THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
ISDN 2602

Laboratory 4: Source and Channel Coding (5%)

Answer Sheet

Please write down your answer here and submit your answer on GitHub by Wednesday (Oct 29th) 23:59

Part I: Source Coding

Task 1 – Length of the bit streams

In this task, we will compare the lengths of the bit streams for four source coding algorithms applied to a black-and-white image: "raw" image encoding, run-length encoding with lengths encoded as 8-bit binary numbers, and run-length encoding with lengths encoded by Huffman coding with one or two dictionaries.

Check Point:

1) Write down the lengths of the bit streams using “raw” image encoding and the run-length encoding. Is the run-length code better than the raw encoding? **Explain why**.

Raw data size is 250000 bits and run length is 301688 bits, in this case raw encoding is better since it requires less bits and takes less storage, run length is better when you have an image that contains large areas of the same color or pixel value.

---

2) Type “help transpose” in the command window to learn how to perform matrix transpose operation on a matrix in MATLAB. Revise the MATLAB codes so that the image will be rotated along the diagonal. Then, write down and compare the lengths of the bitstreams for these four source coding algorithms before and after the rotation. **Explain why**.

Before rotation: Raw data size is 250000 bits, run length is 301688 bits and for huffman with same dictionary
The size is 117374 bits and for different dictionary it is 100981 bits,

After rotation: Raw data size is 250000 bits, run length is 196680 bits and for huffman with same dictionary
The size is 134892 bits and for different dictionary it is 120565 bits,

Raw data size is the same because rotation of the picture does not change the total pixel count, run length is smaller because now it is longer runs that can be grouped together, and huffman increase it might be because of change in frequency of different characters

<u>Fill in the answers to the blanks and Show your result to the TA.</u>

Task 2 – Huffman code

In this task, you will generate the Huffman code for a set of run-lengths, and use it to encode the run- lengths of black or white pixels. You will find that Huffman coding enables us to encode the sequence of run lengths using fewer bits than the standard 8-bit encoding.

Check point:

1) Find an optimal dictionary to represent these 11 symbols using the symbol probabilities and the Huffman coding algorithm. Once you have found it, replace the value of **dict** defined between the line:

   % % % % Revise the following code to generate a valid and efficient dictionary % % % %

   and

   % % % % Do not change the code below % % % %

   The remaining part of the code uses this dictionary to encode the run lengths, and to measure the length of the resulting bit stream. It also checks whether the dictionary is valid by reconstructing the image from the run lengths encoded by the dictionary using the function **huffman_encode_dict**. If your dictionary is correct, the original and reconstructed images should be the same and the **size_huffman** should be equal to 117374.

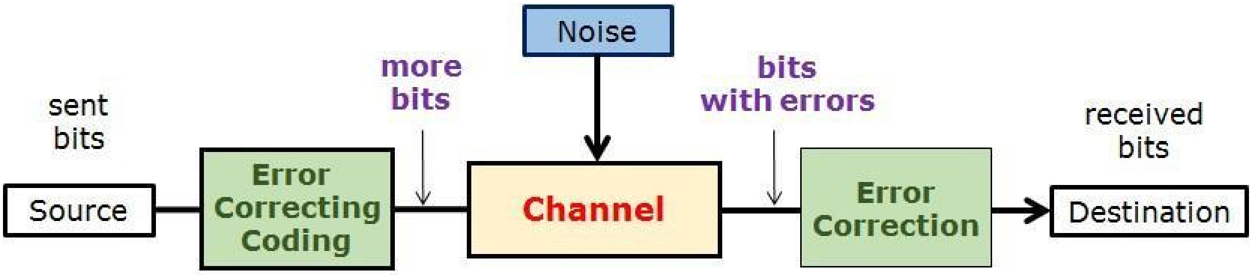
   (Commit the revised codes to GitHub. Show your results to TAs.)

2) Attach the corresponding Huffman tree of the revised optimal dictionary.

Fill in the answers, commit the revised codes to GitHub

and Show your result to the TA.

<u>Part II: Channel Coding</u>



Task 3 – (n,k) block code decoder and Error Correction Capability

In this task, we will implement the (n,k) block code decoder and compare the error correction capability of the repetition code, hamming block code, and no error correction code.

Check point:

1) Generate a figure with three curves representing the BER performance.

（**Show your results to the TA**）

---

2) Write down/Insert a screenshot of the modified code in "**blk_decoder.m**".

(Commit the revised codes to GitHub.)

---

3) Based on your observations, which coding scheme performs the best? **Explain why**.

Based on my observation the repetition coding scheme performs the best because it has a lower BER on all the distances, meaning it is more robust in this case compared to the other methods

---

<u>Fill in the answers, commit the revised codes to GitHub</u>

<u>and Show your result to the TA.</u>

----------------------------------End-----------------------------------