1.a. I-~~mode~~ frame: Intra-~~pred~~ Frame. Is coded without reference to other frames.

P-frame: Predicted Frame. Coded with reference to the previous frame.

B-frame: Also a predicted frame. Coded with reference to both previous and future frame.

Coding efficiency: I-frame << P-frame << B-frame

$\longrightarrow$ ~~less~~ more efficient

Complexity: I-frame << P-frame << B-frame.

$\longrightarrow$ more complex

Encoding delay: I-frame << P-frame << B-frame.

$\longrightarrow$ more delay

b. 1. With layer coder, base layer and enhancement layer are both transmitted. But if ~~enhancement~~ base layer is lost or in error, enhancement layer has no use by itself.

~~2. Since P-frames are predicted from I-frames and further used to predict B-frame, error in P/I-frame would propogate in time.~~

3. Motion ~~estimation~~ compensation would makes error in reference frame propogate ~~in~~ spatially/in time

2. Variable length coding makes subsequent bits of a errored bit ~~is~~ non-decodable.

c. 1. FEC: Forward Error Correction/Detection

2. ARQ: Automatic Retransmission Request.

3. Error Resilient encoding: Add redundancy to video bitstreams to assist decoder recovery.

4. Insert more I-frames.

5. Paketizing and slicing.

d. Spatial Scalability: In terms of picture size. User with lower ~~bitrate~~ bandwidth can see small-size-frames, higher ~~bitrate~~ bandwidth => bigger picture.

Temporal ~~Sd~~ Scalability: Frame rate. Higher ~~bit~~ bandwidth more frames can be seen.

Amplitude Scalability: (SNR). Higher bandwidth users see the frames which are encoded ~~with~~ using smaller Quantization stepsize. Users with low bandwidth see frames encoded using large QP. How ? -2

e. 1. Size of the object. objects further away looks smaller.

2. Occlusion. If A is occluded by B, A is ~~more~~ farther away then B is.

3. Parallax. When moving, objects further away moves much slower than objects near us.

f. 1. Basis should be nearly decorrelating.

2. High energy compaction: few of the basis vectors contain big portion of the energy.

3. Easy to compute.

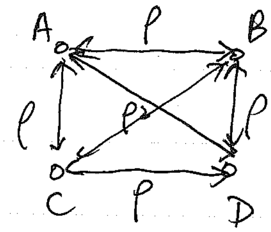
4. Separable. Can ~~co~~ perform ~~code and decode~~ inverse transform and transform.

d: - spatial scalability can be implemented by down sampling original frame to smaller frames. Code small frames ~~as~~ in base layer. Interpolate base layer to original size, code interpolation error as enhancement layer.

- Temporal scalability can be implemented by temporal down sampling. Base layer = lower frame rate, enh ~~to~~ layer = skipped frames

- Amplitude scalability by multiple quantizers.

2. a. $[U] = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$



b. $[C]_s = E\left\{ [A \quad B \quad C \quad D]^* \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \right\}$

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

c. $[C]_t = [V][C]_s[V]^H \qquad [V] = [U]^H = [U]$

$$[V][C]_s = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & \rho & \rho & \rho^2 \\ \rho & 1 & \rho^2 & \rho \\ \rho & \rho^2 & 1 & \rho \\ \rho^2 & \rho & \rho & 1 \end{bmatrix} \sigma^2$$

$$= \frac{\sigma^2}{2}\begin{bmatrix} (\rho+1)^2 & (\rho+1)^2 & (\rho+1)^2 & (\rho+1)^2 \\ (1-\rho^2) & (1-\rho^2) & (\rho^2-1) & (\rho^2-1) \\ (1-\rho^2) & (\rho^2-1) & (1-\rho^2) & (\rho^2-1) \\ (\rho-1)^2 & -(\rho-1)^2 & -(\rho-1)^2 & (\rho-1)^2 \end{bmatrix}$$

$$[V][C]_s[V]^H = \frac{\sigma^2}{2}\begin{bmatrix} (\rho+1)^2 & (\rho+1)^2 & (\rho+1)^2 & (\rho+1)^2 \\ (1-\rho^2) & (1-\rho^2) & (\rho^2-1) & (\rho^2-1) \\ (1-\rho^2) & (\rho^2-1) & (1-\rho^2) & (\rho^2-1) \\ (\rho-1)^2 & -(\rho-1)^2 & -(\rho-1)^2 & (\rho-1)^2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \frac{1}{2}$$

$$= \frac{\sigma^2}{4}\begin{bmatrix} 4(\rho+1)^2 & 0 & 0 & 0 \\ 0 & 4(1-\rho^2) & 0 & 0 \\ 0 & 0 & 4(1-\rho^2) & 0 \\ 0 & 0 & 0 & 4(\rho-1)^2 \end{bmatrix}$$

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

$$\left(\prod_k \varepsilon_{t,k}^2 \sigma_{t,k}^2\right)^{1/4} = \left(\varepsilon_{t,1}^2 \varepsilon_{t,2}^2 \varepsilon_{t,3}^2 \varepsilon_{t,4}^2\right)^{1/4} \cdot \left((\rho+1)^2(1-\rho^2)(1-\rho^2)(\rho-1)^2\right)^{1/4} \sigma^2$$

$$= \left(\prod_{k=1} \varepsilon_{t,k}^2\right)^{1/4} \cdot (1-\rho^2)\sigma^2$$

$$R_1 = R + \frac{1}{2}\log_2 \frac{\varepsilon_{t,1}^2(\rho+1)^2}{\left(\prod_k \varepsilon_{t,k}^2\right)^{1/4}(1-\rho^2)} = R + \frac{1}{2}\log_2 \frac{\varepsilon_{t,1}^2(\rho+1)}{\left(\prod_k \varepsilon_{t,k}^2\right)^{1/4}(1-\rho)} = R + \frac{1}{2}\log_2 \frac{\sigma_{t,k}^2}{(1-\rho^2)\sigma^2}$$

$$\cancel{R_2} R_2 = R + \frac{1}{2}\log_2 \frac{\varepsilon_{t,2}^2}{\left(\prod_k \varepsilon_{t,k}^2\right)^{1/4}} \qquad R_3 = R + \frac{1}{2}\log_2 \frac{\varepsilon_{t,3}^2}{\left(\prod_k \varepsilon_{t,k}^2\right)^{1/4}}$$

$$R_4 = R + \frac{1}{2}\log_2 \frac{\varepsilon_{t,4}^2 (1-\rho)}{\left(\prod_k \varepsilon_{t,k}^2\right)^{1/4}(1+\rho)}$$

Average distortion: $D_t = \frac{1}{4}\left(\varepsilon_{t,1}^2(\rho+1)^2\sigma^2 \cdot 2^{-2R_1} + \varepsilon_{t,2}^2(1-\rho^2)\sigma^2 \cdot 2^{-2R_2} + \varepsilon_{t,3}^2(1-\rho^2)\sigma^2 \cdot 2^{-2R_3} + \varepsilon_{t,4}^2(1-\rho)^2\sigma^2 \cdot 2^{-2R_4}\right)$

$$= 2^{-2R} \cdot \left(\prod_k \varepsilon_{t,k}^2\right)^{1/4}(1-\rho^2)\sigma^2$$

e. $D_s = \varepsilon_s^2 \sigma^2 \cdot 2^{-2R}$

f. $$G_{TC} = \frac{D_s}{D_t} = \frac{\varepsilon_s^2}{\left(\prod_{k=1}^4 \varepsilon_{t,k}^2\right)^{1/4}} \frac{\sigma^2}{(1-\rho^2)\sigma^2} = \frac{\varepsilon_s^2}{\left(\prod_{k=1}^4 \varepsilon_{t,k}^2\right)^{1/4}} \frac{1}{(1-\rho^2)} = \frac{1}{1-\rho^2}$$

Could have assumed $\varepsilon_s = \varepsilon_{t,k} = \varepsilon$

3. a. $E\{f_n(x,y) f_{n-1}(x,y)\} = \rho\sigma^2$

$$\cancel{E\{(f_n(x,y) - f_{n-1}(x,y))^2\} = E\{f_n^2(x,y) + f_{n-1}^2(x,y) - 2 f_{n-1}(x,y) f_n(x,y)\}}$$

$$\cancel{= 2(1-\rho)\sigma^2}$$

$$[R(1,1)][a] = [R(0,1)]$$

$$\sigma^2[1][a] = \rho\sigma^2$$

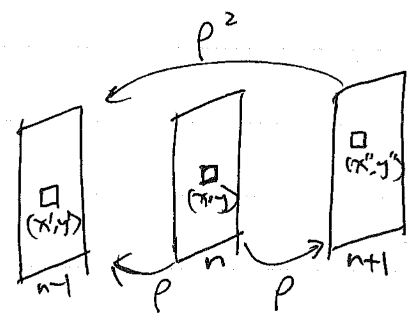
$$a = \rho$$

$$\sigma_p^2 = R(0,0) - R(0,1)\cdot a = (1-\rho^2)\sigma^2$$

$$\begin{bmatrix} R(1,1) & R(2,1) \\ R(1,2) & R(2,2) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} R(0,1) \\ R(0,2) \end{bmatrix}$$

b.

$$\sigma^2 \begin{bmatrix} 1 & \rho^2 \\ \rho^2 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho \end{bmatrix} \sigma^2$$



$$b_1 = b_2 = \frac{\rho}{1+\rho^2}$$

$$\sigma_B^2 = \sigma^2 \left(1 - [\rho \quad \rho] \begin{bmatrix} \frac{\rho}{1+\rho^2} \\ \frac{\rho}{1+\rho^2} \end{bmatrix}\right)$$

$$= \sigma^2 \cdot \frac{1-\rho^2}{1+\rho^2}$$

c. For the P-mode: $D_0 = \varepsilon^2(1-\rho^2)\sigma^2 \cdot 2^{-\alpha R_p}$

$$\Downarrow$$

$$\underline{R_p} = \frac{1}{-\alpha} \cdot \log_2 \frac{D_0}{\varepsilon^2(1-\rho^2)\sigma^2} = \frac{1}{\alpha} \log_2 \frac{\varepsilon^2(1-\rho^2)\sigma^2}{D_0}$$

For the B-mode: $D_0 = \varepsilon^2 \frac{1-\rho^2}{1+\rho^2} \cdot \sigma^2 \cdot 2^{-\alpha R_B}$

$$\Downarrow$$

$$\underline{R_B} = \frac{1}{-\alpha} \cdot \log_2 \frac{D_0(1+\rho^2)}{\varepsilon^2(1-\rho^2)\sigma^2} < R_p$$

$$= \frac{1}{\alpha} \log_2 \frac{\varepsilon^2(1-\rho^2)\sigma^2}{D_0(1+\rho^2)}$$

d. Because B-mode requires extra delay, since it needs future frame to encode current frame. In ~~read~~ real applications, eg. live streaming, video conferencing, ~~video quality~~ user is very sensitive to delay. Thus B-mode is not used.

(Non causal, can use for some frames if future frames are coded first [illegible] B)

-1

4. a. $x_l = F \cdot \frac{X+B/2}{Z}$ $\quad x_r = F \cdot \frac{X-B/2}{Z}$

$\cancel{x_l + x_r = F \cdot \frac{X}{Z} \rightarrow X = \frac{(x_l+x_r)\cdot Z}{F}}$

$\cancel{Y = \frac{y \cdot Z}{F}}$

$x_l - x_r = \frac{FB}{Z}$, $\underline{Z} = \frac{F \cdot B}{x_l - x_r}$

-1

$x_l + x_r = 2F \cdot \frac{X}{Z}$ $\quad \underline{X} = \frac{(x_l+x_r)\cdot Z}{2F} = \frac{(x_l+x_r) \cdot F \cdot B}{2(x_l-x_r)\cdot F} = \frac{x_l+x_r}{2(x_l-x_r)} \cdot B$

$\underline{Y} = y \cdot \frac{Z}{F} = y \frac{B}{x_l - x_r}$

b. $x_i = F \cdot \frac{X + B/4}{Z}$, from (a) we have expression for X and Z, so:

$x_i = F \cdot \frac{\frac{x_l+x_r}{2(x_l-x_r)} \cdot B + \frac{B}{4}}{\frac{F \cdot B}{x_l - x_r}} = \frac{x_l+x_r}{2} + \frac{1}{4}\cdot(x_l - x_r) = \cancel{\frac{5}{4} x_l + \frac{3}{4} x_r}$

$= \frac{3}{4} x_l + \frac{1}{4} x_r$

$y_i = \frac{FY}{Z} = y$

-1

c. 1. Divide the image into block, for each block, assume it's a flat surface, so every point in it has same disparity.

2. For each block k in the right image, minize the disparity (matching) error:

$E = \sum_{(x_r, y_r)} (f_r(x_r, y_r) - f_l(x_l, y_l))^2 = \sum_{(x_r, y_r) \in k} (f_r(x_r, y_r) - f_l(x_r + d_k(x_r, y_r), y_r))^2$

Get the disparity $\underline{d_k(x_r, y_r)}$.

-3

3. For every block k in the intermediate image:

For each pixel (x_i, y_i), compute corresponding x_r using:

$x_i = \frac{5}{4} x_l + \frac{3}{4} x_r = \frac{5}{4}(x_r + d_k(x_r, y_r)) + \frac{3}{4} x_r$. Then set $f(x_i, y_i) = f(x_r, y_r)$

$f(x_i, y_i) = \frac{1}{4} f_l(x_l, y_l) + \frac{3}{4} f_r(x_r, y_r)$

```
5.  function [QF] = Encode(F, PF, width, height, mhmax, mvmax, QS, outfile)
QF=zeros(width, height);
    for(r=1:8:height) for(c=1:8:width)  % for each 8x8 block
            Block= F(r:r+7, c:c+7);
        % first do backward motion estimation
            [mvh, mvv, PredictedBlock] = MotionEstimation(Block, PF, width, height,
                                                           mhmax, mvmax);

            SAD_P=sum(sum(abs(Block-PredictedBlock)));
        % code directly.
            SAD_I= sum(sum(abs(Block-128)));
        if SAD_P > SAD_I   mode = 1;  % code directly.
        else mode=0; end;

        ~~If mode==0    % code using predicted frame + error.~~
        % compute DCT coefficients of the error block.
        if mode==0  % predicted
            ErrBlock= Block- PredictedBlock;   mvh=0, mvv=0;
        else  ErrBlock = Block-128;  end;

          DCTBlock= dct2(ErrBlock);
        [QuantizedDCTBlock]= quantizeDCT(DCTBlock, QS);   % defined below
         EntropyCoding(mode, mvh, mvv, QuantizedDCTBlock, outfile);
          QuantizedDCTBlock2 = dequantizeDCT(QuantizedDCTBlock, QS);
    end;  QuantizedDCTBlock3= idct2(QuantizedDCTBlock2);
        if mode==0  QF(r:r+7, c:c+7)=QuantizedDCTBlock3+PredictedBlock;
  end;  else  QF(r:r+7, c:c+7)=QuantizedDCTBlock3+128;
```

Function for the Quantization of DCT coefficients is defined:

```
function [QuantizedDCTBlock]= quantizeDCT(DCTBlock, Qs)
```

Quantized DCTBlock = ~~floor (DCTBlock)~~
= floor (DCTBlock /QS);

-1

need to shift

$$\frac{DCTBlock + QS/2}{QS}$$

~~dequantize DCT~~

dequantize DCT function:

```
function [deQuantizedDCTBlock] = dequantizeDCT (DCTBlock, QS)
        deQuantizedDCTBlock = DCTBlock*Qs;
```