EEE116 – Multimedia Systems (2007/08)

Solutions to Tutorial Problem Sheet 7
Week 9

(Q.12)
HDTV sequences consist of RGB 4:4:4 format video with 1920x1088 spatial resolution, 10 bit depth for each of three colours and 50 frames per second frame rate.

(i) Compute the bit rate of an uncompressed HDTV sequence.

Data rate: = (pixels/line x lines/frame) x (chroma sample factor x bits/pixel) x frames/sec
= (1920x1088)x(3x10)x50
= 3.1334e+009 bits/sec
= 2.92 G bits/sec. (diving by 2^30)

(ii) What percentage of data rate can be reduced by using YCbCr 4:2:0 format, instead of RGB 4:4:4?

4:2:2 means,
we take for each 2x2 samples of Y 2 samples of chrominance (1 from Cb and 1 from Cr) from the line 1 and another 0 samples of chrominance (1 from Cb and 1 from Cr) from the line 2.

This means, for each 2x2 of Y, we have 1 of Cb and 1 of Cr resulting in 6 (i.e., 2x2+1x1+1x1) samples, instead of 12 (i.e., 2x2+2x2+2x2) samples in the original 4:4:4 case.

That means we can save 6 samples for every 12 samples. i.e., the data rate is reduced by 50% of the original size.

(iii) How much disk space would it take to store a 1-hour HDTV programme in the uncompressed YCbCr 4:2:0 format?

The new data rate = original data rate x 1/2
= (1920x1088)x(3x10)x50 x1/2 bits/sec

For 1 hour data size= (1920x1088)x(3x10)x50 x 1/2 x60 x 60 bits
(divide by 2^{30}x8 to convert to G Bytes)
= 656 G Bytes.

(Q.13)

(i) Compute the coding redundancy of this image block.

An N-bit code (we consider a fixed length code) can represent 2^N different symbols.
An 8-bit code can represent $2^8 = 256$ different values.

[In images, if we use 8 bits to represent sampling values at each coordinate point we can represent 256 different sampling values, which can vary from 0 to 255. In a gray-scale image 0 is mapped to black and 255 is mapped to white. All other values between 0 and 255 represent shades of gray varying between black and white].

The coding redundancy is computed as follows:
Coding redundancy = 1 – Coding Efficiency

Coding Efficiency = (Source entropy)/(Average number of bits used in the code)

$$= \left(\sum_{i=1}^{5} P_i \log_2(1/P_i)\right) / \left(\sum_{i=1}^{5} P_i n_i\right)$$

First we should compute the average source entropy

f		P_i	$-\log_2(P_i)$	$-P_i\log_2(P_i)$
8	8/256	0.0313	5	0.1563
32	32/256	0.1250	3	0.3750
128	128/256	0.5000	1	0.5000
64	64/256	0.2500	2	0.5000
16	16/256	0.0625	4	0.2500
4	4/256	0.0156	6	0.0938
2	2/256	0.0078	7	0.0547
2	2/256	0.0078	7	0.0547
Totals 256	1			1.9844

Average source entropy is 1.98 bits/sample

Average bits using the 8-bit codewords is 8 bits/sample

Therefore, the Coding Efficiency is 1.98/8 =24.7%
Reduncay= 1 – Coding Efficiency= 75.3%

(ii) How much compression can be achieved by using Huffman code to represent these pixel values? (Compute your answer without deriving the Huffman codes).

Since all probabilities are integer powers of (1/2), the Huffman code efficiency is 100%. That means the average code length using the Huffman coding is the same as the theoretical value computed using the Shannon's entropy formula.

We compute the compression ratio as
= (average bits for the original code) / (average bits for the new code)

Therefore the compression ratio = 8 / 1.98 = 4.04:1.

(iii) In order to further compress this image block, you decided to quantise the gray values using a quantisation factor Q=8.

a. Now derive the new frequencies of occurrence for the quantised values.

The quantisation of x with respect to Q results in round(x/Q)
Therefore, after quantisation the grey scale values become:

x	Round(x/8)	frequency
125	16	8
131	16	32
134	17	128
135	17	64
139	17	16
141	18	4
144	18	2
150	19	2

Now we can create a new frequency table for the new symbol set as follows:

symbol	frequency
16	40
17	208
18	6
19	2

b. Compute the new entropy value.

f	P_i	$-\log_2(P_i)$	$-P_i\log_2(P_i)$
40	0.156	2.6781	0.4184
208	0.813	0.2996	0.2434
6	0.023	5.4150	0.1269
2	0.008	7.0000	0.0547
totals			
256	1.0000		0.8434

Entropy 0.84 bits/sample.

c. Derive the Huffman code for the new symbols.

	P	Merge 1	Merge 2		Huffman code
17	0.81			1	1
16	0.16		1	0.19 0	01
18	0.02	1 0.03	0		001
19	0.01	0			000

d. Compute the compression ratio achieved by combined quantisation and entropy (Huffman) encoding.

You have to compute the average code length using the Huffman coding and compare it with the original 3-bi fixed length code representation.

Average code length = $\sum_{i=1}^{4} n_i P_i$ = 1*0.81+2*0.16+3*0.02+3*0.01

= 1.22 bits/symbol

n is the number of bits in each codeword

compression ratio: 8: 1.22= 6.56:1