon_1\sqrt{(1+\rho)}}{\varepsilon_2\sqrt{(1-\rho)}}$$

$$R_2 = R + \frac{1}{2}\log\frac{\varepsilon_2^2\sigma^2(1-\rho)}{\sigma^2\sqrt{\varepsilon_1^2\varepsilon_2^2(1-\rho)(1+\rho)}} = R + \frac{1}{2}\log\frac{\varepsilon_2\sqrt{(1-\rho)}}{\varepsilon_1\sqrt{(1+\rho)}}$$

(d) The coding gain is

$$G_{TC} = \frac{D_{PCM}}{D_{TC}} = \frac{\varepsilon^2 \sigma^2}{\left(\prod_k \varepsilon_k^2 \sigma_{t,k}^2 \right)^{1/2}} = \frac{\varepsilon^2}{\sqrt{\varepsilon_1^2 \varepsilon_2^2 (1-\rho^2)}}$$

It is OK to assume the factor \epsilon are all the same to simplify your solution.

4. (15 pt) Consider a temporal scalable coder. The base layer contains all even frames: frame n (n=even) is predicted from frame n-2, and the prediction error is coded. The enhancement layer contains remaining odd frames: frame n (n=odd) is predicted from the frame n-1 decoded from the base layer, and the prediction error is coded. See Fig. 4. For simplicity, we do not consider motion. The base layer predictor is $f_{b,n}(x,y) = f_{n-2}(x,y)$ and the enhancement layer predictor is $f_{e,n}(x,y) = f_{n-1}(x,y)$. (In reality, the prediction should be based on decoded values. But for ease of analysis, let us assume the prediction is based on the original values). Assume the corresponding pixels in frame n and frame n-1 have correlation $E\{f_n(x,y) f_{n-1}(x,y)\} = \rho\ \sigma^2$, and the corresponding pixels in frame n and frame n-2 have correlation $E\{f_n(x,y) f_{n-2}(x,y)\} = \rho^2\sigma^2$. Also assume the rate-distortion function of a predictive coder can be represented as $D(R) = \varepsilon^2 {\sigma_p}^2 2^{-\alpha R}$, where ${\sigma_p}^2$ is the prediction error variance. Assume the base layer and enhancement layer are coded with R_b and R_e bits/pixel respectively. Determine the distortion (in terms of mean square error per pixel) when only the base layer is decoded, and the distortion when both the base and enhancement layer are decoded. Compare the distortion of this coder (including both base and enhancement layers) with a single layer coder that codes every frame and predicts each frame from a previous frame, using the same average rate of $R = (R_b + R_e)/2$.

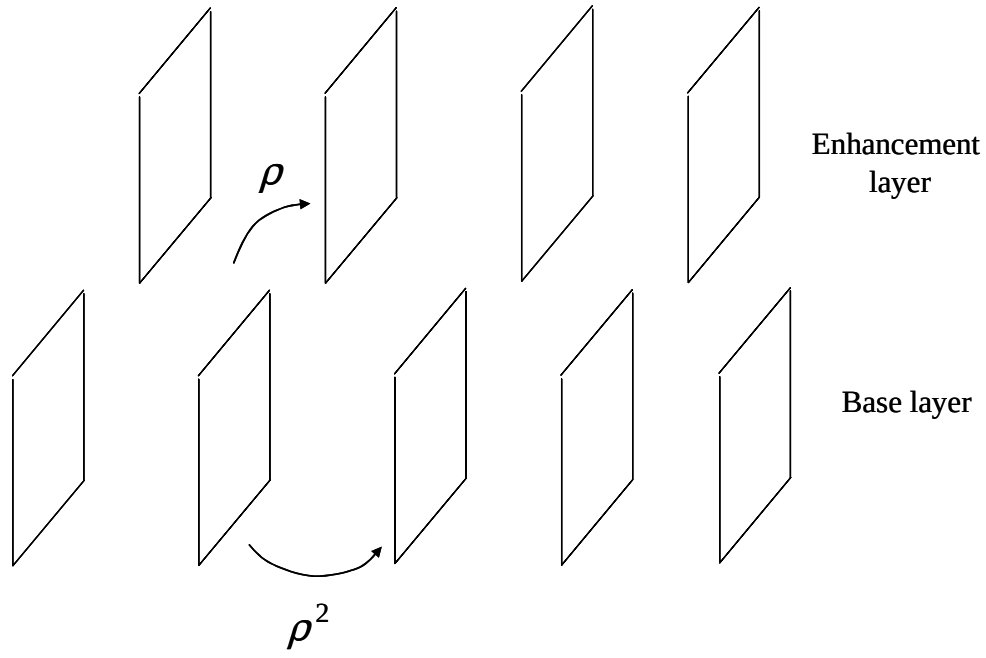


Figure 4

Solution:

Base layer prediction error $\sigma_{p,b}^2 = E\{(f_n - f_{n-2})^2\} = E\{(f_n)^2 + (f_{n-2})^2 + 2 f_n f_{n-2}\} = 2(1-\rho^2)\sigma^2$

Rate distortion function for base layer encoding: $D(R_b) = \varepsilon^2 \sigma_{p,b}^2 2^{-\alpha R_b} = 2\varepsilon^2(1-\rho^2)\sigma^2 2^{-\alpha R_b}$

Enhancement layer prediction error: $\sigma_{p,e}^2 = E\{(f_n - f_{n-1})^2\} = E\{(f_n)^2 + (f_{n-1})^2 + 2 f_n f_{n-1}\} = 2(1-\rho)\sigma^2$

Rate distortion function for enhancment layer encoding: $D(R_e) = \varepsilon^2 \sigma_{p,e}^2 2^{-\alpha R_e} = 2\varepsilon^2(1-\rho)\sigma^2 2^{-\alpha R_e}$

Distortion with base and enhancement layer $D_{be}(R_b, R_e) = 2\varepsilon^2\sigma^2\left((1-\rho^2)2^{-\alpha R_b} + (1-\rho)2^{-\alpha R_e}\right)$

Single layer prediction error: $\sigma_{p,s}^2 = E\left\{\left(f_n - f_{n-1}\right)^2\right\} = E\left\{\left(f_n\right)^2 + \left(f_{n-1}\right)^2 + 2 f_n f_{n-1}\right\} = 2(1-\rho)\sigma^2$

Distortion of single layer coding: $D_s(R) = 2\varepsilon^2\sigma^2\left((1-\rho)2^{-\alpha R}\right)$

To compare single layer distortion at R=(R_b+R_e)/2 with the two layer coder, we compare

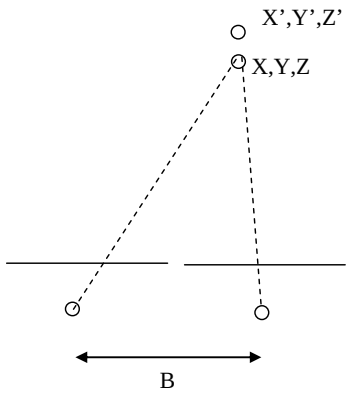
$D_s\left(\left(R_b + R_e\right)/2\right) = 2\varepsilon^2\sigma^2\left((1-\rho)2^{-\alpha\left(R_b+R_e\right)/2}\right)$ and $D_{be}\left(R_b, R_e\right) = 2\varepsilon^2\sigma^2\left((1-\rho^2)2^{-\alpha R_b} + (1-\rho)2^{-\alpha R_e}\right)$

Note that when $R_b >=R_e, D_s <= second term of D_be. D_be ‘s first term is positive. Therefore D_s always <=D_be when R_b is relatively large. In our derivation, we have ignored quantization error in the base layer coding, when considering the prediction in the enhancementt layer, this essentially assumes that the base layer is coded accurately, or R_b is relatively large.

5. (15 pt) Consider stereo imaging using parallel camera configuration with a baseline distance of B. Assuming a 3D point with coordinate *(X,Y,Z)* is projected into left and right image planes with coordinates (x_l, y_l) and (x_r, y_r).

   a. Derive the formulae for recovering *X,Y, Z* from x_l, y_l, x_r, y_r.

   b. Suppose another point has the 3D coordinate $X' = X, Y' = Y, Z' = Z + \Delta Z$. Relate the difference in disparity for these two points Δd with ΔZ. Show that when \delta Z is small, the disparity difference Δd is linearly related to ΔZ.



(a)

$$X_l = X + B/2; X_r = X - B/2; Y_l = Y_r = Y; Z_l = Z_r = Z;$$

$$x_l = F\frac{X + B/2}{Z}; x_r = F\frac{X - B/2}{Z}; y_l = y_r = F\frac{Y}{Z}$$

$$dx = x_l - x_r = \frac{FB}{Z}; \frac{Z}{F} = \frac{B}{dx}$$

$$Z = \frac{FB}{dx}; Y = y_l\frac{Z}{F} = y_l\frac{B}{dx};$$

$$x_l + x_r = 2F\frac{X}{Z} -> X = \frac{x_l + x_r}{2}\frac{Z}{F} = \frac{x_l + x_r}{2}\frac{B}{dx}$$

(b)

$$dx' = \frac{FB}{Z',}$$

$$dx' - dx = \frac{FB}{Z',} - \frac{FB}{Z,} = \frac{FB\Delta}{(Z + \Delta)Z} \approx \frac{FB\Delta}{Z^2}$$

Where \delta is small, the above relation is linearly related to \delta, as shown above.

6. (10pt)
    a. List one major difference between H.261 and H.263 video coding standard, one major difference between MPEG-1 and MPEG-2standards (video part only), and one major difference between H.261 and MPEG-1 video. The difference may be in video formats that can be handled, the motion estimation methods, the use of deblocking filtering, the prediction methods, the intended applications, etc.
    b. Describe some of the new coding techniques adopted in the H.264 standard. You should point out at least 3 new features.

a.
H261 vs H263: H261 uses integer pel motion search only, H263 allows half-pel motion search. H.261 uses explicit loop filtering; H263 does not, because half-pel motion compensation accomplishes filtering implicitly.

MPEG-1 and MPEG-2: MPEG-1 is for progressive sequences, MPEG-2 is for both progressive and interlaced sequences. MPEG-2 has many motion estimation modes and DCT modes that are designed to code the interlaced sequences more efficiently. Also, MPEG-2 has different scalability modes, whereas MPEG-1 does not.

H.261 vs. MPEG-1: MPEG standard is developed for video distribution. In order to enable random access, it organize frames into GOPs, with periodic I-frames. H.261 (and H.263) on the other hand is mainly targeted for video telephone/conferencing and only the first frame is coded as I-frame. MPEG-1 uses bidirectional motion estimation but H.261 uses only unidirectional motion estimation.

b. Intra-prediction, interger transform (replacing DCT), explicit deblocking filtering, more sophisticated motion search (up to 1/8 pel accuracy, variable block size down to 4x4, multiple reference frames)

7. (20 pt) Write a matlab code that implements coding of a B-frame in a block-based hybrid video coder (using 16x16 pixel macroblock for motion estimation and 8x8 block size for DCT and quantization and runlength coding). For simplicity, consider the coding of Y-pixels only. For each macroblock, you may decide to code it in either I-Mode (coding a block directly) or BP-Mode (predict from a best matching block in the previous frame), FP-mode (predict from the best matching block in the following frame) or B-Mode (predict from the average of the best matching blocks in both previous and following frames). For the I-mode, you perform DCT on each 8x8 original image block. For the other modes, you perform DCT on the prediction error block. With either mode, you quantize the DCT coefficients using QP and quantization matrix QMatrix.

Denote the frame to be coded by CurrentFrame, the previous frame used for prediction by PrevFrame, the following frame used for prediction by NextFrame. Your program should write the resulting bits for successive macroblocks into a file outfile. You should first write the bit describing the mode. If the mode is I-mode, you would then write the quantized DCT bits for each 8x8 block in the original macroblock. For the other mode, you would then write the bits for motion vector(s), followed by quantized DCT bits for the prediction errors in each 8x8 block. Your program should also compute and save the reconstructed frame in QuantizedFrame. Also use Width and Height to denote the width and height of a frame and assume both the width and height are dividable by 16. Furthermore, assume the following functions are available (i.e. can be called by your matlab code). Your program can call these functions as well as other MATLAB functions and functions defined by yourself.

function [mvh, mvv,PredictedMBlock]=MotionEstimation(MBlock,ReferenceFrame):
finds the best matching block for a given 16x16 macroblock (MBlock) in ReferenceFrame, [mvh, mvv] are the returned motion vector components, and PredictedMBlock is the best matching macroblock.

function [RunSymbols]=RunlengthOrdering(QuantizedDCTIndexBlock)
converts a 8x8 block of quantized DCT indices (QuantizedDCTIndexBlock) into symbols with each symbol including a runlength and a non-zero value.

function [DCTBits]=HuffmanCodingDCT(RunSymbols)
performs Huffman coding on the RunSymbols using default Huffman tables.

function [MVBits]=HuffmanCodingMV(mvh,mvv)
performs Huffman coding on the motion vector using default Huffman tables.

function WriteDCTBits(DCTBits,outfile): writes the bits for each 8x8 block (DCTBits) into a file outfile which stores the coded bits for this frame.

function WriteMVBits(MVBits,outfile): writes the bits for motion vector of a macroblock into a file outfile which stores the coded bits for this frame.

Suggestions:
You may want to define following functions, and call these functions from your main function
function [QuantizedDCTIndexBlock]=quantizeDCT(DCTBlock,QP,QMatrix)
performs quantization on a block of 8x8 DCT coefficients (DCTBlock), with quantization parameter QP and QMatrix, return the quantization indices in QuantizedDCTIndexBlock.

function [QuantizedDCTBlock]=dequantizeDCT(QuantizedDCTIndexBlock,QP,QMatrix)
takes the quantized DCT indices of a block (QuantizedDCTIndexBlock) and applies inverse quantization to obtain quantized DCT coefficients (QuantizedDCTBlock)

Solution:

```
Function QuantizedFrame=BFrameCoding(CurrentFrame,PrevFrame,NextFrame,QP,QMatrix,outfile)

QuantizedFrame=zeros(height,width);
For (r=1:16:height) for (c=1:16:width) %processing each macroblock
        MBlock=CurrentFrame(r:r+15,c:c+15);
%first do backward motion estimation
        [mvh_BP,mvv_BP,PredictedMBlock_BP]=MotionEstimation(MBlock,PrevFrame);
        SAD_BP=sum(sum(abs(MBlock-PredictedMBlock_BP)));
%Then do forward  motion estimation
        [mvh_FP,mvv_FP,PredictedMBlock_FP]=MotionEstimation(MBlock,NextFrame);
        SAD_FP=sum(sum(abs(MBlock-PredictedMBlock_FP)));
%Form the B prediction from BP and FP
        PredictedMBlock_B=( PredictedMBlock_FP+ PredictedMBlock_BP)/2;
        SAD_B=sum(sum(abs(MBlock-PredictedMBlock_B)));

%Calculate intra SAD
        Mean=mean(mean(MBlock));
        SAD_intra=sum(sum(abs(Mblock-Mean)));

        MinErr=min(SAD_FP,SAD_BP,SAD_B,SAD_intra);

        If (MinErr==SAD_FP) mode=0;
        else If (MinErr==SAD_BP) mode =1;
        else If (MinErr==SAD_B) mode =2;
        else If (MinErr==SAD_intra) mode =3;
        end;
        WriteModeBits(mode,outfile);
        If (mode==0)
                [MVBits]=HuffmanCodingMV(mvh_FP,mvv_FP); WriteMVBits(MVBits,outfile);
        PredictedMBlock= PredictedMBlock_BP;
Else If (mode==1)
                [MVBits]=HuffmanCodingMV(mvh_BP,mvv_BP); WriteMVBits(MVBits,outfile);
PredictedMBlock= PredictedMBlock_FP;
        Else If (mode==2)
                [MVBits]=HuffmanCodingMV(mvh_BP,mvv_BP); WriteMVBits(MVBits,outfile);
                [MVBits]=HuffmanCodingMV(mvh_FP,mvv_FP); WriteMVBits(MVBits,outfile);
                PredictedMBlock= PredictedMBlock_B;
                %note you need to code both motion vectors in B-mode
        else
                PredictedMBlock=zeros(16,16);
        end;
        ErrBlock=MBlock-PredictedMBlock;
        for (i=1:8:9) for (j=1:8:9)  %for each 8x8 block in ErrBlock, do following
                Block=ErrBlock(i:i+7, j:j+7);
                DCTBlock=dct2(Block);
                [QuantizedDCTIndexBlock]=quantizeDCT(DCTBlock,QP,QMatrix); %defined below
                RunSymbols=RunlengthOrdering(QuantizedDCTIndexBlock);
                [DCTBits]=HuffmanCodingDCT(RunSymbols);
                WriteDCTBits(DCTBits,outfile);

                QuantizedDCTBlock=dequantizeDCT(QuantizedDCTIndexBlock,QP,QMatrix); %defined
below
```

```
                QuantizedBlock=idct2(QuantizedDCTBlock);
                QuantizedErrBlock(i:i+7,j:j+7)=QuantizedBlock;
        end;end
        QuantizedMBlock=PredictedMBlock+QuantizedErrBlock;


        QuantizdFrame(r:r+15,c:c+15)=QuantizedMBlock;
end;end

%definition of functions:
function [QuantizedDCTIndexBlock]=quantizeDCT(DCTBlock,QP,QMatrix)
QuantizedDCTIndexBlock=floor((DCTBlock+QMatrix*QP/2) ./(QMatrix*QP));

Function QuantizedDCTBlock=dequantizeDCT(QuantizedDCTIndexBlock,QP,QMatrix)
QuantizedDCTBlock=QuantizedDCTIndexBlock.*(QMatrix*QP);
```