RLE Performance Comparison on Distinctive Images

Huseyin Can Ercan

25 October, 2017

1 Introduction

Run-length encoding is a simple lossless data compression method, rather than symbols of the original data method care about runs of the data. Storing all of the same consecutive symbols is an inefficient way to store data, instead of storing all individual symbols method stores only symbols and counts, run lengths. This approach has various implementations and practices; I decided to use simplest implementation for encoding algorithm. Storing the symbol count and symbol is our approach for RLE, except monochrome data encoding. For monochrome data, single bits are used for symbol storage, while encoding white-black-white sequence used for symbol count storage. Thus for monochrome there is no need for symbol indication.

1.1 Algorithm and Data Format

Storing the symbol count and symbol is one method for RLE implementation, for every symbol run using storing the symbol number and the symbol may be inefficient for complex images. Sharp color changes of the close pixels can generate overhead for this method. Even for some cases compressing will generate a larger data than original data. But I have selected this method because our purpose is comparison rather than constructing a more efficient method for RLE, applying the same method for all cases answers our purpose.

For image formats three types are selected for comparison, monochrome, 4 bit gray code and 255 color table. Monochrome is the simplest form, pixels are describes as white and black, white pixel value is 1, black is 0. 4 bit gray scale uses 16 value for grey color. 255 color table use 8 bits to define colors.

For transformation rules; monochrome uses half value threshold for 1 and 0 pixels, (R + G + B)/3 should exceed the 128 limit for white(1) pixel, 4 bit grey scale divides (R + G + B)/3 value with 16 (reduce 8 bit 4 bit), 256 color table is constructed RRRGGGBB as values. 8 bit red reduced to 3 bit, 8 bit green to 3 bit 8 bit blue to 2 bit. Input data 24-bit bmp file.

4 different RLE runs paths are used. Column, row and zigzag and segmented zigzag. Segmented zigzag divides the picture to 4 quarters and applies compression to generated sequence.

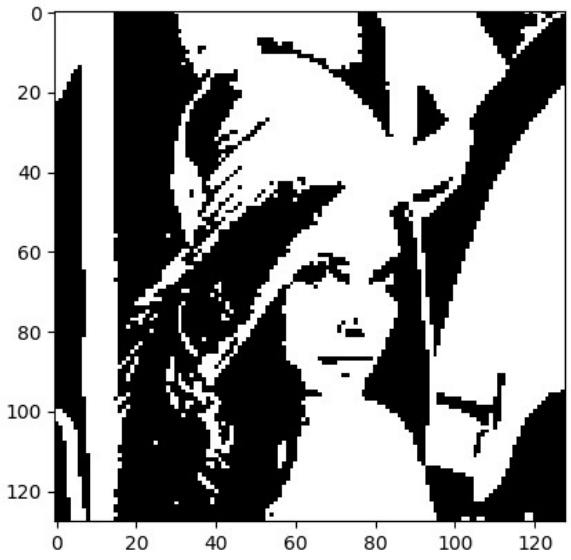

Image 1 Monochrome lenna

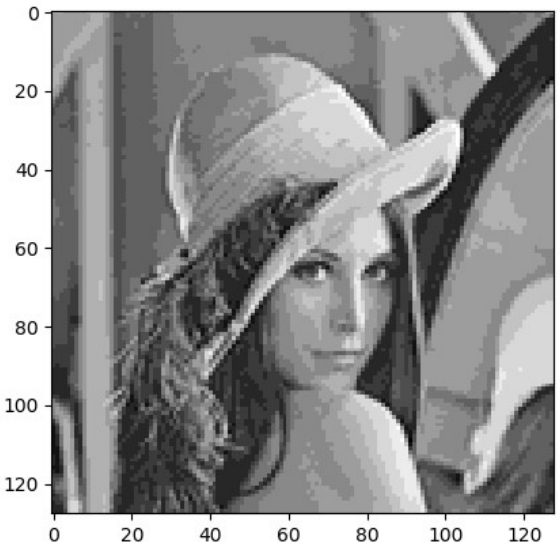

Image 2 4 bit gray scale lenna

Table 1 Monochrome transformation example

Bit:	0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Count:	\x00\x02\x02\x01\x02\x02\x03\x16

Monochrome data encoding we accept first number describes the white pixel number of the original data, if there is no white pixel at the beginning of the data we need to describe count as zero to keep order, white pixel values are 1 for our implementation.

4 bit gray scale and 255 color table same approach is used, differently we need to express the symbol and symbol count. For 4 bit gray scale 4 bit used to store symbol count, as a result 1 byte defines a reconstruct able unit. For 255 color table 1 byte used for symbol, 1 byte used for symbol run length.

1.2 Implementation Details

For implementation Python 3.5.2 is used. For bmp file input “struct” package is used. “struct” package provides required utility for C type data structures. For image plotting “matplotlib” module is used.

1.3 Performance

Compression results shows method performance is related to data itself. Pixel distributions, shaped areas, color change boundaries are affecting the performance. For a sight picture row travel is more effective, for lena column more effective. Also zigzag and segmented zigzag have same behavior, lena show more performance for segmented zigzag travel. For goldhill segmented zigzag algorithm have failed, algorithm can`t travel in goldhill`s divided dimensions, because of this reason goldhill`s segmented zigzag cells are empty.

1.4 Conclusion

Compression with RLE highly depends on the data. Because, for every RLE approach there is a presupposition in hypothesis. Method expects the symbols in travel order, otherwise it loses performance. Because of this reason data characteristics become a critical factor on performance. For effective usage RLE may benefit from an extra component for method decision. Otherwise selected method`s results are going to be unpredictable.

Table 2 Compression performance table

	biber	lenna	ucak	goldhill
MONO ROW	67	16	52	66
MONO COLUMN	69	45	53	57
MONO ZIGZAG	58	16	39	46
MONO ZIGZAG SEGMENT	44	20	50	x
4BIT ROW	66	55	67	63
4BIT COLUMN	71	66	63	60
4BIT ZIGZAG	62	55	58	51
4BIT ZIGZAG SEGMENT	61	63	62	x
256COLOR ROW	27	3	42	25
256COLOR COLUMN	44	24	32	19
256COLOR ZIGZAG	19	2	22	0
256COLOR ZIGZAG SEGMENT	18	16	20	x

Table 3 Compression ratio table

	biber	lenna	ucak	goldhill
MONO ROW	0.32	0.83	0.47	0.33
MONO COLUMN	0.30	0.54	0.46	0.42
MONO ZIGZAG	0.41	0.83	0.60	0.53
MONO ZIGZAG SEGMENT	0.55	0.79	0.49	x
4BIT ROW	0.33	0.44	0.32	0.36
4BIT COLUMN	0.28	0.33	0.36	0.39
4BIT ZIGZAG	0.37	0.44	0.41	0.48
4BIT ZIGZAG SEGMENT	0.38	0.36	0.37	x
256COLOR ROW	0.72	0.96	0.57	0.74
256COLOR COLUMN	0.55	0.75	0.67	0.80
256COLOR ZIGZAG	0.80	0.97	0.77	0.99
256COLOR ZIGZAG SEGMENT	0.81	0.83	0.79	x

Table 4 Compression performance table, efficiency selection

	biber	lenna	ucak	goldhill
MONO ROW	67	16	52	**66**
MONO COLUMN	**69**	**45**	**53**	57
MONO ZIGZAG	58	16	39	46
MONO ZIGZAG SEGMENT	44	20	50	x
	biber	lenna	ucak	goldhill
4BIT ROW	66	55	**67**	**63**
4BIT COLUMN	**71**	**66**	63	60
4BIT ZIGZAG	62	55	58	51
4BIT ZIGZAG SEGMENT	61	63	62	x
	biber	lenna	ucak	goldhill
256COLOR ROW	27	3	**42**	**25**
256COLOR COLUMN	**44**	**24**	32	19
256COLOR ZIGZAG	19	2	22	0
256COLOR ZIGZAG SEGMENT	18	16	20	x

References

1. https://en.wikipedia.org/wiki/8-bit_color
2. https://algorithm.yuanbin.me/zh-hans/problem_misc/matrix_zigzag_traversal.html