

Image Compression (Huffman coding technique)

MATIN MONSHIZADEH^{1,*}

¹ The Department of Computer Science Engineering and Information Technology, Shiraz University, Iran

*Corresponding author: matinmonshizadeh@gmail.com

Compiled January 28, 2023

Abstract

Image compression is a topic we checked in this research. Many algorithms and techniques exist to implement it, but we have used a unique algorithm for this subject and fully described its implementation with the python programming language.

INTRODUCTION

Image compression is used in different fields and has been effective in other parts. Algorithms play a significant role in this field and should be optimally effective in this situation to get the desired result.

1. IMAGE COMPRESSION

Image compression is an application of information compression on digital images; in other words, this work aims to reduce the redundancy of photo content for the ability to store or transfer information in an optimal form [1]. Many algorithms and techniques exist for image compression. We used the Huffman coding technique to implement this part.

Huffman Coding

The history of the Huffman coding algorithm came about as the result of a class project at MIT by David A. Huffman as a student in 1951. [2]. This algorithm is lossless compression, which means this method keeps the same information as the original image. Lossless compression is also reversible and has the highest possible quality standards. This method is against lossy compression, also known as irreversible compression. Table 1 shows the comparison between lossy and lossless compression.

Table 1. Comparison Between Lossy and Lossless Compression

| | |
|---|--------------------------------------|
| Lossy Compression | Lossless Compression |
| Also known as irreversible compression | Also known as reversible compression |
| Data reduction is higher | Data reduction is lower |
| Resultant file is smaller than the original | Resultant file is not as small |

Huffman is widely used in all the mainstream compression

formats, from GZIP, PKZIP (WinZip, etc.), and BZIP2 to image formats such as JPEG and PNG.

Method

This section describes the implementation of Image Compression Using Huffman Coding Techniques. The first step is to open and read the image as binary and store it in the string, like in figure 1.



Fig. 1. Open and read image as binary.

The second step is to separate the string in which we have stored the binary code into 8 bits chunks and get the frequency of each byte and store it in dictionary, like in figure 2.

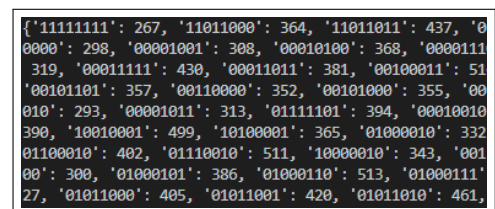


Fig. 2. Storing each byte with its frequency.

After that, we import the `Heapq` library to use min-heap

and make a Huffman tree. Figure 3 shows each byte with its Huffman code stored in the dictionary.

| | | |
|----------|----|----------|
| 10001100 | -> | 00000000 |
| 01011111 | -> | 00000010 |
| 00010011 | -> | 00000011 |
| 10000100 | -> | 00000100 |
| 11110000 | -> | 00000101 |
| 11111000 | -> | 00000110 |
| 11000010 | -> | 00000111 |

Fig. 3. Storing each byte with its frequency.

In the following, we replace the Huffman code on the first string to store compression data.

If the first string that we stored binary code of the image is S_1 and the second string that replaces the Huffman code is S_2 , the compression ratio is calculated as follows:

$$CR = \frac{S_1}{S_2} \quad (1)$$

If we implement the Huffman coding algorithm correctly, the CR will be more than one.

For the last part, we implement the previous parts again in reverse to decompress the image and show that the Huffman coding algorithm is lossless compression.

Algorithm 1. Huffman Coding algorithm

```

1: procedure HUFFMAN(C)
2:    $n \leftarrow |C|$ 
3:    $Q \leftarrow C$ 
4:   for  $i \leftarrow 1$  to  $n - 1$  do
5:     allocate a new node  $z$ 
6:      $left[z] \leftarrow x \leftarrow EXTRACT - MIN(Q)$ 
7:      $right[z] \leftarrow y \leftarrow EXTRACT - MIN(Q)$ 
8:      $f[z] \leftarrow f[x] + f[y]$ 
9:      $INSERT(Q, z)$ 
10:  return  $EXTRACT - MIN(Q)$ 

```




| balloon Properties | balloon_compressed.bin Properties | balloon_decompressed Properties |
|---|---|---|
| General | General | General |
| Security | Security | Security |
| Details | Details | Details |
| Previous Ve | Previous Ve | Previous Ve |
|  |  |  |
| balloon | balloon_compressed.bin | balloon_decompressed |
| Type of file: JPG File (.jpg) | Type of file: BIN File (.bin) | Type of file: JPG File (.jpg) |
| Opens with: Photos | Opens with: Pick an app | Opens with: Photos |
| Location: C:\Users\matin\OneDrive | Location: C:\Users\matin\OneDrive | Location: C:\Users\matin\OneDrive |
| Size: 99.7 KB (102,108 bytes) | Size: 99.6 KB (102,080 bytes) | Size: 99.7 KB (102,108 bytes) |
| Size on disk: 100 KB (102,400 bytes) | Size on disk: 100 KB (102,400 bytes) | Size on disk: 100 KB (102,400 bytes) |

Fig. 4. Compare the size of 3 files. Left picture is the original image, Middle picture is the compressed file, and Right picture is the decompressed file.

Also, we show the Huffman coding algorithm in algorithm 1.

REFERENCES

1. A. Subramanya, "Image compression technique," IEEE potentials **20**, 19–23 (2001).
2. A. Moffat, "Huffman coding," ACM Comput. Surv. (CSUR) **52**, 1–35 (2019).