RRY025- SOLUTIONS TO PROBLEMS

PROBLEM SET E - IMAGE COMPRESSION

1) The minimum number of bits/symbol for lossless compression is given by the entropy

$$H = -\sum_i p_i log_2 p_i$$

where p_i is probability of each symbol.

M=1 In this case we code each pixel seperately.

```
Symbol     Probability          Huffman Code

  0           0.95     \ 0           0
                        ---
  1           0.05     / 1           1
```

Entropy per symbol $H_1 = 0.286$ bits/pixel. Average length of Huffman code = 1. Efficiency is 0.28.

M=2 In this case we encode pairs of pixels Since we are told successive pixels are independant we can easily calculate the probability of each possible combination of two pixels

```
Symbol     Probability                              Huffman Code

  00          0.9025                           \0            0
                                                 1.000
  01          0.0475               \0          /1           10
                                     0.0975
  10          0.0475    \0         /1                      110
                          - 0.050 -
  11          0.0025    /1                                 111
```

The entropy per symbol $H_2 = 0.572$ bits/symbol or 0.286 bits/pixel. Average length of the Huffman code is $1 \times 0.9025 + 2 \times 0.0475 + 3 \times 0.0475 + 3 \times 0.0025 = 1.147$. Efficiency $0.573/1.147 = 0.5$.

M=3

```
Symbol Probability                                                 Huffman Code

000    0.87375   -------------------------------------------- 0       0
                                                              \
001    0.04512   \0                         0                 - 1    100
                    -0.09024 ------------------------- 0.1426 /1
010    0.04512   /1                                    /              101
                                                      /1
100    0.04512   ----------------------              /                110
                                        \           /
011    0.002375  \0                      \0        /                 11100
                    -0.00475              - 0.052175
101    0.002375  /1           \0         /1                          11101
                                - 0.00725
110    0.002375  \0           /1                                     11110
                    -0.00250
111    0.000125  /1                                                  11111
```

The entropy per symbol $H_3 = 0.859$ bits/symbol or 0.286 bits/pixel. Average length of the Huffman code is 1.3bits. Efficiency $0.859/1.3 = 0.66$.

2) a) Huffman Coding. Calculate probability of each symbol.

```
Symbol     Probability                             Huffman Code

  3           7/16                           \0             0
                                               - 1-
  2           6/16                \0         /1            10
                                     -9/16-
  1           2/16   \0           /1                      110
                         - 3/16 -
  0           1/16   /1                                   111
```

Mean bits/pixel $1 \times 7/16 + 2 \times 6/16 + 3 \times 2/16 + 3 \times 1/16 = 28/16 = 1.75$ bits/pixel. Plain message required 2 bits/pixel. Compression ratio achieved is $2/1.75 = 1.14$

b) Form differences via horizontal raster scan

```
        0    0   -1

    0   1    0    0

    0  -1    0    0

    0  -1    0   -1
```

Calculate Huffman code.

```
Symbol     Probability                            Huffman Code

  0           10/15                    \0               0
                                          - 1-
 -1           4/15           \0        /1              10
                               -5/15-
  1           1/15           /1                        11
```

Number of bits to transmit image $1 \times 10 + 2 \times 4 + 2 \times 1 = 20$ but must add 2 bits to transmit first undifferenced value in image top left corner, hence total of 22 bits needed. Plain message has 16 pixels with 2bits/pixel = 32bits. Compression ratio is therefore $32/22 = 1.45$.

c) Run length coding. Transmit value and length of run, no need to represent a run of zero hence we can represent runs of lengths 1 to 4 with two bits.

Code	Run Length
00	1
01	2
10	3
11	4

Using horizontal raster scanning image can be represented as

$$(3,3)(2,2)(3,4)(2,4)(1,2)(0,1)$$

or six runs, where the first number in brackets is the value and the second is the length of the run. Using 2 bits for the value and 2 bits for the length of run it requires 24bits to transmit image and so the compression ratio is $32/24 = 1.33$.

d) Both the compressor and decompressor know that the original image has 2bits/pixels or 4 levels per pixel. There are only two topologies of Huffman trees for a 4 level system, viz

```
Symbol                           Symbol

A                    \0          A     \0
                        --                ---
B             \0     /1          B     /1      \1
                 ---                              ----
C     \0      /1                 C     \0      /0
         ---                              ---
D     /1                         D     /1
```

The compressor needs to communicate to the decompressor which type of Huffman tree we have (which we can indicate by the first bit of the proptocol 0 or 1), and then which numbers 00,01,10,11 correspond to the leaf nodesA,B.C and D. In fact only values for A,B,C need be communicated; the symbol for D is the remaing 2 bit binary number. Hence the protocol requires only 7 bits at the start to communicate the Huffman table that is being used, 1 bit for the topology, and 2 bits each to send A, B and C. Once the receiver get this information it can construct the same Huffman codebook used by the transmitter and then decode the rest of the bit stream

To communicate N images requires 32N bits uncompressed. If we Huffman code we must send 7 + 16*1.75N = 7+28N bits, so overall we get compression for $N > 1$.

3) a) Block transform. Divide the image into seperate areas and transform each one. A fast transform of a MxM block takes of order $M^2 log_2 M$ operations. If we divide an NxN image into $(N/M)^2$ blocks, the full block transform takes $N^2 log_2 M$ operations compared to $N^2 log_2 N$ operations for a normal transform. The speed ratio is them $log_2 N/log_2 M = 9/3 = 3$ for M=8,N=512.

b) Compression comes largely by elliminating interpixel redundancy within each block. If the block becomes too small it doesn't contain all the correlated pixels and the compression ratio is reduced. If the correlation between adjacent pixels is $\rho = 0.9$ and the image can be approximated as a Markov-1 process then the degree of correlation of two pixels just depends on the distance Δ between them and equals ρ^{Δ}; for $\rho = 0.9$ and $\Delta = 8$ the correlation becomes 0.43. Hence for a M=8 sized block pixels at opposite ends of block are only just correlated. Having a larger block does not increase compression ratio but just slows transform down.

c) Can think of DCT in terms of taking original blocks adding flipped versions and then repeating an infinite number of times, then taking continuous FT. This procedure gives no sharp edges, and therefore there is a reduced need for large high spatial frequencies components compared to the DFT.

4) a) In lossless compression, no information is lost and it is possible to reconstruct the original image exactly. Only relatively small amounts of compression are possible. A possible use is the storage of images which have been very expensive of difficult to obtain. In lossy compression higher compression rates are possible, but the process is irreversible. A use would be transmit images found on a web page which are only to be quickly viewed.

b) Even though the range of values that must be transmitted is larger, the entropy of the difference histogram is much smaller than in the original image. Using entropy coding it is therefore to get higher ratios of lossless compression.

c) Using the suggested predictor rule and quantiser

u(n)	u(n-1)	e(n)	e'(n)	u'(n-1)	u'(n)
101					101
110	101	9	5	101	106
107	110	-3	-5	106	101
108	107	1	1	101	102
105	108	-3	-5	102	97
102	105	-3	-5	97	92

d) Using a DPCM system we can simulate the receiver at the transmitter and we can we can calculate *what the receiver thought the last pixel was.* We can form the error between the present pixel value and a prediction based on this receiver estimate, quantise this and transmit. Using such a system we obtain

u(n)	u(n-1)	e(n)	e'(n)	u'(n-1)	u'(n)
101					101
110	101	9	5	101	106

107	106	1	1	106	107
108	107	1	1	107	108
105	108	-3	-5	108	103
102	103	-1	-1	103	102

5)

	Coding Redundancy	Interpixel Redundancy	Psychovisual Redundancy	Lossless
Huffman coding of pixel values	yes	no	no	yes
Simple Run length coding of a binary image	no	yes	no	yes
Delta Modulation	no	yes	no	no
Standard JPEG	yes	yes	yes	no

6) a) The single symbol entropy is

$$-0.1 log_2 0.1 - 0.9 log_2 0.9 = 0.4690$$

b) Create Huffman tree.

```
Symbol     Probability                                  Huffman Code

  00          0.81                          \0             0
                                                 -1.000
  01          0.09                \0        /1            10
                                      -0.19 -
  10          0.09   \0           /1                     110
                           - 0.1 -
  11          0.01   /1                                  111
```

The mean number of bits per symbol is $0.81 + 2 \times 0.09 + 3 \times 0.09 + 3 \times 0.01 = 1.29$ or 0.64 bits/pixel. The lossless compression ratio is $1/0.64 = 1.5504$.

c) The probability of n zeros then a 1 is $p^n(1-p)$, the probability of having M zeros is p^M.

d) The mean number of pixels per run is $10(1 - 0.9^7) = 5.217$. Since for M=7, m=3, each run requires 3 bits to give its length. On average we therefore replace 5.217 pixels or 5.217 bits in the plain message with 3 bits; hence the compression ratio is $5.217/3 = 1.739$.

e) In the case of Run length coding a wrong bit causes a wrong length of run to be assumed. This will cause all later pixels to be shifted in position. In the case of Huffman coding we will decode a wrong value and will likley mess up the decoding of subsequent codes. May later recover though. Causes some bad pixels at start of image plus possible shift of the image by a few pixels.