Lossless JPEG Image Compression using Huffman Coding

Bushra Rahman Tauhidul Islam Simanto Saha Rakib Ahmad

Department of Computer Science and Engineering
Independent University, Bangladesh
Dhaka, Bangladesh.
{[1]2131283,[2]2030441,[3]2130220,[4]2130227}@iub.edu.bd

Abstract

Images are one of the most important visual representations used in almost every field. They require memory for their storage which necessitates a lot of space. Image compression plays a pivotal role in reducing the size of an image so that more images can be stored and thereby increasing the transmission speed. Many coding algorithms are written to compress images and reduce redundancy. In this project, we proposed the Lossless method of image compression and decompression using a simple coding technique called Huffman coding. Huffman coding is a form of entropy encoding used in lossless data compression. This algorithm is widely used in JPEG image compression and this entropy coding is the main focus of our project work.

Introduction

As technology improved, so did the quality of the images. However, this improvement in quality and resolution comes at a cost, i.e. the amount of data of each pixel stored in the images' raw files takes up more space. Some common issues faced due to this consumption of space, **1.** Bandwidth gets overloaded and slows down. **2.** Limitations in the amount of data a cloud server can store. **3.** Size of an image is directly proportional to the time required to process the image. Image compression lowers the amount of information needed to represent sampled digital images, which lowers the cost of storage and transmission. Image compression acts as a major task in many real-time applications like image databases, satellite imagery for weather, remote sensing image communications, and earth-resource applications. Multimedia data can be compressed using a variety of methods and standards, most notably the JPEG and JPEG 2000 standards for images. We're working with the JPEG standard for this project. The primary reason for compressing JPEG photographs is to minimize file size and conserve space on your hard disk or portable storage device. Huffman encoding, quantization, and other similar processes are necessary for JPEG format image compression. Huffman coding is normally used in the entropy coding phase because it's the most efficient entropy lossless algorithm, which is easy to implement and uses less memory for compressing images. Huffman coding is a type involving lossless data compression. A lossless compression is a form of compression that allows the original data to be perfectly reconstructed with no information loss from the compressed data. Huffman coding is based on the frequency of occurrence of a data item, i.e., a pixel in an image. The approach entails using fewer bits to encode data into binary codes that occur more often. The **four observations** serve as the foundation for the Huffman coding method : **a)** More often appearing symbols will have shorter code words than less frequently occurring symbols. **b)** The two symbols that appear the fewest times will be the same length. **c)** No symbol's code word is a prefix of another symbol's code word. As a result, Huffman coding is uniquely decodable. **d)** Each source symbol must be allocated a unique code word. There are the following **two major steps** involved in Huffman coding : **1.**First, construct a Huffman tree from the given input string or characters or text. **2.** Assign a Huffman code to each character by traversing over the tree. The Huffman code is created by merging the least probable symbols; this procedure is continued until only two probabilities of two compound symbols remain; this produces a code tree, and Huffman codes are derived by labeling the code tree. The Greedy algorithm is the Huffman coding paradigm. A greedy algorithm is a method of solving problems that selects the best option available at the time. At each step, the algorithm makes the near choice that appears to lead toward the goal in the long term. It does not matter if the present best outcome leads to the ultimate best result. The algorithm never reverses the earlier decision, even if the choice is wrong. This approach may not provide the optimal solution for all problems. The greedy technique operates by choosing the locally optimal option at each stage with the goal of discovering the global optimum. This is achieved by minimizing or maximizing the objective function at each stage If both of the properties below are true, a greedy algorithm can be used to solve the problem. **1.Greedy Choice Property 2. Optimal Substructure Property** The greedy approach in Huffman Algorithm is described below with a tree.
1. Categorizing the characters based on their frequency. **2.** Create a new node and then greedily choose the first two nodes from the sorted characters, making the first node the left child of the new node and the second node the right child. The new node's value is the total of the values of its children nodes. **3.** We sort the vertices again based on their values and then repeat the greedy procedure with the first two nodes.

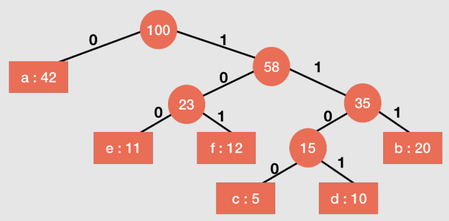

Figure 1. Huffman Algorithm Tree using Greedy Approach

This is the final Binary Tree for optimal prefix code using Huffman Algorithm which uses the Greedy approach.

Rationale for the Algorithm's Selection

In this section, we will discuss why we chose the Huffman algorithm for entropy coding in JPEG image compression. Huffman Coding Advantages: It is easy to encode and decode It is easy to implement It is a fast algorithm Maximum compression ratio can be achieved using correct probability of occurrence This encoding scheme results in saving a lot of storage space, since the binary codes generated are variable in length. There are other reasons why we chose Huffman. Why is this more efficient than other techniques can be proved by computing the efficiency of Huffman coding, **Average length** is, $L(z) = \sum_{i=1}^{n} L(ai)P(ai)$ where L is the length of each code and P is the probability. In our example (shown in part-5), L=1X0.4+2 X 0.3+3X0.1+4 X 0.1+5X0.07+5X0.03L=2.2bits/symbol **Coding Efficiency**, **=H(z)L(z)** where,

$$H(z) = -\sum_{i=1}^{n} P(a_i) * \log P(a_i)$$

It is also known as Information Entropy. H(z) = -(0.4 log(0.4) + 0.3 log(0.3) + 0.1 log(0.1) + 0.1 log(0.1) + 0.07 log(0.07) + 0.03 log(0.03)) = 1.8 Now, =1.82.2=0.81 This means that our Huffman coding is **81 percent efficient.** Another reason for us to pick Huffman is that Shannon's source coding theorem in information theory states that an independent and identically distributed random variable data code rate (average code length for symbols) cannot be less than Shannon's entropy. It has been proven that the Huffman Coding Algorithm achieves optimality in accordance with **Shannon's source coding theorem**, i.e., it delivers the lowest feasible bit rate after encoding. Another type of **entropy coding is Lempel Ziv**. LZW is a dictionary-based compression tool that is widely popular. This implies that instead of tabulating character counts and building trees LZW encodes data by referencing a dictionary. Our primary goal is to achieve a decent compression ratio for all picture sizes. For all text sizes, **Huffman provides a reasonable compression ratio**. This is another reason to use Huffman over the other entropy methods.

Methodology

Entropy coding attains the additional lossless compression by encoding the quantized DCT coefficient more densely based on their statistical distinctiveness. In JPEG two Entropy coding methods are available, **1) Huffman Coding 2) Arithmetic Coding** The algorithm that we chose for the project is Huffman coding. It's an entropy coding algorithm used for lossless data compression. There are **2 essential steps** for Huffman coding in the image compression process- **1. Building Tree 2. Assigning Codes 1. Build a Huffman Tree : a**. Combine the two leaf nodes with the lowest probability into a new node. **b.** Replace the two leaf nodes with the new node, and then sort the nodes based on the new probability values. **c.** Repeat steps (a) and (b) until we have a single node with a probability of 1.0. This node will be referred to as the root. **Assigning Codes : a.** Backtrack from the root, assigning "0" or "1" to each intermediate node until we reach the leaf nodes. The tree of Huffman codes and their codes are given below in a table.

Methodology

symbol	probability	Code	Reduction 1	Reduction 2	Reduction 3	Reduction 4
s1	0.4	1	0.4 (1)	0.4 (1)	0.4 (1)	0.4 (1)
s2	0.3	00	0.3 (00)	0.3 (00)	0.3 (00)	0.6 (0)
s3	0.1	010	0.1 (010)	0.1 (010)	0.3 (01)	-
s4	0.1	0110	0.1 (0110)	0.2 (011)	-	-
s5	0.07	01110	0.1 (0111)	-	-	-
s6	0.03	01111	-	-	-	-

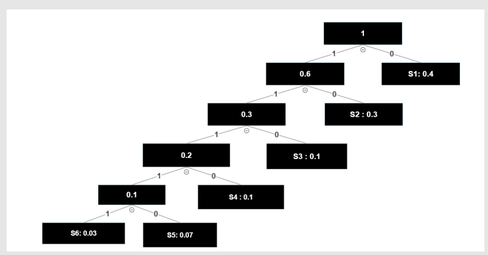

Figure 2. Image Compression using Huffman Algorithm

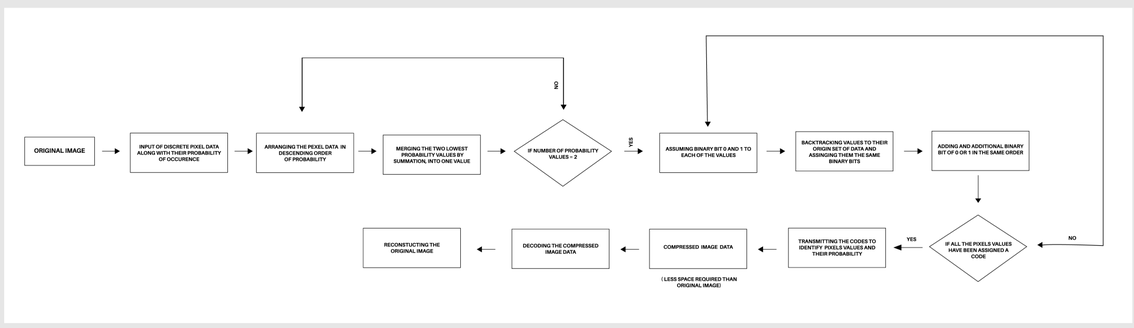

Figure 3. Flowchart of Huffman Algorithm

Substitute Use of the Algorithm

Although image compression is one of the most common applications of the Huffman compression algorithm, It has many other widely used applications as well, some of which include: 1. **Text compression** 2. Multimedia codecs MP3 uses Huffman encoding for audio compression The MP3 also employs the traditional Huffman algorithm approach. It acts at the conclusion of the compression to code information, making it a coding method rather than a compression algorithm. This coding generates variable-length codes on an entire bit set. The codes for higher-probability symbols are shorter. Because Huffman codes contain a unique prefix, they can be accurately decoded despite their varied length. The decoding process is really rapid (via a correspondence table). This type of coding helps you to save around 20 percent of your disk space. It is an excellent complement to perceptual coding. Perceptual coding is particularly efficient in large polyphonies because many sounds are masked or reduced, but little information is identical, therefore the Huffman technique is seldom efficient. There are limited masking effects during "pure" sounds, but Huffman is particularly efficient since digitized sounds include numerous repeating bytes that will be replaced by shorter codes.

Alternative Solution

There are primarily two forms of entropy coding in our JPEG picture compression: **1.Huffman Coding**,**2. Arithmetic coding**. The basic principle underlying Arithmetic coding is to give an interval to each symbol. The experimental results of the Huffman and arithmetic coding methods for compression ratio and execution time are shown in a table compiled from studies on images of various sizes such as 128x128, 256x256, 512x512, 1024x1024 and 2048x2048. As shown in the table, the compression ratio of arithmetic coding for various picture sizes is higher than that of Huffman coding.

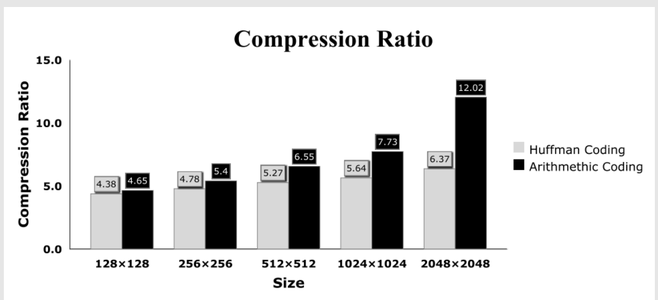

Figure 4. Compression Ratio of Huffman and Arithmetic Algorithm

The compression ratio demonstrates that the Arithmetic method outperforms the Huffman algorithm by a wide margin. Since the compression ratio is our primary concern, the Arithmetic Method outperforms the Huffman algorithm. As a result, the Huffman method can be substituted with the Arithmetic algorithm to improve overall performance. **For overall improvement of the system,**
1. Use DWT instead of DCT. DWT gives greater image quality than DCT due to its higher compression ratio. DWT will help keep the image quality in its original form, which was previously impossible with other image compression algorithms. Because our goal is to minimize information loss, DWT is optimal for the purpose. So, in terms of quality, the DWT approach outperforms the DCT technique.
2. Employing different methods instead of Huffman compression code to increase the overall performance Arithmetic coding should be chosen for entropy coding in JPEG compression. Because Huffman encoding provides a lower compression ratio than other compression techniques like Arithmetic coding, it is only appropriate for encoding text and program files and is ineffective for encoding digital pictures.

Conclusion

The purpose of image compression is to reduce storage requirements while retaining picture quality, which is eventually accomplished through Huffman and DCT. There are other compression strategies, however, Huffman coding demonstrates that it is an excellent compression approach for lossless picture compression. In Huffman, coding duplication may be reduced by assigning codes more effectively. Discrete Cosine Transform with Huffman Coding was employed in the proposed methodology, and better image quality with high PSNR value and MSE was achieved with a modest compression rate, however other studies demonstrate that there are alternative algorithms that surpass DCT and Huffman Coding and can reach maximum compression rate.