Technical University of Cluj-Napoca
Faculty of Automation and Computer Science
Department of Computes Science

Grayscale Image Lossless Compression Using Huffman Encoding

Student: Bogdan Tudor-Alexandru, gr. 30434
Supervisor: Asist.dr.ing. Robert Varga
5/22/2022

1. Introduction

What is **image compression**? **Image compression** is a type of data compression applied to digital images, to reduce their cost for storage or transmission. Algorithms may take advantage of visual perception and the statistical properties of image data to provide superior results compared with generic data compression methods which are used for other digital data.

2. Approach

One of the key factors affecting the compression rate of the images is the number of bits used in the representation of each symbol. In a normal grayscale image, for the representation of each pixel 8 bits are used, so a good approach would be to reencode the symbols with new codes with a variable length, such that more occurring symbols will have a shorter representation, while the less occurring one will have a longer representation. This reencoding can be achieved using **Huffman Encoding**.

3. User Manual

When running the application there will be 3 options to choose from :

1 – Encode Image,

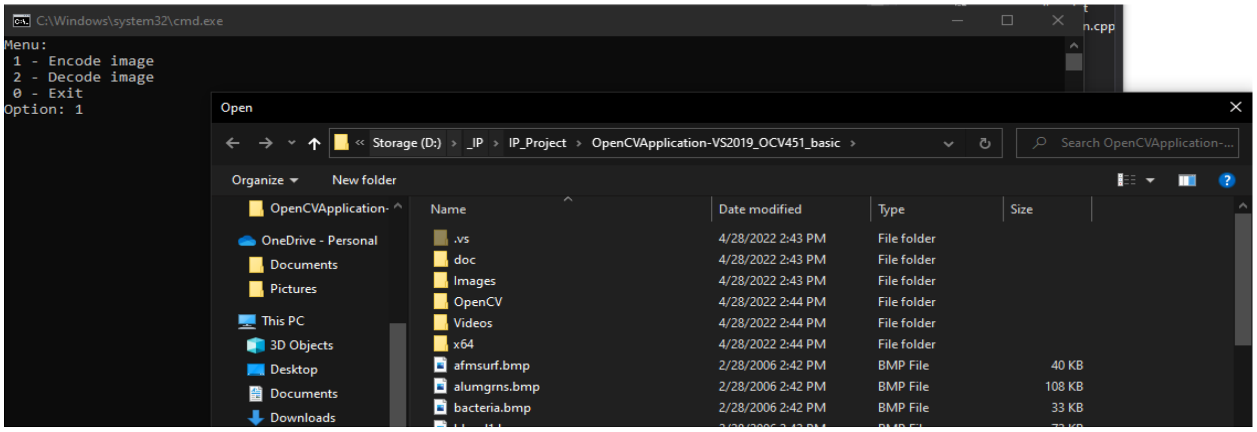
2 – Decode Image,

0 – Exit.

If the encode image is chosen, the user will choose an image to encode, that will be saved in a file named “compressedImg.txt”.

If the decoding option is chosen the application will decode the last image encoded in the “compressedImg.txt”.

The last option will exit the application.


C:\Windows\system32\cmd.exe
Open
Storage (D:) › _IP › IP_Project › OpenCVApplication-VS2019_OCV451_basic ›
Search OpenCVApplication-...
Organize
New folder
OpenCVApplication-
OneDrive - Personal
Documents
Pictures
This PC
3D Objects
Desktop
Documents
Downloads

```
Menu:
 1 - Encode image
 2 - Decode image
 0 - Exit
Option: 1
```

Name	Date modified	Type	Size
.vs	4/28/2022 2:43 PM	File folder	
doc	4/28/2022 2:43 PM	File folder	
Images	4/28/2022 2:43 PM	File folder	
OpenCV	4/28/2022 2:44 PM	File folder	
Videos	4/28/2022 2:44 PM	File folder	
x64	4/28/2022 2:44 PM	File folder	
afmsurf.bmp	2/28/2006 2:42 PM	BMP File	40 KB
alumgrns.bmp	2/28/2006 2:42 PM	BMP File	108 KB
bacteria.bmp	2/28/2006 2:42 PM	BMP File	33 KB

4. Libraries/Functions used

Most of the functions and structures are implemented by myself. The most important ones used that are not implemented by myself are: **PriorityQueue** (the heap used for the building of the Huffman tree) and the **fread** and **fwrite** functions used for binary writing and reading into the files.

5. General Approach and implementation details

The approach of the compression is as follows:

Encoding:

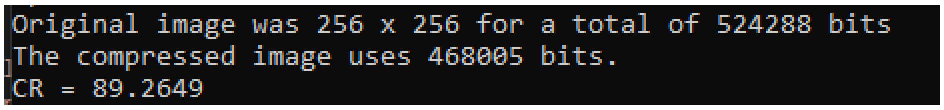
1. Count the number of different symbols occurring in the representation of the image.
2. Make a leaf node for each symbol present
3. While there are more than one node without a parent in the list of nodes, extract the 2 nodes which have the least amount of occurrences, merge them, and put the new node into the list. When there is only one more node remaining, we will consider it the root.
4. Starting from the root, we will go down the newly build tree and allocate codes to each of the current node's children, until we reach a leaf node, as follows: currentNode.code + '0' for the left child and currentNode.code + '1' for the right child (Due to the building of the Huffman tree, the closer a node is to the root, the higher it's number of occurrences will be, and the smaller the code used for it's representation).
5. We will save the new codes corresponding to the leaf nodes (each leaf node represents one of the symbol present in the representation of the original image, and the Huffman code corresponding to it).
6. The only thing left is to build the string corresponding to the encoded image using the following structure: totalNumberOfBits(32bits), numberOfSymbols(8bits), {symbol(8bits), huffmanCodeLength(8bits), huffmanCode(huffmanCodeLength bits)} for each symbol, numberOfRows(8bits), numberOfCols(8bits), huffmanEncodedImage.
7. We make groups of 8 bits (a byte) and we write them 1 by 1 in a txt file (binary writing).

Decoding:

1. We reach the 4 bytes in the txt file to get the totalNumberOfBits in the encodedMessage.
2. Next we decode: numberOfSymbols, {symbol(8bits), huffmanCodeLength(8bits), huffmanCode(huffmanCodeLength bits)} for each symbol, numberOfRows(8bits), numberOfCols(8bits).
3. Based on the Huffman nodes decoded from the encoded message we will rebuild the Huffman tree using a trie.
4. For the remaining bits in the encoded message (huffmanEncodedImage a.k.a the image content), we will read the bits 1 by 1 and go from the root of the Huffman tree downwards, until we reach a leaf node (if the bit value is 0 we go left, otherwise we go right). When a leaf node is reach we will extract the symbol value from the leaf and put it into the reconstructed image.

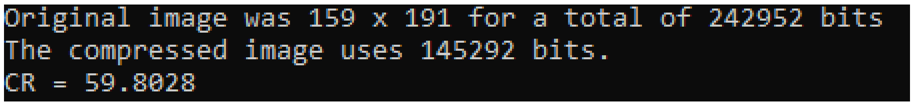
6. Results

1. Cameraman:



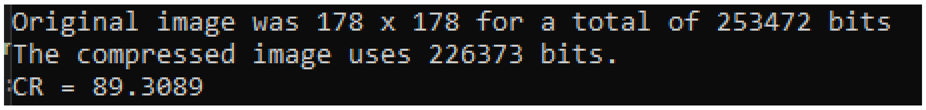
```
Original image was 256 x 256 for a total of 524288 bits
The compressed image uses 468005 bits.
CR = 89.2649
```

2. Cell:



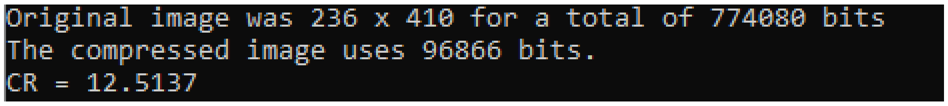
```
Original image was 159 x 191 for a total of 242952 bits
The compressed image uses 145292 bits.
CR = 59.8028
```

3. Bacteria:



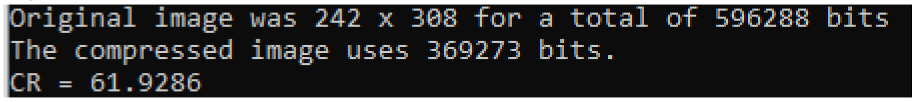
```
Original image was 178 x 178 for a total of 253472 bits
The compressed image uses 226373 bits.
CR = 89.3089
```

4. Letters:



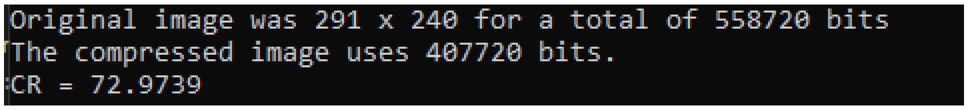
```
Original image was 236 x 410 for a total of 774080 bits
The compressed image uses 96866 bits.
CR = 12.5137
```

5. Eight:



```
Original image was 242 x 308 for a total of 596288 bits
The compressed image uses 369273 bits.
CR = 61.9286
```

6. Pout:



```
Original image was 291 x 240 for a total of 558720 bits
The compressed image uses 407720 bits.
CR = 72.9739
```

7. Bibliography:

1. Huffman Coding | Greedy Algo-3 - GeeksforGeeks
2. Image Compression using Huffman Coding - GeeksforGeeks
3. (PDF) Lossless Grey-scale Image Compression using Source Symbols Reduction and Huffman Coding (researchgate.net)
4. IJIREEICE1E-s-sharankumar-IMPLEMENTATION-OF-HUFFMAN.pdf