1. (a) From the Fig. 1(b) \[ a = \frac{b}{3} \]
\[ b = \frac{a + l}{2} \]

\[ \Rightarrow \begin{cases} a = \frac{1}{3} \\ b = \frac{2}{3} \end{cases} \]

\[ \text{mse} = \sum_{i} \frac{1}{1} \left[ (x - \frac{1}{3})^2 + y^2 \right] 3 \text{dBoby} = \frac{40}{27} \]

(b) For the codebook configuration in Fig. 1(c), it is equivalent to that in Fig 1(b) if the same value for \( a \) and \( b \) are chosen. 
Because the minimal mean square errors are the same between two codebooks.

(c) The codebook has less mean square error than Fig 1(b) (c).

+5
\[
\begin{align*}
\begin{bmatrix} R(A, A) & R(A, B) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} &= \begin{bmatrix} R(F, A) \\ R(F, B) \end{bmatrix} \\
\begin{bmatrix} R(B, A) & R(B, B) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} &= \begin{bmatrix} R(F, B) \end{bmatrix}
\end{align*}
\]

\[
\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \rho \\ \frac{\rho^2}{2} \end{bmatrix} \Rightarrow \begin{align*}
a &= \frac{\rho - \frac{\rho^2}{2}}{1 - \rho^2} \\
b &= \frac{-\frac{\rho^2}{2}}{1 - \rho^2}
\end{align*}
\]

\[
\sigma^2_p = R(F, F) - \begin{bmatrix} \rho & \frac{\rho^2}{2} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \frac{1 - \frac{\rho^2}{2} + \frac{\rho^2}{2}}{1 - \rho^2} \sigma^2
\]
3. (a) \[ U_{00} = u_0 \cdot u_0^T = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \]
\[ U_{01} = u_0 \cdot u_1^T = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \]
\[ U_{10} = u_1 \cdot u_0^T = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \]
\[ U_{11} = u_1 \cdot u_1^T = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \]

(b) \[ U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{bmatrix} \]

(c) \[ C_8 = \begin{bmatrix} 1 & \rho & \rho^2 & \rho^4 \\ \rho & 1 & \rho^2 & \rho^3 \\ \rho^2 & \rho & 1 & \rho \\ \rho^4 & \rho^3 & \rho^2 & 1 \end{bmatrix} \]
\[ C_8 = V C_8 V^H = V C_8 U^H = \begin{bmatrix} 1+\rho^2 & \frac{1}{2} \rho & \frac{1}{2} \rho & 0 \\ \frac{1}{2} \rho & 1-\rho^2 & 0 & -\frac{1}{2} \rho \\ \frac{1}{2} \rho & 0 & 1-\rho^2 & -\frac{1}{2} \rho \\ 0 & -\frac{1}{2} \rho & -\frac{1}{2} \rho & 1+\rho^2 \end{bmatrix} \]

\[ \sigma_{\text{ex}}^2 = \begin{bmatrix} 1+\rho^2, 1-\rho^2, -1+\rho^2, -1-\rho^2 \end{bmatrix} \]

(d) \[ R^* = R + \frac{1}{2} \log \frac{e^{\frac{\sigma_{\text{ex}}^2}{2} - \frac{e^{\sigma_{\text{ex}}^2}}{2}}}{(2 \pi e)^{\frac{3}{2}} \sigma_{\text{ex}}^2} = R + \frac{1}{2} \log \frac{e^{\frac{\sigma_{\text{ex}}^2}{2} - \frac{e^{\sigma_{\text{ex}}^2}}{2}}}{(2 \pi e)^{\frac{3}{2}} \sigma_{\text{ex}}^2} \]

where \[ \sigma_{\text{ex}}^2 = \begin{bmatrix} 1+\rho^2, 1-\rho^2, -1+\rho^2, -1-\rho^2 \end{bmatrix} \]
4. Solution: Since we will use parallel currents configuration,

\[ X_1 = \frac{F}{Z}(x + \frac{B}{2}) \quad X_2 = \frac{F}{Z}(x - \frac{B}{2}) \]

\[ \Delta X = X_1 - X_2 = \frac{FB}{Z} \]

\[ \Rightarrow Z = \frac{FB}{\Delta X} \]

So \[ \Delta Z = Z_2 - Z_1 \]

\[ = FB \left( \frac{1}{\Delta x_2} - \frac{1}{\Delta x_1} \right) \]

It gives us one way to solve the problem.

Assume we can do automatic segmentation on each image to identify pixels in the two regions.

So step 1. Use segmentation to identify pixels in two regions on each image.

2. We can use the idea about mean squared error.

And call them \( R_{u_1}, R_{u_2}, R_{r_1}, R_{r_2} \)

2. Since the two foreground objects have flat front surface.

So \( \Delta x_1, \Delta x_2 \) can consider as constant.

3. We can use

\[ \text{minimize} \sum_{R_{u_1}} \left[ \frac{f(x_1, y) - f_{R_{u_1}}(x, y)}{\Delta x_1} \right]^2 \]

4. same method for \( R_{u_2}, R_{r_2} \)

\[ \text{minimize} \sum_{R_{u_2}} \left[ \frac{f_{R_{u_2}}(x + \Delta x_2, y) - f_{R_{u_2}}(x, y)}{\Delta x_2} \right]^2 \]

\[ \text{minimize} \sum_{R_{r_1}} \left[ \frac{f_{R_{r_1}}(x + \Delta x_1, y) - f_{R_{r_1}}(x, y)}{\Delta x_1} \right]^2 \]

5. Solve the function:

\[ \Delta Z = FB \left( \frac{1}{\Delta x_2} - \frac{1}{\Delta x_1} \right) \]
1. Layered coding: A coding method to divide bitstream into multiple layers. Decoding only base layers can achieve a low-quality video, while decoding all layers can achieve full quality. Video will be reconstructed to meet the needs of transmission error. Note that the base layers should be protected.

2. Multiple reference frame prediction

3. Adaptive blocking filtering

4. Inter-mode prediction

5. 2-layer DCT transform with variable block sizes

6. The coding efficiency of LC is higher than that of MDC, because LC lends more bit to code, while the cost of one channel is more error resilient than MDC. Incorporating video into different channels reduce the correlation between frames, thus reducing transmission error.
LC with transmission control. 2 channels are good to get better quality.

CHANNEL 1

SVC decoder, playback

CHANNEL 2

Select depending on network cond.

(*) Note that the above diagram omits the Layered encoder part.

6.

(*) Block Diagram.

Real layer code

DCT

Mode Selection between

IDCT

Entropy coding

Llth frame

Reconstruct

i-l th frame

i-th frame

Enhance layer

MV if mode = 1

Entropy coding

(*) the above diagram omits some components in decoder of prediction loop

1b) (Implementation on the proceeding page)
Function spatialCoding(CF, PF, BCF, Gp, G_matrix, output)
[Height, Width] = size(CF);
RF = zeros(Height, Width); IFrame = interpolate(BCF);
for r = 1: 16: Height
  for c = 1: 16: Width
    MBlock = CF(r = r+15, c = c+15);
    Emvh_P = mvh_PF, predictedMBlock = MotionEstimation(MBlock, PF);
    SAD_PF = sum(sum(abs(MBlock - predictedMBlock_PF))); 
    predictedMBlock_BCF = IFrame(r = r+15, c = c+15);
    SAD_BCF = sum(sum(abs(MBlock - predictedMBlock_BCF))); 
    MinErr = min(SAD_PF, SAD_BCF);
    if MinErr == SAD_PF
      Mode = 0; mvh = mvh_PF, mvw = mvw_PF;
    else if MinErr == SAD_BCF
      Mode = 1; mvh = 0, mvw = 0;
    end;
    if Mode = 0
      predictedMBlock = predictedMBlock_PF;
    else if Mode = 1
      predictedMBlock = predictedMBlock_BCF;
    end;
    ErrorBlock = MBlock - predictedMBlock;
    Block1 = ErrorBlock(1:8, 1:8);
    DCTBlock1 = dct2(Block1)
    QDCT1 = quantizeDCT(DCTBlock1, Gp, G_matrix).

Block2 = Error Block (9:16; 1:8);
DCT Block2 = dct2(Block2);
QDCT2 = quantize DCT(DCT Block2, OP, QMatrix);
Block3 = Error Block (1:8; 9:16);
DCT Block3 = dct2(Block3);
QDCT3 = quantize DCT(DCT Block3, OP, QMatrix);
Block4 = Error Block (9:16; 9:16);
DCT Block4 = dct2(Block4);
QDCT4 = quantize DCT(DCT Block4, OP, QMatrix);
Entropy Coding (mode, mwh, mv, QDCT1, QDCT2, QDCT3, QDCT4, outfile);
Quantized DCT Block1 = decodequantize DCT(QDCT1, OP, QMatrix);
Quantized DCT Block2 = decodequantize DCT(QDCT2, OP, QMatrix);
Quantized DCT Block3 = decodequantize DCT(QDCT3, OP, QMatrix);
Quantized DCT Block4 = decodequantize DCT(QDCT4, OP, QMatrix);
Quantized Error Block = zeros(16, 16);
Quantized Error Block (1:8, 1:8) = Quantized DCT Block1;
Quantized Error Block (9:16, 1:8) = Quantized DCT Block2;
Quantized Error Block (1:8, 9:16) = Quantized DCT Block3;
Quantized Error Block (9:16, 9:16) = Quantized DCT Block4;
Quantized MB Block = predicted MB Block + Quantized Error Block;
RT (r=r+15, c=c+15) = Quantized MB Block;
end;
end,