A REVIEW ON IMAGE COMPRESSION TECHNIQUES

Implementation and Performance Evaluation of Huffman Coding and Block Truncation Coding (BTC) for Image Compression in MATLAB

Gupta Mohil Amitkumar
Computer Science and Engineering
202151060@iiitvadodara.ac.in

Harsh D Makwana
Computer Science and Engineering
202151063@iiitvadodara.ac.in

Kale Atharva Anna
Computer Science and Engineering
202151069@iiitvadodara.ac.in

Kartavya Ravindrakumar Makwana
Computer Science and Engineering
202151072@iiitvadodara.ac.in

ABSTRACT: This paper provides a comprehensive review of diverse image types and various techniques employed in image compression. Through this thorough examination, we propose a general method for image compression that aims to reduce image size while maintaining image quality. Image compression serves as a crucial solution for the efficient transmission and storage of substantial amounts of digital image data. The applications of image transmission encompass areas such as television broadcasting, satellite-based remote sensing, and long-distance communication. Simultaneously, image storage is essential for preserving medical images, satellite imagery, documents, and pictures. This paper delves into different image types and explores a range of compression techniques, addressing the multifaceted challenges associated with the transmission and storage of digital images.

KEYWORDS: Image, Image compression technique, Huffman Coding,Compression, Decompression,Block Truncation coding,Lossless and lossy image compression.

INTRODUCTION

In the digital age, where the generation and consumption of images have become ubiquitous, the demand for efficient storage and transmission solutions is ever-growing. Image compression stands at the forefront of addressing this challenge, serving as a key technology to reduce file sizes while preserving image quality. This paper explores the intricate landscape of image compression, focusing on two fundamental techniques: lossless compression, exemplified by Huffman coding, and lossy compression, with a spotlight on Block Truncation Coding (BTC). Through a detailed examination of these techniques, we aim to provide insights into their mechanisms, applications, and trade-offs, contributing to a deeper understanding of image compression in the digital era.

Why Compression is needed and What are the Principles of compression?

Uncompressed images demand substantial memory in both RAM and storage media, posing challenges in terms of storage space and transfer times between devices.

The fundamental principle behind compression lies in the inherent redundancy within digital images, which are essentially arrays of pixel values. Compression algorithms aim to eliminate redundant bits, reducing image size significantly. Image compression comprises

two key components: redundancy reduction and irrelevant data reduction. Redundancy reduction involves the removal of extra or repeated bits, while irrelevant data reduction omits less critical information that is not essential for the receiver. Three types of redundancies exist:
Coding Redundancy: This occurs when a smaller number of code words are needed instead of larger symbols.
Pixel Redundancy: Arising from correlated pixels in an image.
Psycho-Visual Redundancy: Data ignored by the normal visual system. The goal of image compression is to minimise the number of bits representing the image, addressing these redundancies through a systematic approach.

TYPES OF IMAGES

A.GIF
Graphics Interchange Format (GIF) is useful for images that have less than 256 colours, grayscale.GIF is limited to an 8 bit or 256 colours. so that it can be used to store simple graphics ,logos and cartoon style images. It uses lossless compression.

B.PNG
The PNG (portable Network Graphics) file format supports 8 bit, 24 bit, 48 bit true colour with and without alpha channel. Lossless PNG format is best compared to lossy JPEG. Typically, an image in a PNG file can be 10% to 30% more compressed than in a GIF format.PNG format has smaller size and more colours compared to others.

C.JPEG
Joint Photographic Expert Group (JPEG) is a lossy compression technique to store 24 bit photographic images. It is widely accepted in multimedia and imaging industries. JPEG is a 24 bit colour format so it has millions of colours and is superior compared to others it is used for VGA(video graphics Array) display.JPEG has lossy compression and it supports 8 bit grayscale image and 24 bit colour images.

D. JPEG2000
JPEG 2000 is a compression standard for lossless and lossy storage.JPEG2000 improves the JPEG format. It is nearly the same as JPEG.

3. COMPRESSION ALGORITHM

There are two categories of compression algorithms: Lossless and Lossy. In Lossless compression, the compressed image perfectly replicates the original input image, with no loss whatsoever. Conversely, in Lossy compression, the compressed image differs from the input image, containing some degree of loss.

A. Lossless compression Techniques
Lossless image compression refers to the process of reducing the size of an image file without losing any original image data. This technique employs algorithms that preserve all image details during compression and decompression. It achieves compression by identifying and eliminating redundancies in the data without compromising image quality.By eliminating redundant information and storing only essential data, lossless compression ensures the exact reconstruction of the original image from the compressed file, making it valuable for applications where maintaining image fidelity is crucial, such as medical imaging, professional photography, and archival purposes.

1. Huffman Encoding
Huffman coding is a widely employed technique in lossless image compression, playing a pivotal role in reducing the overall file size while preserving the original image information. This algorithm operates on the principle of variable-length coding, assigning shorter codes to more frequently occurring pixel values and longer codes to less frequent

ones. In the context of image compression, Huffman coding begins by creating a frequency table that records the occurrence of each pixel value in the image. Subsequently, a binary tree, known as the Huffman tree, is constructed based on these frequencies, where shorter codes represent more common pixel values, and longer codes denote less common ones.

As the compression process unfolds, each pixel value in the image is replaced with its corresponding Huffman code. This results in a more efficient representation of the image, where commonly occurring pixel values are encoded with shorter binary sequences, contributing to a reduction in file size. Upon decompression, the original image is reconstructed faithfully by decoding the Huffman-coded binary sequence, ensuring a lossless retrieval of the initial image data. The adaptability of Huffman coding to the frequency distribution of pixel values makes it a powerful tool for achieving effective lossless image compression.

EXPERIMENTAL RESULTS:

Original Image (Scenery)

Compressed Image (Scenery)

Original Image (Game)

Compressed Image (Game)

Image Name : Scenery
Original Size : 246 kB
Compressed Size : 188 kB
Compression ratio : 1.31

Image Name : Game
Original Size : 70.3 kB
Compressed Size : 67.7 kB
Compression ratio : 1.04

Lossy Compression Techniques
Lossy image compression techniques are methods that reduce digital image file sizes by discarding less critical data imperceptible to the human eye. Unlike lossless compression, lossy techniques permanently remove certain image details using processes like quantization, discrete cosine transform (DCT), and subsampling. By sacrificing some image data, these methods achieve significantly smaller file sizes, ideal for web use, digital media distribution, and storage optimization.

Block Truncation coding

Block truncation coding (BTC) stands as a straightforward, swift, lossy, and fixed-length compression method designed for grayscale images. This approach, proposed by Delp and Mitchell employs a block-adaptive binary encoding technique grounded in moment-preserving quantization. The BTC procedure entails initially dividing an image into uniform n × n blocks, (generally n = 4) with each block being independently coded. The grayscale levels within each image block undergo quantization using a Q-level quantizer. The quantizer levels are chosen to preserve a select few low-order moments in the quantized output. In its basic form, BTC maintains the first two moments, representing each block with two quantization levels. Through the inclusion of additional constraints, higher-order moments can also be retained. Let k (equal to n^2) denote the number of pixels in a block, and let $f(x_i)$, where $x_i \in C$, represent the grayscale values of pixels within an original image block, with C denoting the set of pixel coordinates within the block.

$C = \{x_1, x_2, x_3, ... x_k\}$ the first two sample moments are given by

$$m_1 = \frac{1}{k}\sum_{i=1}^{k} f(x_i) \qquad (1)$$

$$m_2 = \frac{1}{k}\sum_{i=1}^{k} f(x_i)^2 \qquad (2)$$

m_1 is the sample mean and the sample variance (σ^2) of the image block is given by

$$\sigma^2 = (m_2^2 - m_1^2) \qquad (3)$$

If the pixel value of each block is greater than or equal to mean, it is represented by 1 and if less than the mean, it is represented by 0. The collection of 1s or 0s for each block is called a bit plane. In BTC, two statistical moments a and b are computed using the equations (4) and (5) and are preserved along with the bit plane for reconstructing the image. The compressed image is transmitted or stored as a set {B, a, b}

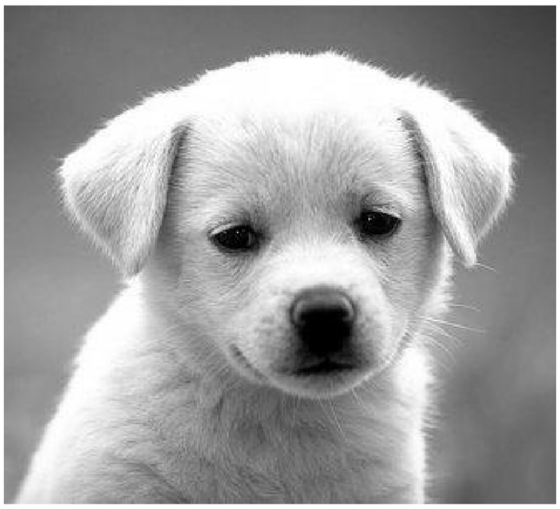
$$a = \bar{x} - \sigma\sqrt{\frac{q}{m-q}} \qquad (4)$$

$$b = \bar{x} + \sigma\sqrt{\frac{q}{m-q}} \qquad (5)$$

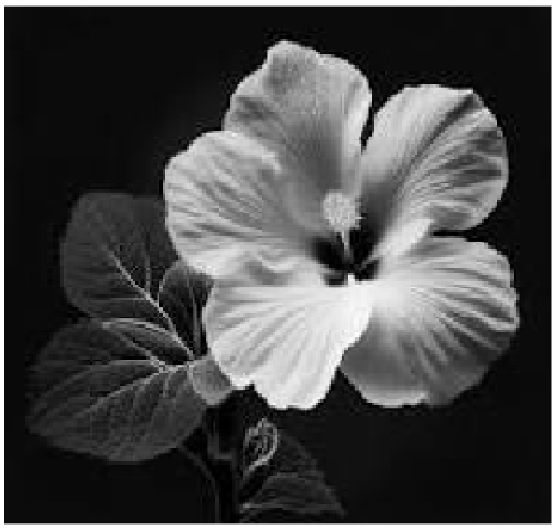
Where, q is the number of pixel values greater than or equal to X, and (m-q) is the number of pixels whose gray levels are less than X. While reconstructing the image, the 0 in the bit plane is replaced by a and the 1 in the bit plane is replaced by b.

EXPERIMENTAL RESULTS:

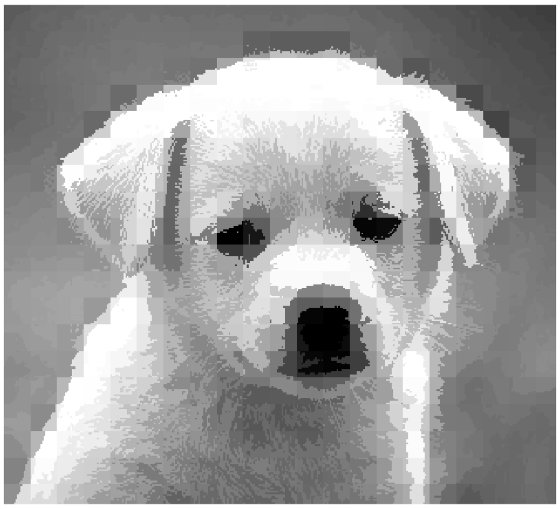
Original Image (Dog)



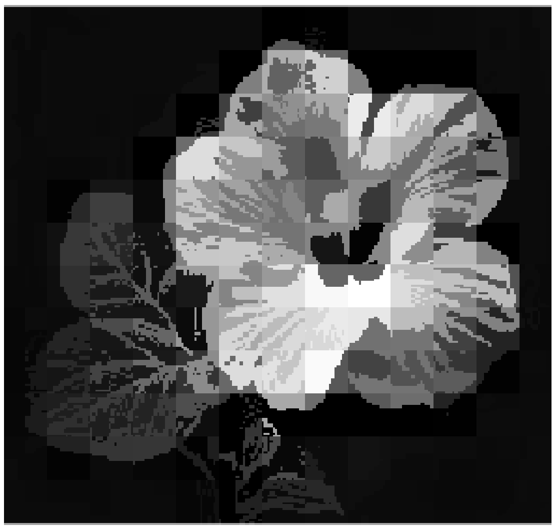
Original Image (Flower)



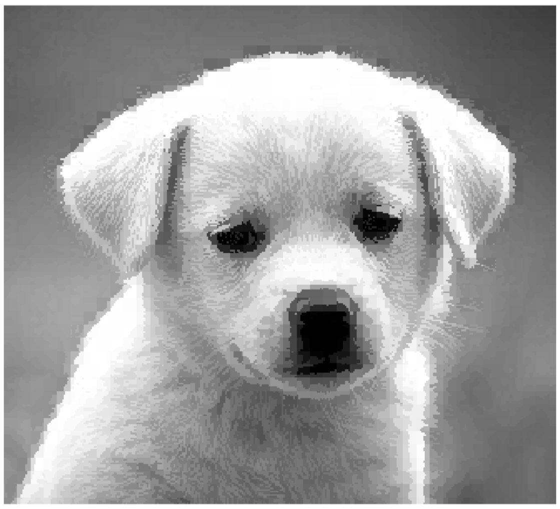
Compressed Image (Dog : n = 16)



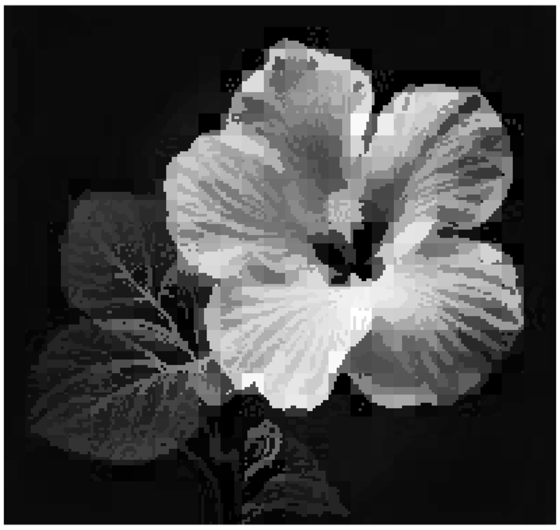
Compressed Image (Flower : n = 16)



Compressed Image (Dog : n = 8)



Compressed Image (Flower : n = 8)



Block size (n = 16)

Image Name : Dog
Original Size : 110 kB
Compressed Size : 36 kB
Compression ratio : 3.05

Block size (n = 8)

Image Name : Dog
Original Size : 110 kB
Compressed Size : 51.5 kB
Compression ratio : 2.13

Block size (n = 16)

Image Name : Flower
Original Size : 40.4 kB
Compressed Size : 10.7 kB
Compression ratio : 3.77

Block size (n = 8)

Image Name : Flower
Original Size : 40.4 kB
Compressed Size : 23.9 kB
Compression ratio : 1.69

OBSERVATION

The results of applying Huffman coding to two different images demonstrate varying levels of effectiveness in terms of compression ratios. For the first image, originally sized at 246 kB, the compressed version achieved a size of 188 kB, resulting in a compression ratio of approximately 1.31. This suggests a relatively efficient compression, indicating that Huffman coding was able to significantly reduce the file size while preserving the essential information in the image.

On the other hand, the second image, with an initial size of 70.3 kB, yielded a compressed file of 67.7 kB, leading to a compression ratio of approximately 1.04. Although there is still a reduction in size, the compression ratio is lower compared to the first image. This implies that the second image may have had characteristics that were less amenable to compression using Huffman coding. It's worth noting that certain types of images or data may not benefit as much from this particular compression technique, and other methods or algorithms may be more suitable.

In conclusion, the effectiveness of Huffman coding for image compression depends on the nature of the image data. While it demonstrated notable compression for the first image, the second image exhibited a more modest reduction in size. Further exploration and experimentation with alternative compression techniques could be considered to optimise the compression performance for different types of images.

In employing the Block Truncation Coding (BTC) algorithm for image compression with varying block sizes, distinct compression ratios and outcomes were observed. For the initial image of 110 kB, compression with a block size of 16 yielded a compressed size of 36 kB, resulting in a compression ratio of 3.05. Subsequently, using a block size of 8, the compressed image size increased to 51.5 kB, and the compression ratio decreased to 2.13. The second image, originally 40.4 kB, achieved a compression ratio of 3.77 with a block size of 16, producing a compressed size of 10.7 kB. However, using a block size of 8 for the same image led to a compressed size of 23.9 kB and a compression ratio of 1.69.

The block size defines the dimensions of these non-overlapping blocks. A larger block size means that more pixels are considered together, providing a higher level of spatial averaging within each block. As a result, larger block sizes often lead to more effective compression because the redundancy within the blocks is exploited more efficiently.

However, the choice of block size involves a trade-off. While larger block sizes generally result in higher compression ratios, they may lead to loss of fine details in the image. On the other hand, smaller block sizes preserve more local details but might not achieve as high compression ratios.

REFERENCES

[1] R. Patel, V. Kumar, V. Tyagi and V. Asthana, "A fast and improved Image Compression technique using Huffman coding," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 2016, pp. 2283-2286, doi: 10.1109/WiSPNET.2016.7566549.

[2] Mohammed, Doaa, and Fatma Abou-Chadi. "Image compression using block truncation coding." Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), February Edition (2011)