Department of Electrical and Computer Engineering
The Johns Hopkins University
520.643 Digital Multimedia Coding and Processing – Fall 2000

Homework Assignment II

1. (20 points) Consider the coding of a source with output probabilities

$$\{\ 0.05,\ 0.1,\ 0.15,\ 0.17,\ 0.13,\ 0.4\ \}.$$

   a. Design a ternary Huffman code (using $\{0, 1, 2\}$ as letters instead of the popular binary $\{0, 1\}$) for this source. Draw the Huffman tree. *Hint: dummy helps!*

   b. What is the resulting average codeword (expected) length?

   c. Compare the average codeword length with the source entropy. What logarithmic base yields a meaningful comparison?

   d. Design a binary Huffman code for this source and compute its average codeword length.

2. (30 points) Consider a source $\mathcal{A}$ that generates symbols from an alphabet of size four, $\mathcal{A} = \{a, b, c, d\}$, with probabilities

$$P(a) = \frac{1}{2}, \quad P(b) = \frac{1}{4}, \quad P(c) = \frac{1}{8}, \quad \text{and} \quad P(d) = \frac{1}{8}.$$

   a. Find the binary arithmetic code for the sequence *abacabd*. Show all of your encoding steps along with the coding intervals.

   b. Decode the binary code above and verify that your encoding is correct. Is it possible to have an integer implementation of the arithmetic codec in this case?

3. (20 points) Two random variables X_1 and X_2 are uniformly distributed on the diamond shown in Figure 1 below.

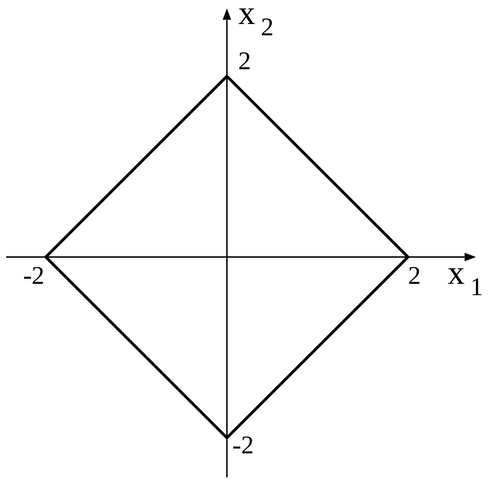

Figure 1: Distribution of random variables X_1 and X_2.

a. Find $f_{X_1}(x_1)$ and $f_{X_2}(x_2)$.

b. Assume that each random variable is quantized using a 4-level uniform quantizer. What is the resulting distortion? What is the resulting number of bits needed per (X_1, X_2) pair?

c. Now use a vector quantizer with the same number of bits per source output pair (X_1, X_2)as in part b. Compute the distortion for the vector quantizer. Compare your results.

4. **Computer Assignment** (30 points). In this assignment, you are asked to implement a Huffman coding/decoding algorithm. Feel free to download and use the software package on the course webpage *http://thanglong.ece.jhu.edu/Course/643/* (with your own modification of course).

   a. Write a subroutine (`huff.c`) to construct Huffman codes. The input is the probability distribution of the symbols. The output is an ASCII Huffman table with the symbols and their corresponding Huffman codes.

   b. Write a program to encode a file using Huffman coding. Treat the input file as a sequence of bytes, each byte as an `unsigned char`. Hence, the set of symbols is $\{0, 1, 2, \ldots, 255\}$. This program can be a two-pass process. In the first pass, frequency of each symbol is collected. In the second pass, string of symbols are converted into cascaded codewords which are saved into an output file. If the total number of bits is not a multiple of 8, pad enough zero bits at the end. You should also save the Huffman table into a separate file.

   c. Write the corresponding Huffman decoder. The inputs are the compressed bitstream and the Huffman table. The command lines should look like this:

   ```
   huffmanencode filename huffman_table compressed_bitstream
   huffmandecode compressed_bitstream huffman_table filename_rec
   ```

   d. Test your Huffman codec on the first 10 frames of the `glasgow` sequence. Is the compression lossless? How much compression can you achieve on the first 10 frames, not counting the Huffman tables?

   e. Now, instead of coding each frame independently, Huffman encode the difference between sequential frames. Say, encode Frame 1, then Frame 2 - Frame 1, Frame 3 - Frame 2, Frame 4 - Frame 3, etc. Write the appropriate decoder for this algorithm. Can you achieve losslessness? Compare your result with Part d's above.

Due date: **October 13** in class