Image Processing HW3

Shang-Ying,He
Communication Engineering
National Tsing Hua University
Hsinchu,Taiwan. R.O.C.
sunnyhe1122@gmail.com

I. Introduction

Program about the image coding and image permutation, for image coding, include Huffman, Arithmetic and LZW coding, and compute entropy of the image also. For the image permutation, it's about recover image which is randomly shifted by rows or columns.

II. METHOD

A. Image Coding

First part of image coding is to compute entropy and horizontal difference entropy of image. For the entropy, first step is to calculate the probability of intensities in image and just to follow the function (1) at last to get the result we want. For the horizontal difference entropy, it's different from entropy with that it chained each pixel on right hand side to a pair and using these pairs to calculate the probabilities(e.g. TABLE I.), and also follow the function (1) to get the result.

$$H = -\sum_{k=1}^{L-1} P_r(r_k)\log_2 P_r(r_k) \tag{1}$$

TABLE I.

21 21 21 95 169 243 243 243		
21 21 21 95 169 243 243 243		
21 21 21 95 169 243 243 243		
21 21 21 95 169 243 243 243		
pair	***count***	***probability***
21,21	8	1/4
21,95	4	1/8
95,169	4	1/8
169,243	4	1/8
243,243	8	1/4
243,21	4	1/8

Second part of image coding is obtained Huffman, Arithmetic and LZW coding to compress the image.

For the Huffman coding, first step is to construct binary tree by the probability of intensities and for each node, set all the leaf of left as 1, right as 0 naturally. After constructing the tree we can get the code for each intensity simply.

For the Arithmetic coding, it is better than Huffman coding in achieving the Shannon value. I compress four pixels into a numeric range from 0 to 1 and every time when I got this numeric, I rounding it into 8 number of digits, in other words I only have to record 4 pixels by 27 bits.

For the LZW coding, this method builds the dictionary dynamically. TABLE II. is an example, first we need an dictionary obtain A to Z, 26 letters and corresponding to their codeword from 1 to 26. Next step is to read the data in sequence, we need to check the current and next letters if they are in the dictionary or not, if current is in the dictionary we read next into current until combine of current and next doesn't in dictionary but current is in dictionary, so that next is to encode current into codeword and add combine of current and next into dictionary. After iterations of doing this, we would get the codes and dictionary which can decode this data.

TABLE II.

Data	TOBEORNOTTOBEORNOTTOBEORNOT		
Initial Dictionary	A-Z:1-26		
current	***next***	***code***	***dictionary***
T	O	20	TO:27
O	B	15	OB:28
B	E	2	BE:29
E	O	5	EO:30
O	R	15	OR:31
R	N	18	RN:32
N	O	14	NO:33
O	T	15	OT:34
T	T	20	TT:35
TO	B	27	TOB:36
BE	O	29	BEO:37
OR	T	31	ORT:38
TOB	E	36	TOBE:39
EO	R	30	EOR:40
RN	O	32	RNO:41
OT		34	

B. Image Permutation

This task totally has two part, first part is to recover a randomly row shifted image and second part is different from first part that the randomly shifted are columns. Thus, propose method of recover two of these images is similar.

In randomly shifted row image, because the first row is not shifted, and for every normal images, their correlations between each row are very large roughly. So we can roll the next row continuously until it back to the original position and for each rolling we compute correlation coefficient matrix between the rolling row and previous row, and choose the most related rolling number to roll the next row. For many iterations to do so until to the last row we can get the recovered image.

For the randomly shifted column image, it's method is complete the same as the randomly shifted row image, only different with the row and column.

III. EXPERIMENTS

A. *Image Coding*

For image coding, we can see the result in TABLE III. LZW coding could get the better compression ratio than Huffman coding and Arithmetic coding always get the 1.1851 compression ratio because that this method will compress 4 pixels into a float range from 0 to 1 for every images, and I have try to compress more than 4 pixels but it would cause a problem that the more pixels I compressed the float would stop in a value and caused error when we decoding. Thus, in experiments, compressed 4 pixels into a float is the best value to avoid error when decoding.

TABLE III.

TABLE III.			
Entropy		7.5714	7.4078
Horizontal Difference Entropy		6.2136	6.1346
compression Ratio	Huffman coding	1.0533	1.0767
	Arithmetic coding	1.1851	1.1851
	LZW coding	1.0628	1.0834

B. *Image Permutation*

We can see the result in Figure 1. It is very simple to get the result if we did the right step in first iteration, but this task is very slow to calculate, in experiments, it might cost twenty seconds to compute in my device(I5-6500).

IV. CONCLUSION

This project let me know about the Huffman, Arithmetic and LZW coding in detail, and entropy is the target of bit per pixel we want to achieve when encoding, this concept is the same as the information theory I have learned in college, so when I contacted in second time it really impressed me.

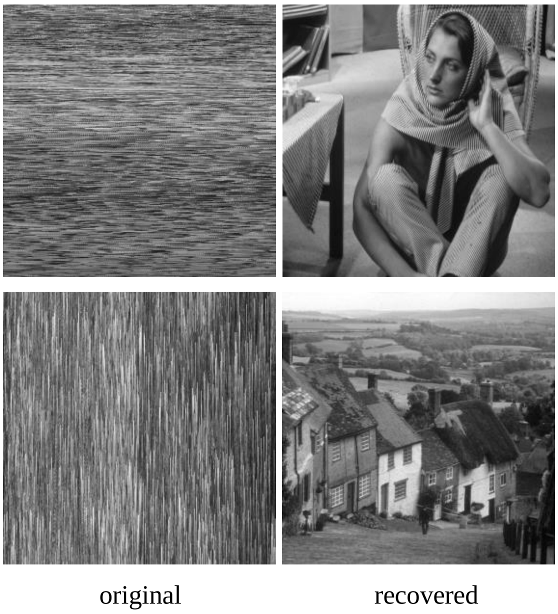

Figure 1. recover of the randomly shifted image