EE6427 Assignment

Wu Tianwei

Matriculation No. G2101446F

e-mail: WU0008EI@e.ntu.edu.sg

(1) Calculate two-dimensional transform of figure 1 by using row-column decomposition method with basis function is figure 2, please show all the intermediate steps to obtain the result.

$$\begin{pmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \\ 20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \\ 40 & 40 & 40 & 40 & 40 & 40 & 40 & 40 \\ 40 & 40 & 40 & 40 & 40 & 40 & 40 & 40 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \end{pmatrix}$$

Figure 1

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

Figure 2

If the quantization matrix on the page 6 of the lecture note “JPEG” is used, calculate the quantization output. What is the one-dimensional output after zig-zag scanning?

Solution:

Calculate two-dimension transform by using row column decomposition method

$$\begin{pmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \\ 20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \\ 40 & 40 & 40 & 40 & 40 & 40 & 40 & 40 \\ 40 & 40 & 40 & 40 & 40 & 40 & 40 & 40 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \end{pmatrix}$$

1) The result of each row can be computed as:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{pmatrix} = \begin{pmatrix} 80 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Transpose each result:

$$\begin{pmatrix} 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 160 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 160 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 320 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 320 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2) The result of each column can be computed as:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 80 \\ 80 \\ 160 \\ 160 \\ 320 \\ 320 \\ 80 \\ 80 \end{pmatrix} = \begin{pmatrix} 1280 \\ 0 \\ 320 \\ 0 \\ -320 \\ 0 \\ -640 \\ 0 \end{pmatrix}$$

The result is:

$$\begin{pmatrix} 1280 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 320 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -320 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -640 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3) Quantization

The quantization matrix is

$$\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

After applying quantization with default $QF = 1$, the result is

$$\begin{pmatrix} 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

4) After Zig-Zag scanning, the one-dimensional output is

[80, 0, 0, 23, 0, 0, 0, 0, 0, 0, -18, 0, 0, 0, 0, 0, 0, 0, 0, 0, -13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

(2) Define a character string using YOUR FULL NAME as appearing in Matriculation Card (all in capital letters and remove all spaces in your name) and follow by "VIDEOSIGNALP".

Use arithmetic coding method in the lecture note "compression fundamental" to encode the first 8 letters of the character string. Show the steps of the divisions of the interval during arithmetic encoding the character string and show the codeword produced by the encoding procedure.

Solution:

My name is Wu Tianwei. So, the character string is "WUTIANWEIVIDEOSIGNALP". The string of the first 8 letters of it is "WUTIANWE".

Table 1

Character	Probability	Interval (Range)
W	2/8	[0.000 − 0.250)
U	1/8	[0.250 − 0.375)
T	1/8	[0.375 − 0.500)
I	1/8	[0.500 − 0.625)
A	1/8	[0.625 − 0.750)
N	1/8	[0.750 − 0.875)
E	1/8	[0.875 − 1.000)

Table 2

New Character	Low Value	High Value
	0.0	1.0
W	0.00	0.25
U	0.06250	0.09375
T	0.07421875	0.07812500
I	0.07617187500	0.07666015625
A	0.07647705078125	0.07653808593750
N	0.07652282714843750	0.07653045654296875
W	0.0765228271484375000	0.0765247344970703125
E	0.076524496078491210937 5	0.076524734497070312500 0

According to Table 2, the final low value $0.076524496078491210937 5$ represents the message "WUTIANWE".

(3) Define a character string using YOUR FULL NAME as appearing in Matriculation Card (all in capital letters and remove all spaces in your name) and follow by "VIDEOSIGNALP". Please remove all spaces in the string and let "A" = 1, "B" = 2, …, "Y" = 25, "Z" = 26 as an input to fill up the following 4x4 matrix.

Let the 4x4 matrix (obtained from above) be a two-level discrete wavelet transform decomposition result. Applying the EZW coding scheme to the wavelet coefficients and show the encoding result. Note that four symbols in dominant pass for EZW are T (zerotree root), Z (isolated zero), P (positive) and N (negative) respectively.

Solution:

My name is Wu Tianwei. So, the character string is "WUTIANWEIVIDEOSIGNALP".

The 4x4 matrix is

$$\begin{bmatrix} 23 & 21 & 20 & 9 \\ 1 & 14 & 23 & 5 \\ 9 & 22 & 9 & 4 \\ 5 & 15 & 19 & 9 \end{bmatrix}$$

According to EZW coding, we choose initial threshold

$$T_0 = 2^{\left\lfloor \log_2(\max(abs(w_{x,y}))) \right\rfloor} = 2^4 = 16$$

So, we have

1) Threshold: 16

D1: PPZZ PTPT TPTT TTPT

S1: 000000

23	21	20	23	22	19
1	1	1	1	1	1
0	0	0	0	0	0
1	1	1	1	1	0
1	0	0	1	1	1
1	1	0	1	0	1

2) Threshold: 8

D2: ZZZP TPTT PTTP PTTP

S2: 111110100100

23	21	20	23	22	19	14	9	9	15	9	9
1	1	1	1	1	1						
0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	0	1	0	0	1	0	0
1	0	0	1	1	1	1	0	0	1	0	0
1	1	0	1	0	1	0	1	1	1	1	1

3) Threshold: 4

D3: ZZZZ TTTP TTPT TPTT

S3: 100111100100000

23	21	20	23	22	19	14	9	9	15	9	9	5	5	4
1	1	1	1	1	1									
0	0	0	0	0	0	1	1	1	1	1	1			
1	1	1	1	1	0	1	0	0	1	0	0	1	1	1
1	0	0	1	1	1	1	0	0	1	0	0	0	0	0

1	1	0	1	0	1	0	1	1	1	1	1	1	1	0

4) Threshold: 2

D4: ZTTT

S4: 110101011111110

23	21	20	23	22	19	14	9	9	15	9	9	5	5	4
1	1	1	1	1	1									
0	0	0	0	0	0	1	1	1	1	1	1			
1	1	1	1	1	0	1	0	0	1	0	0	1	1	1
1	0	0	1	1	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	1	1	1	1	1	1	1	0

5) Threshold: 1

D5: ZTPT TTTT

(4) Solution:

The tmn.exe supports inputting QP from 1 to 31. So, I choose QP from 2 to 31 to calculate the PSNR-Y and MSE-Y.

First, I use tmn.exe to obtain the reconstructed video. Here is my cmd code.

Encode: tmn -i ./football_cif.yuv -x 3 -b 149 -S 0 -O 0 -I 2 -q 2 -o football_cif_2.yuv -B football_cif_2.263
Decode: tmndec -o5 -l football_cif_02.263 football_cif_02_decode.yuv

After obtaining the reconstructed videos, the overall PSNR and MSE of each video can be calculated. In this experiment, I use Matlab to compute PSNRs and MSEs.

Matlab Code:

```
% Different QPs
PSNR_Y = zeros(1, 31);
MSE_Y = zeros(1, 31);
```

```
for i = 2 : 31
    if i < 10
        str = ['0', num2str(i)];
    else
        str = num2str(i);
    end
    disp(str);
    [PSNR_Y(i), MSE_Y(i), ~] = yuvpsnr(['football_cif_',
str,'_decode.yuv'], 'football_cif.yuv', 352, 288, '420', 'y');
end

bitrate = [5355.12 3784.63 2718.93 2211.47 1763.83 1522.87 1283.76
1147.70 1000.27 913.09 817.12 758.08 689.69 648.08 599.02 567.63
529.91 ...
506.87 477.66 459.59 436.98 422.14 404.62 392.17 377.48 368.11
355.22 347.15 337.59 330.61];

figure(1);
plot(bitrate, PSNR_Y(2 : end), 'b-*');
xlabel('bitrate(kbit/sec)');
ylabel('PSNR-Y');

figure(2);
plot(bitrate, MSE_Y(2 : end), 'g--o');
xlabel('bitrate(kbit/sec)');
ylabel('MSE-Y');

PSNR_Yt = PSNR_Y';
MSE_Yt = MSE_Y';
```

```
function [PSNR, msemean, mse] = yuvpsnr(File1, File2, width,
height, format, parameter)
    % Set factor for UV-sampling
    fwidth = 0.5;
    fheight = 0.5;
    if strcmp(format, '400')
        fwidth = 0;
        fheight = 0;
    elseif strcmp(format, '411')
        fwidth = 0.25;
        fheight = 1;
    elseif strcmp(format, '420')
        fwidth = 0.5;
        fheight = 0.5;
    elseif strcmp(format, '422')
```

```
        fwidth = 0.5;
        fheight = 1;
    elseif strcmp(format, '444')
        fwidth = 1;
        fheight = 1;
    else
        disp('Error: wrong format');
    end

    % Get Filesize and Frame number
    filep = dir(File1);
    fileBytes = filep.bytes; % Filesize1
    clear filep
    framenumber1 = fileBytes/(width*height*(1+2*fheight*fwidth)); %
Framenumber1
    filep = dir(File2);
    fileBytes = filep.bytes; % Filesize2
    clear filep
    framenumber2 = fileBytes/(width*height*(1+2*fheight*fwidth)); %
Framenumber2
    if mod(framenumber1, 1) ~= 0 || mod(framenumber2, 1) ~= 0 ||
framenumber1 ~= framenumber2
        disp('Error: wrong resolution, format, filesize or different
video lengths');
    else
        h = waitbar(0,'Please wait ... ');
        mse = zeros(1, framenumber1);
        disp(framenumber1);
        for cntf = 1 : framenumber1
            waitbar(cntf / framenumber1, h);
            % Load data of frames
            YUV1 = loadFileYUV(width, height, cntf, File1, fheight,
fwidth);
            YUV2 = loadFileYUV(width, height, cntf, File2, fheight,
fwidth);
            % Get MSE for single frames
            if parameter == 'y'
                mse(cntf) = sum(sum((double(YUV1(:, :, 1)) -
double(YUV2(:, :, 1))).^2)) / (width * height);
            elseif parameter == 'u'
                mse(cntf) = sum(sum((double(YUV1(:, :, 2)) -
double(YUV2(:, :, 2))).^2)) / (width * height);
            elseif parameter == 'v'
                mse(cntf) = sum(sum((double(YUV1(:, :, 3)) -
```

```
double(YUV2(:, :, 3))).^2)) / (width * height);
          elseif parameter == 'yuv'
              mse(cntf) = sum((double(YUV1(:)) -
double(YUV2(:))).^2) / length(YUV1(:));
          end
       end
       % Compute the mean of the mse vector
       msemean = (sum(mse) / length(mse));
       % Compute the psnr
       if msemean ~= 0
          PSNR = 10 * log10((255^2) / msemean);
       else
          PSNR = Inf;
       end
       close(h);
    end
end


% Read YUV-data from file
function YUV = loadFileYUV(width, heigth, Frame, fileName, Teil_h,
Teil_b)
    % Get size of U and V
    fileId = fopen(fileName, 'r');
    width_h = width * Teil_b;
    heigth_h = heigth * Teil_h;
    % Compute factor for framesize
    factor = 1 + (Teil_h * Teil_b) * 2;
    % Compute framesize
    framesize = width * heigth;

    fseek(fileId,(Frame - 1) * factor * framesize, 'bof');
    % Create Y-Matrix
    YMatrix = fread(fileId, width * heigth, 'uchar');
    YMatrix = int16(reshape(YMatrix, width, heigth)');
    % Create U- and V- Matrix
    if Teil_h == 0
       UMatrix = 0;
       VMatrix = 0;
    else
       UMatrix = fread(fileId,width_h * heigth_h, 'uchar');
       UMatrix = int16(UMatrix);
       UMatrix = reshape(UMatrix,width_h, heigth_h).';
```

```
    VMatrix = fread(fileId,width_h * heigth_h, 'uchar');
    VMatrix = int16(VMatrix);
    VMatrix = reshape(VMatrix,width_h, heigth_h).';
end
% Compose the YUV-matrix:
YUV(1:heigth,1:width,1) = YMatrix;

if Teil_h == 0
    YUV(:,:,2) = 127;
    YUV(:,:,3) = 127;
end
% Consideration of the subsampling of U and V
if Teil_b == 1
    UMatrix1(:,:) = UMatrix(:,:);
    VMatrix1(:,:) = VMatrix(:,:);

elseif Teil_b == 0.5
    UMatrix1(1:heigth_h,1:width) = int16(0);
    UMatrix1(1:heigth_h,1:2:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,2:2:end) = UMatrix(:,1:1:end);

    VMatrix1(1:heigth_h,1:width) = int16(0);
    VMatrix1(1:heigth_h,1:2:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,2:2:end) = VMatrix(:,1:1:end);

elseif Teil_b == 0.25
    UMatrix1(1:heigth_h,1:width) = int16(0);
    UMatrix1(1:heigth_h,1:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,2:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,3:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,4:4:end) = UMatrix(:,1:1:end);

    VMatrix1(1:heigth_h,1:width) = int16(0);
    VMatrix1(1:heigth_h,1:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,2:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,3:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,4:4:end) = VMatrix(:,1:1:end);
end

if Teil_h == 1
    YUV(:,:,2) = UMatrix1(:,:);
    YUV(:,:,3) = VMatrix1(:,:);

elseif Teil_h == 0.5
```

```
        YUV(1:heigth,1:width,2) = int16(0);
        YUV(1:2:end,:,2) = UMatrix1(:,:);
        YUV(2:2:end,:,2) = UMatrix1(:,:);

        YUV(1:heigth,1:width,3) = int16(0);
        YUV(1:2:end,:,3) = VMatrix1(:,:);
        YUV(2:2:end,:,3) = VMatrix1(:,:);

    elseif Teil_h == 0.25
        YUV(1:heigth,1:width,2) = int16(0);
        YUV(1:4:end,:,2) = UMatrix1(:,:);
        YUV(2:4:end,:,2) = UMatrix1(:,:);
        YUV(3:4:end,:,2) = UMatrix1(:,:);
        YUV(4:4:end,:,2) = UMatrix1(:,:);

        YUV(1:heigth,1:width) = int16(0);
        YUV(1:4:end,:,3) = VMatrix1(:,:);
        YUV(2:4:end,:,3) = VMatrix1(:,:);
        YUV(3:4:end,:,3) = VMatrix1(:,:);
        YUV(4:4:end,:,3) = VMatrix1(:,:);
    end
    YUV = uint8(YUV);
    fclose(fileId);
end
```

The results are shown in Table 3.

Table 3

QP	Bitrate	PSNR-Y	MSE-Y
2	5355.12	42.40208	3.740007
3	3784.63	40.84601	5.351548
4	2718.93	38.35847	9.489293
5	2211.47	37.35224	11.96344
6	1763.83	35.84034	16.94519
7	1522.87	35.1712	19.76791
8	1283.76	34.15151	24.99937
9	1147.70	33.64074	28.11943
10	1000.27	32.87845	33.51469
11	913.09	32.47345	36.79046
12	817.12	31.88999	42.08054
13	758.08	31.5628	45.37325
14	689.69	31.09413	50.54371

15	648.08	30.826	53.76265
16	599.02	30.42398	58.97694
17	567.63	30.19534	62.16501
18	529.91	29.86721	67.04387
19	506.87	29.68389	69.93452
20	477.66	29.39422	74.75813
21	459.59	29.22624	77.70629
22	436.98	28.9854	82.13729
23	422.14	28.8232	85.26299
24	404.62	28.60127	89.73331
25	392.17	28.46895	92.50921
26	377.48	28.27544	96.7245
27	368.11	28.13002	100.0181
28	355.22	27.95235	104.1946
29	347.15	27.83291	107.0999
30	337.59	27.6793	110.956
31	330.61	27.58892	113.2893

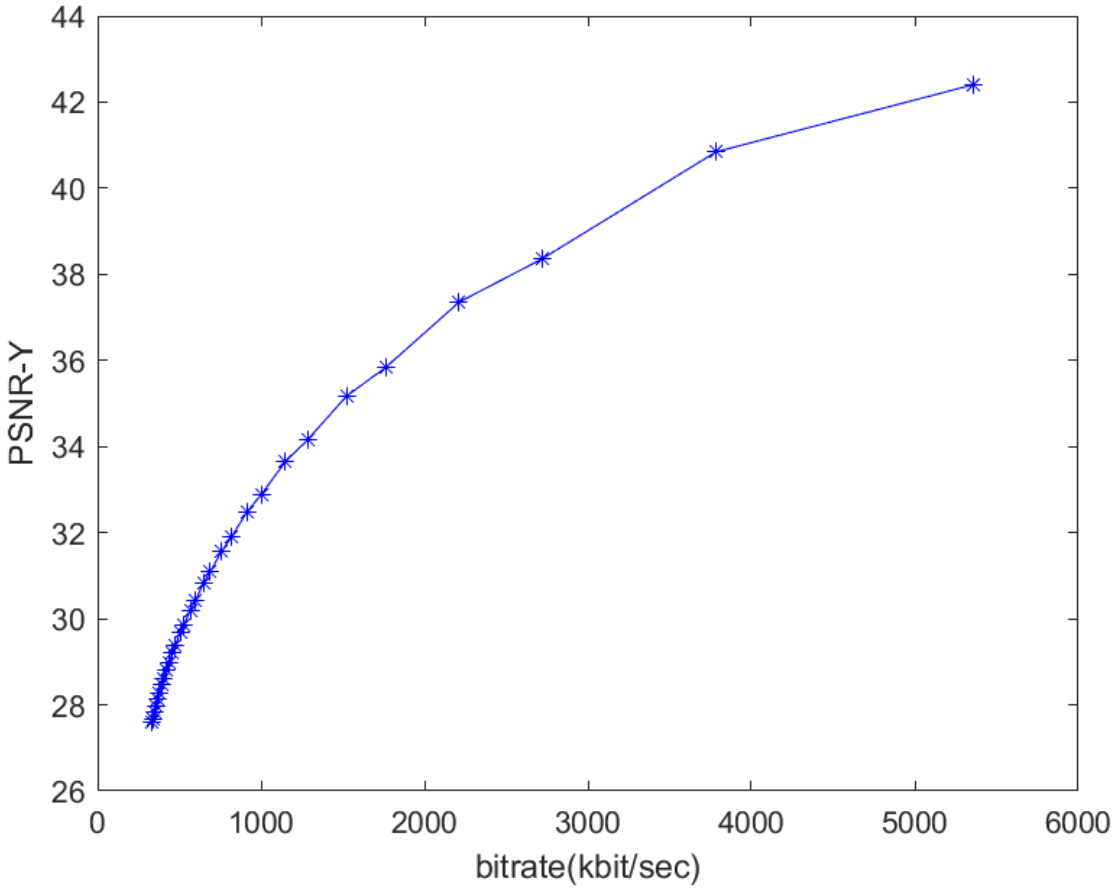

Figure 3

Figure 3 shows the PSNR-Y against various bitrate. It's obvious that the PSNR-Y is positive correlated with bitrate. This result perfectly meets expectation. As the QP increases, the compression ratio of the video also increases, which means lower bitrate is required to represent the data. Meanwhile, higher compression ratio means more noise in the decoded video. So, the PSNR-Y will become smaller.

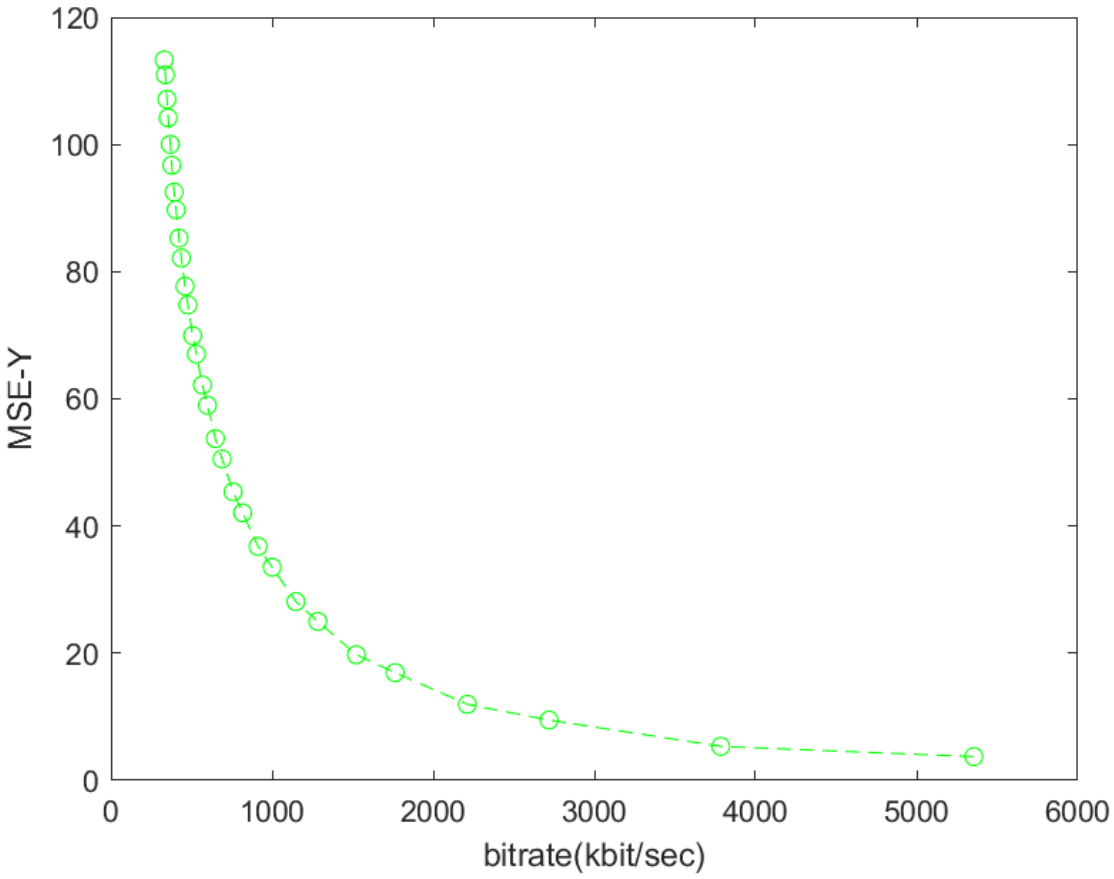

Figure 4

Figure 4 shows the MSE-Y against various bitrate. It's obvious that the MSE-Y is negative correlated with bitrate. It's easy to explain the result. As the QP increases, more noise will be in the decoded video, which means the error between the original pixels x_i and the reconstructed pixels $\hat{x}_i$ will increase. Thus, the MSE-Y of the video will also increase.

To get the MSE-Y against frame number, I choose 6 different bitrates. Here is my cmd

code.

```
tmn -i ./football_cif.yuv -x 3 -b 149 -S 0 -O 0 -r 2450000 -R 30 -o football_cif_fixedB_2450.yuv -B football_cif_fixedB_2450.263
```

Matlab Code:

```
% Fixed bitrate
bitrate = [2450 3200 3800 4200 4800 5200];
MSE_Y_PF = zeros(6, 150);
for i = 1 : 6
    str = num2str(bitrate(i));
    disp(str);
    [~, ~, MSE_Y_PF(i, :)] = yuvpsnr(['football_cif_fixedB_', str,'.yuv'], 'football_cif.yuv', 352, 288, '420', 'y');
end

s = 1 : 150;
figure(3);
plot(s, MSE_Y_PF(1, :), s, MSE_Y_PF(2, :), s, MSE_Y_PF(3, :), s, MSE_Y_PF(4, :), s, MSE_Y_PF(5, :), s, MSE_Y_PF(6, :));
xlabel('Frame no');
ylabel('MSE-Y');
legend('2447.63kbit/sec', '3176.61kbit/sec', '3780.11kbit/sec', '4161.79kbit/sec', '4565.25kbit/sec', '4753.75kbit/sec');
```

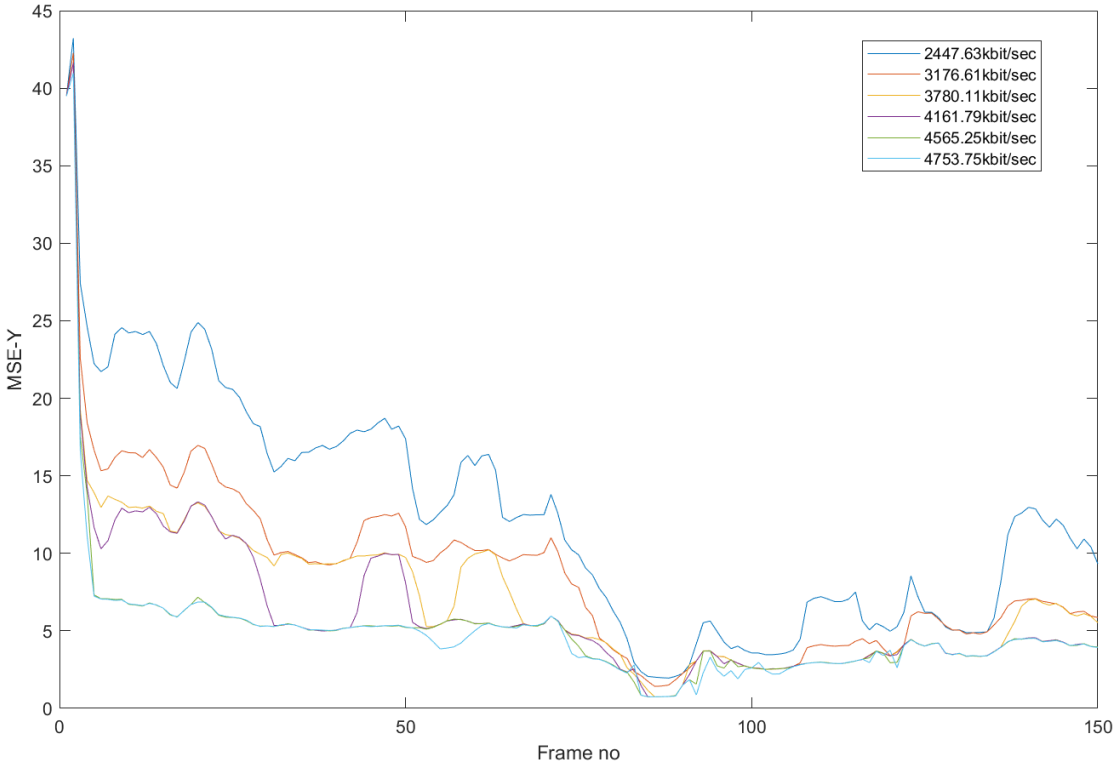

Figure 5

Respectively, the six fixed bitrates are

bitrate(kbit/sec)	2447.63	3176.61	3780.11	4161.79	4565.25	4753.75

Figure 5 shows the MSE-Y against frame number. As we can see, using higher bitrate will achieve lower MSE-Y and the six curves share similar trend. It's obvious that there is a minimum value of all the curves. In my opinion, the reason maybe the two adjacent frames closely resemble, which means the motion vectors need to be encoded is much fewer than other frames' and much less noise will be introduced. Therefore, the MSE-Y is much lower.