***LOSSLESS IMAGE COMPRESSION AND TRANSMISSION***

Aim

To execute Lossless image compression using Huffman Coding

Theory

To store large sized images and to make them available on internet, compression techniques are needed. Image compression addresses the problem of reducing the amount of data required to represent a digital image.The underlying basis of the reduction process is the removal of redundant data. According to mathematical point of view, this amounts to transforming a two-dimensional pixel array into a statistically uncorrelated data set.At reciever, the comressed image is decompressed to the approximation of the original image.There are two types of compression techniques, lossless compression and lossy compression.In general, lossy compression techniques have an advantage of greater compression ratios and a disadvantage of image quality degradation.Huffman coding, a lossless image compression technique is implemented in the following procedure,

Huffman coding

Huffman code procedure is based on the two

observations.

a. More frequently occurred symbols will have shorter

code words than symbol that occur less frequently.

b. The two symbols that occur least frequently will have the same length.

Table 1: Huffman source reduction.

Original source Source reduction

S P 1 2 3 4

a2 0.4 0.4 0.4 0.4 0.6

a6 0.3 0.3 0.3 0.3 0.4

a1 0.1 0.1 0.2 0.3

a4 0.1 0.1 0.1

a3 0.06 0.1

a5 0.04

S-source, P-probability

Table 2 : Huffman Code Assignment Procedure

Original source Source reduction

S P 1 2 3 4

a2 0.4[1] 0.4[1] 0.4[1] 0.4[1] 0.6[0]

a6 0.3[00] 0.3[00] 0.3[00] 0.3[00] 0.4[1]

a1 0.1[011] 0.1[011] 0.2[010] 0.3[01]

a4 0.1[0100] 0.1[0100] 0.1[011]

a3 0.06[01010] 0.1[0101]

a5 0.04[01011]

S-source, P-probability

At the far left of the table I the symbols are listed and corresponding symbol probabilities are arranged in decreasing order and now the least two

probabilities are merged as here 0.06 and 0.04 are merged, this gives a compound symbol with probability 0.1, and the compound symbol probability is placed in source reduction column1 such that again the probabilities should be in decreasing order. so this process is continued until only two probabilities are left at the far right shown in the above table as 0.6 and 0.4. The second step in Huffman’s procedure is to code each reduced source, starting with the smallest source and working back to its original source. The minimal length binary code for a two-symbol source, of course, is the symbols 0 and 1. As shown in table II these symbols are assigned to the two symbols on the right (the assignment is arbitrary; reversing the order of the 0 and would work just and well). As the reduced source symbol with probabilities 0.6 was generated by combining two symbols in the reduced source to, the 0 used to code it is now assigned to both of these symbols, and a 0 and 1 are arbitrary appended to each to distinguish them from each other. This operation is then repeated for each reduced source until the original course is reached. The final code appears at the far-left in table . The average length of the code is given by the average of the product of probability of the symbol and number of bits used to encode it. This is calculated below

Lavg = (0.4)(1) +(0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5)

+ (0.04)(5) = 2.2 bits/ symbol

and the entropy of the source is 2.14bits/symbol, the

resulting Huffman code efficiency is2.14/2.2 = 0.973.

Entropy, H**=**-**Σ**P(aj)log P(aj)

Huffman’s procedure creates the optimal code for a set of symbols and probabilities subject to the constraint that the symbols be coded one at a time.

Huffman decoding

After the code has been created, coding and/or decoding is accomplished in a simple look-up table manner. The code itself is an instantaneous uniquely

decodable block code. It is called a block code, because each source symbol is mapped into a fixed sequence of code symbols. It is instantaneous, because each codeword in a string of code symbols can be decoded without

referencing succeeding symbols. It is uniquely decodable, because any string of code symbols can be decoded in only one way. Thus, any string of Huffman encoded symbols can be decoded by examining the individual

symbols of the string in a left to right manner. For the binary code of table 2, a left-to-right scan of then coded string 010100111100 reveals that the first valid code word is 01010, which is the code for symbol a3.The next

valid code is 011, which corresponds to symbola1. Valid code for the symbol a2 is 1,valid code for the symbols a6 is 00,valid code for the symbol a6 is Continuing in this manner reveals the completely decoded message a5 a2 a6

a4 a3 a1 , so in this manner the original image or data can be decompressed using Huffman decoding as explained above. At first we have as much as the compressor does a probability distribution. The compressor made a code

table. The decompressor doesn't use this method though. It instead keeps the whole Huffman binary tree, and of course a pointer to the root to do the recursion process. In our implementation we'll make the tree as usual and then you'll store a pointer to last node in the list, which is the root. Then the process can start. We'll navigate the tree by using the pointers to the children that each node has. This process is done by a recursive function which accepts as a parameter a pointer to the current node, and returns the

symbol.