Approximate Compression – Enhancing compressibility through data approximation

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Harini Suresh

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

John Sartori

November, 2015

© Harini Suresh 2015
ALL RIGHTS RESERVED

Acknowledgements

I would like to express my heartfelt gratitude to my advisor, Professor John Sartori, for guiding me through this thesis and his continuous support. His insightful remarks have greatly helped in improving the quality of this thesis, right from setting new directions of research to the presentation of the work.

I would also like to thank Professor Keshab Parhi and Professor Pen Chung Yew for taking time off their schedule and serving on the thesis committee.

I would like to thank my friends – Layamrudhaa Venkatesan, Hari Cherupalli, Vinayak Srinath, Sneha Deshapande, Ashwin Nagarajan, Krithika Rai, Vaishnavi Santhapuram, Yokesh Ramasamy and Karthikeyan Thavasi for the interesting discussions and memories created, throughout my Master's.

Finally, I would like to thank my family for their unconditional love and support.

Dedication

In the memory of my Uncle, Sriram.

Abstract

The implicit noise tolerance of emerging Recognition, Mining and Synthesis (RMS) applications provides the liberty from conforming to the "correct" output. This attribute can be exploited by introducing inaccuracies to the datasets, to achieve performance benefits. Data compression provides better utilization of the available bandwidth for communication. Higher gains in compression can be achieved by understanding the characteristics of the input data stream and the application it is intended to be used for. We introduce simple approximations to the input data stream, to enhance the performance of existing lossless compression algorithms by gradually and efficiently trading off output quality. For different classes of images, we explain the interaction between the compression ratio and the output quality, time consumed for approximation, compression, and decompression. This thesis demonstrates and quantifies the improvement in compression ratios of lossless compression algorithms with approximation, compared to the state-of-the-art lossy compression algorithms.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The pervasion of technology in our lives, in different form factors ranging from desktops, laptops to mobile computing, has seen a tremendous shift in the types of applications preferred by the user base. Widely used applications enabling Voice over IP, i.e.,communication through images, video and audio messages over Internet Protocol (IP) networks, have increased the amount of multimedia data to be processed. With the advancements in lenses and cameras, low-resolution images with 352x288 pixels are now being replaced with millions of pixels. For example, iPhone6s features a 12MP iSight camera, about 120x more pixels. The explosion of digital data, from the perspectives of technological advancement and ubiquity, has resulted in higher demands for increased storage and bandwidth for communication. Emerging trends in computing like cloud computing, mobile computing, internet of things, etc. have increased the stress on bandwidth. At the same time, improvements in communication data rates and storage technologies have been offset by the manifold increase in data to be processed and transmitted.

Improved utilization of communication bandwidth and storage capacity can be obtained by data compression. The two types of compression are lossless and lossy compression. Lossless compression works by eliminating redundancy from the input stream and enables a faithful reproduction of the input at the receiver. Lossy compression achieves higher compression by removing data that is irrelevant to user perception, in addition to redundant data from the input stream. The necessity for decompression at the receiver is contingent on whether the application can directly work with the

compressed file.

Medical and scientific applications with the pursuit of the unknown are best served by lossless compression. Lossy compression is employed in applications like graphics, image, audio, and video processing, that can tolerate relaxation in output quality. These applications exhibit the property of implicit noise tolerance – the ability to tolerate inaccuracies in the output [4]. The introduced approximations may not be perceived by the user, say, while watching movies. Alternately, the user may have to accept a compromise on the accuracy, by way of losing display brightness and image capturing functionality, while operating laptops and mobile phones in power saving mode.

With the trends in computing shifting towards mobile-based and cloud-based computing, data is being captured and transmitted at an enormous rate. To work within the restrictions imposed by the bandwidth of the communication links, data to be transmitted is compressed in advance. Despite the advancements across generations of wireless communication standards, the bandwidth for each generation is fixed and is unable to match the enormity of the data that we handle. This causes us to experience delays in video streaming, webpage load times, etc, while working with compressed data. From a survey conducted on webpages that can adapt their layout to devices of different form factors, on an average, images have been found to occupy about 1 MB of data in a webpage of 1.7 MB. To the mobile user, these webpages place high demands on the bandwidth of internet connections, the battery, and the memory of the device. Figure 1.1 shows that images constitute 50-60% of the overall page weight, where page weight refers to the size of the page in bytes. When the page weight exceeds 1.7 MB, the load times become unacceptable. A slow webpage load time can significantly reduce the possibility of the user to come back to the webpage.

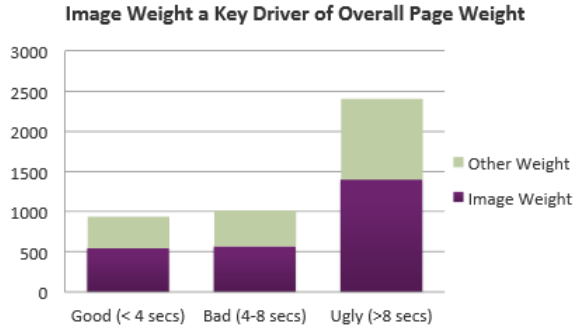


Figure 1.1: Attention to image size is essential for faster loading of webpages [1]

This thesis revisits data compression for noise tolerant applications with the perspective of approximation, to maximize compression for every unit of quality sacrificed. Specifically, approximations are introduced to the input data stream with the understanding of its characteristics, to enhance its compressibility. Our approximate compression techniques are compared against existing lossy compression techniques. Controlling the extent of approximation trades off output quality with the size of the compressed file. Higher compression leads to better bandwidth utilization and lower latency.

Determining the extent to which quality can be traded off for improvement in compression ratio may be challenging, as the decision may often depend on user preferences and may differ based on the application. As such, we provide techniques that allow fluid trade-offs between quality and compression ratio, enabling a user to select a desired point in the tradeoff space depending on desired quality and/or compression.

This thesis provides a suite of approximation techniques for enhancing the performance of existing lossless compression algorithms and a quantification of the benefits in compression obtained by approximation, with reference to the original, unapproximated images. Images are classified based on their characteristics and guidelines are provided to enable the choice of a suitable approximation technique and a lossless compression algorithm, based on the user's requirements of quality, compression ratio and the times to approximate, compress and decompress.

The thesis is organized as follows:

- Chapter 2 provides the background for this thesis, relevant work in the areas of lossless and lossy compression, and the motivation behind the problem statement.

- Chapter 3 explains the working of the proposed approximation schemes.
- Chapter 4 describes the images used for testing, the choice of metrics behind the objective evaluation of output quality, and the performance of approximation techniques and lossless compression algorithms.
- Chapter 5 provides results and analysis, presenting different tradeoffs between compression ratio, output quality, and time taken for approximation, compression, and decompression.
- Chapter 6 concludes the thesis and provides ideas for future work.

Chapter 2

Background and Related Work

Data compression reduces the number of bits required to store or transmit information from an input data stream. The recovery of the information from the compressed data stream is the corresponding decompression process. Compression ratio is the ratio of the size of the original, uncompressed file to the size of the compressed file. A higher compression ratio implies a smaller compressed file.

2.1 Information Theory

Information theory introduced by Claude Shannon in the late 1940s quantifies information from an event in terms of the probability of occurrence of the event (p).

$$Information\ gained\ from\ the\ occurrence\ of\ an\ event = -log2(p) \tag{2.1}$$

Hence, there is a lot of information to be gained from an unlikely event. Suppose, a source transmits n identically, independently distributed (i.i.d.) symbols with probability of occurrence $p_1, p_2, ...p_n$. The entropy of the source, H is given by 2.2

$$H = \sum_{i=1}^{n} (p_i * -log_2(p_i)) \tag{2.2}$$

Since the entropy of the message is related to the information content present per symbol from the source, H also represents the lower bound on the number of bits to be used to encode the symbols, to enable them to be uniquely decoded at the receiver.

An encoding scheme is said to be *optimal* when it achieves coding at the number of bits equivalent to the entropy of the symbol set. *Redundancy* is the difference between the symbol set's largest possible entropy and the actual entropy. When data has zero redundancy, the file is said to be completely compressed [5].

2.2 Types of data compression

Data compression can be classified into lossless and lossy compression methods. Lossless compression enables the receiver to reconstruct the original message, without loss of information. Lossy compression compromises on the accuracy of the information to achieve a better compression.

2.2.1 Lossless compression

The two main types of lossless compression methods are Statistical compression and Dictionary-based compression methods.

Statistical compression

Statistical compression methods encode data by assigning shorter codes to frequently-occurring symbols and longer codes to rarely-occurring symbols. The encoding scheme is called entropy encoding, since the number of bits used to code doesnt exceed the entropy of the system. The two most widely used entropy encoding schemes are Huffman and Arithmetic encoding.

The variable-length codes used in statistical compression are to be designed with the prefix property, which would enable the unique decoding of the codes at the receiver. Prefix property of the code states that on assigning a string with a code, the same code cannot prefix another code. For example, if a code of 0 has been assigned to represent a letter, the codes for the other letters cannot start with 0.

Statistical compression was first used in 1836, in the design of Morse code to transmit messages through an electric telegraph system. Morse code is a variable length code that encodes letters and numbers in the form of dots and dashes, with frequently-used letters like vowels receiving shorter codes and rarely-used letters like X, Q receiving

longer codes. The goal behind this scheme is to reduce the length of Morse encoded messages.

In 1955, Predictive Coding [6] emerged as a solution for efficient coding for systems that have an inherent predictability of data. The knowledge of the past symbols transmitted by the source is used to predict the current symbol. The coding systems replaced the original value with the predicted value, when the prediction was made within a threshold of the original value, and the error was transmitted. With the same prediction made at the receiver based on the incoming stream and the error information in the encoded stream, the original symbol is recovered.

Modeling for statistical compression

A modeling stage precedes the entropy encoder, which provides the encoder with the probability of the next symbol. There are different types of modeling that can be used – finite context models, ergodic, grammar based, etc. The most widely used model is finite context modeling, which predicts the relative frequencies of the symbols from the context. A zeroth order model considers each symbol equally likely and statistically independent of the other symbols. A first order model considers each symbol statistically independent of the others, but with a skewed probability distribution, where some characters are more likely to occur than others. For example, in English, the vowels are more likely to occur than most of the other letters. Order n models consider the context in which the symbol appears, done by modeling each symbol's probability of occurrence as being dependent on the previous n symbols and independent of all the symbols occurring before the n symbols.

A static context-based modeler contains all possible contexts of the input along with their probabilities of occurrences and is known to both the encoder and decoder. Say, if we are to define a probability model for order 2 contexts of English alphabets, all possible digrams are tabulated along with their probabilities, based on having read large texts. This model does not depend on the input stream.

A semi-adaptive model constructs a fixed probability model from the input stream to be compressed. The constructed model needs to be sent to the decoder along with the compressed file.

Adaptive context-based modelers build the probability model online, as the input data is being read. Since the model is built specific to the input symbol and its context,

it provides a greater accuracy of probabilities. They compute the conditional probability of the symbol, knowing the context – P(X—A) – where X is the current symbol and A is the context in which X occurs. The adaptive models start out knowing nothing about the input data. As more data is read, they start computing the probabilities based on inferences from the contexts encountered so far and they weigh recently-read data more than old data. Since the model is updated as the input data is encoded, to accurately reconstruct the output at the receiver, the decoding process also has to update the tables in the same way, as was done during compression.

The entropy encoding techniques encode the input stream based on the frequency or the probability of occurrence of the symbols. The encoding techniques can be classified as static or dynamic. Static encoding schemes have a fixed relationship between the symbols and the codes. Dynamic or Adaptive encoding techniques change the mapping between symbols and codes with time. If in an input stream a symbol appears frequently towards the beginning and is not present in the rest of the file, dynamic encoding assigns the symbol a shorter code at first, and later the short code would be mapped to a different symbol that appears frequently at that point in time. The decoder must have the same model as the encoder. An adaptive modeling and encoding scheme is as shown in Figure 2.1

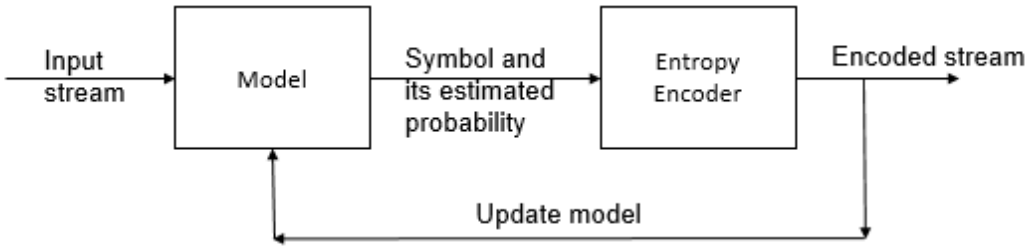


Figure 2.1: An adaptive modeling and encoding scheme updates the probability message as an input stream is encoded.

Hence, the are many ways to model data for encoding; all of them could be used with the same coding technique and yet give different compression ratios. The probabilities generated are model-specific. The compression ratio depends on how accurate the

probability estimated by the model is, correspondingly, how close the minimum code length generated by the entropy encoder is to the entropy of the input stream.

Dictionary-based Compression

Dictionary-based compression methods use a dictionary that holds strings of symbols. They read strings of symbols from the input file and try to match the strings with those present in the dictionary. If there is a match, the string is replaced with a reference to the dictionary, called a token, instead of a code for the symbol. Longer repeated strings provide a better compression ratio. If a match with the dictionary is not found, the uncompressed data is written. For better compression, the stream of tokens and uncompressed data maybe further compressed using entropy encoders like Huffman or Arithmetic encoders.

The dictionary can be static or adaptive. Static dictionaries can add strings of symbols but do not permit their deletion. Hence, a static dictionary would work well for circumstances where the possible strings in the input stream are limited and known beforehand. Adaptive dictionaries permit additions to or deletions from the dictionary as the input is being read. In case of adaptive dictionaries, the method starts with an empty dictionary. The input stream is broken down and compared against existing phrases in the dictionary; if a match exists, the token is sent out, else the uncompressed stream is sent out and is added to the dictionary. When the dictionary gets too big, the old data is deleted from the dictionary.

Since the compressed stream consists of tokens and uncompressed data, the principle behind the working of the decoder to recover the original stream is to differentiate between uncompressed data and the tokens. The construction of the decoder is not complicated as in the case of statistical compression, where the decoder is of the same functionality as the encoder. There are different methods for dictionary-based compression, which differ on how the dictionary is built [7].

2.2.2 Lossy compression

Lossy compression methods are applicable only for applications in which perceptual inaccuracies in the output are permissible. Typical applications of lossy compression include image, video, and audio compression. The loss of output fidelity from the

original file is either imperceptible to the user or has been accepted by the user for the application, and hence, can be traded off for a higher compression ratio.

The difference between the input stream to be compressed and the reconstructed stream after decompression is called *distortion*. The goal of lossy compression is to maximize the compression, i.e., minimize the average number of bits to encode the input stream, within the given distortion limits. Rate-distortion theory describes how much of the information from the source needs to be captured to obtain a specified distortion.

A typical rate-distortion function of a random variable X is as shown in Figure 2.2. The rate of encoding is measured in number of bits per symbol. When the distortion is 0, the number of bits required to code is equivalent to the entropy of the symbols generated by the source, H(X). This implies lossless compression, since the best case for minimum number of bits required to encode equals the entropy of the symbols generated from the source.

The reduction in the number of bits required to encode the input stream to below its entropy, becomes feasible with the introduction of distortions to the input stream. The point indicated by R(0) on the Y-axis is representative of the rate at which the input stream is to be encoded to obtain an imperceptible loss of information after decompression.

Rate-distortion theory helps formulate the two types of problems lossy compression can be optimized for –

- Given a predetermined distortion, maximize the rate of encoding the input stream.
- Given the rate of encoding the input stream, minimize the distortion.

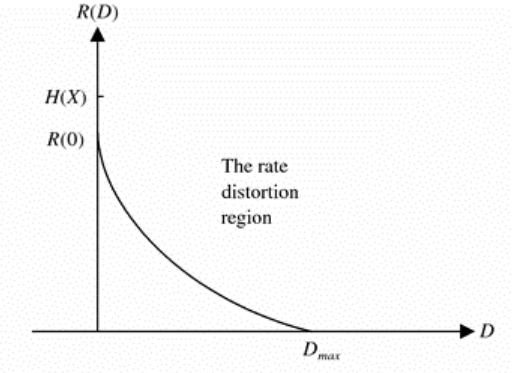


Figure 2.2: A typical rate distortion function R(D) [2]. R(D) traces the rate of encoding with the introduction of distortions, D to the input stream.

Lossy compression introduces losses through transformation and quantization of the input data, which are then losslessly compressed using a modeler and an encoder.

Transformation

Transformation algorithms like Discrete Cosine Transform, and Discrete Wavelet Transform are used to transform the input to a form of lower entropy, so that lesser bits can be used for coding. Transformation is done to identify components of the image that are irrelevant to human perception.

For instance, RGB color images are transformed to YCbCr space, which represents the Luminance, Chrominance-blue and Chrominance-red. In the RGB color space, the red, green and blue components are to be stored in equal bandwidth to identify a color. Since the human visual system is more sensitive to variations in brightness than in color, luminance is stored in greater detail than chrominance while transforming RGB to YCbCr. There are different YCbCr sampling formats like 4:4:4, 4:2:2, 4:2:0, 4:1:1.

In YCbCr 4:4:4 sampling format, every pixel in the YCbCr domain has a luminance, chrominance-red and chrominance-blue component. With a requirement of 8 bits to represent each component in consumer applications, every pixel needs 24 bits for representation.

In YCbCr 4:2:2 sampling format, for a pair of consecutive pixels, there is one chrominance-red and chrominance-blue component and each of them have a luminance

component. Hence 32 bits are needed for the representation of two pixels.

In YCbCr 4:2:0 sampling format, every block of 2x2 pixels shares one chrominance-red component and one chrominance-blue component, and has one luminance component for each pixel.

In YCbCr 4:1:1 sampling format, 4 consecutive pixels in a row share one chrominance-red and one chrominance-blue component and each of the pixels have their own luminance component. In JPEG Compression, the YCbCr representation is the input to Discrete Cosine Transformation stage, which converts the spatial information from the image to a frequency representation, which helps identify information that does not compromise on the image quality.

Quantization

Quantization algorithms reduce the distortions in the input stream by limiting the output values to a smaller set of unique values. Based on the data to which quantization is applied, there are two types of quantizations – scalar and vector quantizations.

Scalar quantizations map the domain of one-dimensional input values to a set of intervals, thus effectively reducing the spread in the values. The input values are replaced with midpoints of the interval. This is an irreversible, many-to-one mapping. Scalar quantizations are used to reduce the number of grayscale levels in an image. They can be uniform or non-uniform, depending on whether the input data has a uniform or a non-uniform distribution. If the input data has a non-uniform distribution, the levels can be selected to assign more intervals to regions with more values and fewer intervals to other regions.

Vector quantizations map multidimensional input vectors to a finite set of vectors. This can be used to reduce the number of colors in an image, by working with RGB vectors, instead of having to work with the red, green, and blue components individually.

Modeling for Entropy Encoding in lossy compression

The introduction of losses to the input stream is done by *transformations* and *quantizations*. The output is entropy encoded to achieve the minimum bit representation provided by the entropy of the symbols transmitted by the source. The entropy encoding schemes like Huffman and Arithmetic coding require the presence of models to estimate probability distribution of the symbols, to assign codes.

The models used in lossless compression schemes strive for an exact representation of the source and accuracy in the probability estimations of each symbol, to provide the best compression. In case of lossy compression, instead of having to work with the probability of each symbol, the source is conformed with one of the widely used models, because explicit formulas and procedures are available in literature to perform functions associated with the model. The models that are used in the design and analysis of lossy compression schemes are Uniform, Gaussian, Laplacian, and Gamma distributions [8].

The probability model based on a uniform distribution assigns equal probability to any symbol that the source generates. In case of Gaussian distribution, the probability that the source generates values that are within one standard deviation from the mean is about 68%. Laplacian distribution is more appropriate than Gaussian distribution when the distributions of the symbols that we deal with have the highest frequency at 0 and a lot of values close to 0. Gamma distribution is even more sharp peaked than Laplacian distribution.

When the dependencies between the symbols generated by the source cannot be modeled by the above probability models, models of the source based on the physics governing its functionality are used. For example, in case of speech production, the input is the air forced into the vocal cord and the output is the speech. The conversion of air into speech is decided by the shape and physics of the vocal tract, which can be modeled mathematically to predict dependencies within the generated speech signal [8].

2.3 Related Work

This section covers work relevant to the use of approximations in data compression.

A Mathematical Theory of Communication [5], put forth by Shannon in 1948, presented a quantification of the amount of information one gains from a message and

introduced a concept of entropy for a message, which gives a lower limit on the number of bits required to code the message losslessly.

Lossless compression is typically comprised of two stages – modeling and encoding. The modeling stage provides the probability distribution for each of the predicted symbols in the input stream to the entropy encoder. If the modeling is context-based, it takes into account the previous characters that have been observed while assigning a probability to the current symbol. The longer the context, a better probability estimate for the current symbol could be obtained. The number of such contexts that needs to be stored from an input stream, with increase in the length of the contexts, grows exponentially. With adaptive techniques, the probability values assigned to symbols are updated dynamically, as they are further encountered in different contexts while reading the input stream. Different lossless compression methods assign probabilities differently to the symbols, which determines the compression efficiency.

Predicting the probability of a symbol based on the context in which it occurs helps remove more redundancy by capturing higher order statistics of an input stream. The family of Prediction by Partial Matching (PPM) algorithms [9, 10, 11] generates the probability distributions based on a set of finite-length contexts. Given the current symbol and the contexts in which has occurred, PPM methods generate a conditional probability of occurrence of the symbol, given the present context – $P(symbol|context)$. The generated probability distribution is based on weighting the probability estimates from the different context lengths, longer contexts being assigned higher weights.

If the current symbol has never been seen before, PPM switches to an order -1 context and sends N+1 escape characters followed by the encoding of the symbol with a fixed probability [12]. The potential reduction of compression efficiency due to the presence of escape characters is mitigated by Prediction by Partial Approximate Matching (PPAM) [13]. PPAM starts off by searching for exact matches to the current context. If the number of exact contexts is below a predetermined threshold, the search is on for an approximate, longer context. PPAM introduces approximations in the context matching to gain better compression, yet allows lossless recovery of the encoded sequence. For image compression, the noise in pixel data compared to the highly redundant language data reduces the length of context matches which can be obtained by PPM. PPAM obtains longer context lengths with approximate context matching, providing greater

compression for images.

Approximation of the input stream to remove redundant and irrelevant data is the basis of lossy compression algorithms, which typically achieve higher compression ratios than lossless algorithms. Compression of images takes place only when there is redundancy in the image, equivalently, correlation between pixels. By transforming an image to a representation where the dependency between the pixels is eliminated, i.e., the pixels are decorrelated, the symbols in the equivalent representation can be encoded independently.

Transforms such as Discrete Fourier, Cosine, and Wavelet identify terms that can be approximated with minimal effect on output quality. DFT, DCT represent the pixels of the image by the transform coefficients, that correspond to different spatial frequencies. A segment of the image with small details and sharp edges contains a high spatial frequency. The principle behind quantization of the coefficients to obtain compression is that the human eye is more sensitive to variations in the lower frequency regions than in the higher frequency regions. DWT captures both spatial and temporal information. There has been a lot of research on introducing approximations to these transforms to reduce the complexity of implementation, thus trading off accuracy [14, 15]. Integer approximations to Fourier Transforms have also been attempted, and the approximated transformation has been shown to reconstruct the input perfectly [16]. A methodology to approximate Discrete Sinusoidal transforms resulting in integer coefficients has been applied to Discrete Fourier, Hartley, and Cosine transforms [17]. The benefits of the integer approximations to transforms are in the lower multiplicative complexity of the computations compared to traditional approaches.

The use of approximate hardware to trade off accuracy to reduce power consumption has been investigated through the design of approximate circuit blocks. For example, the authors in [18] propose the design of imprecise adders by affecting the number of transistors in an adder block and its load capacitance. Using approximate addition for the least significant bits for the computations involved in discrete cosine transform (DCT) and Inverse DCT blocks for image compression reveals savings in power and area without an appreciable loss of quality. The design of approximate multipliers has been approached from different angles, such as using approximate adders to sum the partial products [19], the design of approximate compressors [20] and the use of an inaccurate

2x2 multiplier as a building block for a larger multiplier [21].

Existing work introduces approximations to the working of the modeling scheme for encoding to improve the probability estimates, as in the case of PPAM; to the frequency domain transformation coefficients of pixels in DCT to achieve higher compression by identifying irrelevant data, and to reduce the computational complexity of DFT and DCT, by converting floating-point arithmetic to integer arithmetic and the use of approximate hardware. This thesis proposes introducing software-level approximations to the input stream of pixels, by mapping the pixels from the image to a limited set, thereby increasing their frequencies of occurrence, which leads to a shorter code assignment by the entropy encoder, enhancing lossless compression.

2.4 Motivation

Digital images are represented by pixels along m rows and n columns, where m x n is the resolution of the image. The number of bits associated with each pixel decides the type of image. For example, one bit per pixel describes a monochrome image, whereas multiple bits per pixel can be used to represent different shades of colors or gray.

Continuous-tone images exhibit a continuous change of color (or gray) as the eye moves along any dimension. A sample of a region from a continuous-tone image is shown in Figure 2.3(a).

Natural photography typically falls under the category of continuous-tone images. There has been extensive research relating to the lossy and lossless compression of continuous-tone images.

Discrete-tone images are artificially generated images, where pixels values are within a small set of discrete values. An example of a discrete-tone image is a screenshot of the display from a mobile phone. A sample of a region from a discrete-tone image is shown in Figure 2.3(b).

Discrete-tone images have adjacent pixels that are either identical to the current pixel or are very different in value from the current pixel. Due to this large variance between adjacent pixels, any losses introduced in the image can significantly change the perception of the image. The presence of higher redundancy in discrete-tone images has been capitalized by Flexible Automatic Block Decomposition (FABD), which divides

the image into blocks that are repetitive, blocks composed of a single color and blocks that fall into neither of the above two categories, for efficient entropy encoding. The redundancy in the image is exploited by expressing the repetitive blocks in terms of themselves. Blocks with large solid regions are efficiently coded as flood fills. The blocks of pixels which cannot be categorized into the above are encoded individually. FABD provides 1.5x - 5.5x higher compression than GIF [22].

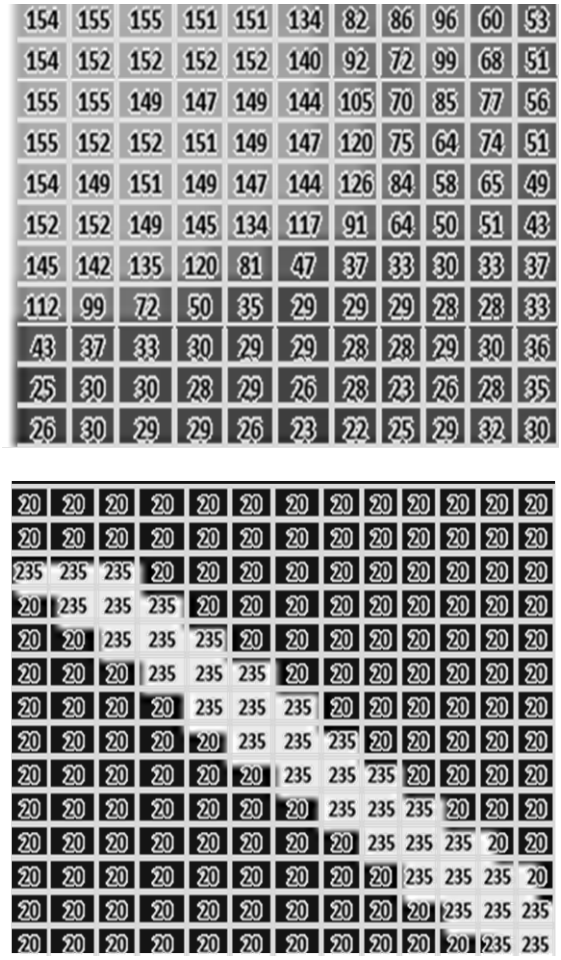


Figure 2.3: Top: (a) Continuous-tone images show a continuous change of color (b) The adjacent pixels in discrete-tone images either are very similar or are very different in value

The principle used in continuous-tone image compression is that there is a high

probability of the surrounding pixels of a given pixel to have similar values. This correlation between neighboring pixels is called spatial redundancy. Lossless image compression compresses images by the removal of redundancy. There are typically two stages in lossless image compression – modeling and encoding. The encoder assigns shorter codes to frequently occurring symbols and longer codes to the rarely occurring symbols, where symbol refers to the constituents of the data stream. To distinguish between which symbols occur frequently and which do not, we need a modeling element which gathers information about the image, in the form of a probabilistic model. The performance difference between compression algorithms arises in the modeling phase, since they are followed up with entropy encoders most of the times, to assign codes with the minimum length of the code close to the entropy of the message.

The models used to generate the probability information for images should consider the correlations between pixels to achieve high compression. Prediction by Partial Matching (PPM) algorithms predict the probability of the current symbol based on the context from previously encountered data. PPM algorithms dominated the compression benchmark tests in 1990s in terms of compression ratio. PAQ is an archiver that has evolved from PPM algorithms which uses context mixing followed by arithmetic encoding [23]. Predictions are made by a large number of models independently, which work with different contexts. Some of the contexts used by PAQ that are beneficial to image compression are based on:

- The previously encoded n bits, as in PPM.
- The consideration of the higher order bits of the previously encoded bytes.
- Two dimensional contexts for images and contexts based on the knowledge of how the compression was achieved (for example, in image files, the structure of the compressed image is based on which algorithm was used – JPEG, PNG, etc).

Context mixing is the process of sending a single probability distribution to the arithmetic encoder, by combining the generated contexts, accomplished in PAQ by training neural networks. The improved compression ratios achieved by PAQ algorithms come at the cost of increased time for compression and decompression and memory requirements due to the complexity of the modeling stage.

With the promise shown by the context mixing of the family of PAQ algorithms to provide good compression ratios for images, the goal of this thesis is to introduce approximations to gray scale, continuous-tone images, to enable an efficient trade off between the output quality and the compression ratio obtained from the existing algorithms providing lossless compression. By developing approximation schemes that take advantage of the different characteristics of the images to be compressed, higher gains in compression can be obtained. For example, the large, uniformly textured sections in continuous-tone images can be grouped together into blocks of pixels and concisely represented using a quadtree data structure. The regions of the images which do not exhibit much variation in the values of the adjacent pixels can be approximated using their average or their mode. The approximations made to the pixel data are targeted at property of the entropy encoder to assign shorter codes to frequently occurring symbols or by using an alternate means of representation of the pixel information.

Reducing the range of unique pixel values in an image to a discrete set of values enhances the performance of the modeling stage too. If the modeling is context based, with the increased frequency of repetition of pixel values, the estimated probability value is more accurate since it would have been computed from the results of many contexts.

Chapter 3

Approximation Techniques

This chapter describes the proposed approximation techniques that introduce controllable losses to images, which improve the performance of lossless compression algorithms. The proposed approximation and compression setup is shown in Figure 3.1. The approximation techniques are implemented in MATLAB, to be able to make use of the curve fitting toolbox and quadtree representations for images.

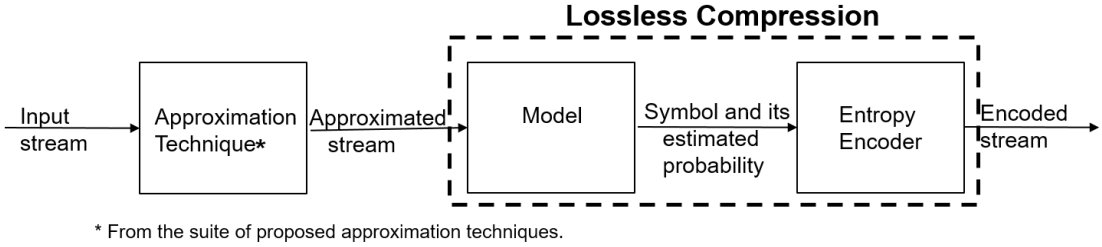


Figure 3.1: A suite of approximation techniques are proposed to enhance the compressibility of the input stream by lossless compression algorithms.

The developed approximation techniques can be classified into techniques that approximate the values of the pixels and preserve the structure of the image, and those that represent the information obtained from the pixels in a different format.

3.1 Pixel value modulation

With the understanding of how the entropy of the message is reduced by frequent repetition of letters and enhances the compression from Section 2.1, this section presents approximation techniques that modulate the input data stream to enhance its compressibility by lossless compression algorithms.

Pixel modulation techniques work by increasing the repetition of a few pixel values from the original image and minimizing the occurrences of the rest to 0, to enhance lossless compression. The approximation techniques proposed in this section are different ways of choosing representative values for the pixel distribution. There is no change in the size of the approximated file from the original file. On decompression, the pixels can be obtained without any requirement of further processing.

We introduce *distance of approximation* as a metric to control and understand the impact of approximation on compression. By sweeping the distance of approximation, we trade-off output quality for improvements in compression.

Midrange, mode and average-based approximations

One approach to identify representative values is to divide the entire spectrum of pixels into equal-sized ranges and use one value from each of the ranges to replace all the pixel values from within each range. The distance of approximation for these set of techniques is defined as the radius of each of the ranges, that the pixel distribution is divided into. A greater distance of approximation corresponds to choosing fewer values to represent the pixel distribution. This improves the compressibility because of the increased repetition of fewer unique values. Similarly, a lower distance of approximation chooses more representative values, providing a lower compression in comparison.

An elementary method of choosing pixel values from each range is to use the midpoint of each range as the representative for each range. This method is called the **Midrange-based** approximation scheme, illustrated in Figure 3.2.

This approximation technique has a very simple implementation. Since it chooses representative values that are independent of the underlying pixel distribution, the time taken to approximate does not vary with the image. This also increases the possibility of error.

To minimize the errors introduced by midrange-based approximation scheme, we propose choosing representative values that involve the underlying pixel distribution. After dividing the spectrum of pixels into equal-sized ranges, the pixel value that repeats the most from within each range is selected to represent the pixel values from the original image belonging to that range.This method is called the **Mode-based** approximation scheme, illustrated in Figure 3.2.

In case of encountering two values from a range with the highest frequency of occurrences, their mean is taken to represent the range.

Mode-based approximation provides increased accuracy as it considers the underlying pixel distribution. The computational complexity is greater than the mid-range based approximation, since it needs to identify the most frequently occurring pixel values to represent each range. The downside of this technique is that it provides an unequal weighting of the pixel values that belong to a range, since it looks at only the most frequently occurring value.

To provide an equal weighting of all the pixel values that occur within a range, the average of the pixel values from within each range is used to represent a range, as illustrated in Figure 3.2. This method is called the **Average-based** approximation scheme. Hence, average-based approximation is more accurate and has a greater complexity of implementation than mid-range based approximation scheme.

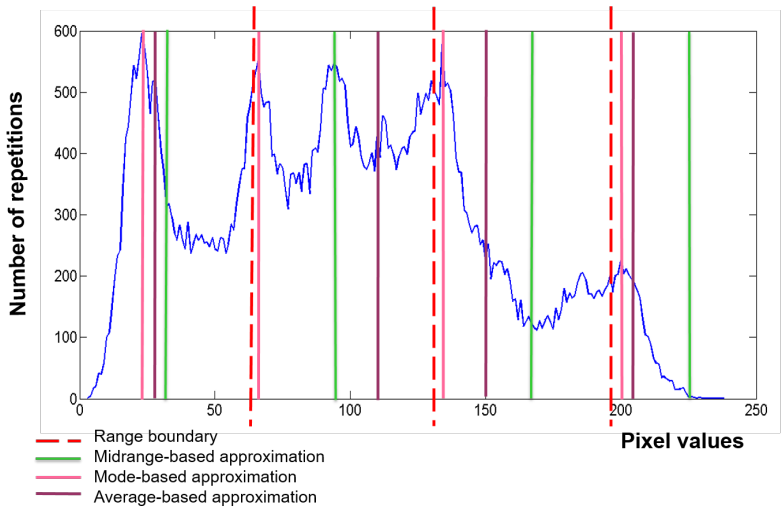


Figure 3.2: Mode-based and average-based approximation schemes provide better representative of pixel values compared to midrange-based approximation

NxN approximations

The approximation schemes considered so far - the midrange, mode and average-based, look at the image on a global perspective and fail to capture the local variations in the pixels in the image, resulting in a loss of accuracy. Hence the previously discussed approaches are applied to local blocks of pixel values from the image.

In this approach, the entire image is subdivided into blocks of size NxN and the above mean- and mode-based approximations are applied individually to each block. This method will result in the pixel values closer to the original values, since the average and mode are now localized.

The reduction in entropy, in comparison to the average and mode based approximations, is traded off for increased accuracy of the pixel values. The set of discrete values, which was earlier defined for the entire image, is now defined for each pixel block. With a higher value of N, the image is dominated by blocks that are bigger sized. But there are lesser blocks to approximate, hence presenting a high compression at the cost of loss in output quality. When the N is lowered, the number of blocks to approximate increases, implying smaller blocks. This results in the approximated images of good quality, but does not improve the compression by a lot. The optimum value of N was

found to be 8.

Clustering-based approximation

In the case of NxN approximation schemes, the variations in the image pixels are constrained by square blocks. Clustering-based approximation permits the formation of clusters of pixels with similar values. The maximum difference between the pixel values that can be permitted within a cluster is limited by the distance of approximation.

The pixels are traversed along the rows of the image and clusters are formed if the pixels are within the given distance of approximation from the starting pixel. The clusters are then replaced with the mean of the values belonging to the cluster.

The approximation introduced by this technique is mild, hence, the distortions do not dominate and the similarity with the original image is retained. This method doesn't provide a significant impact on the entropy of the image, so the increase in compression ratio provided by approximation is minimal.

Floodfill-based approximation

We observe that in an image, regions with uniform pixel values do not conform to a square or a horizontal scan-line as represented by the NxN and the clustering-based approximation schemes. Regions of pixels form similar-valued groups along multiple directions in the image. This observation is the basis for the development of floodfill-based approximation schemes.

The Floodfill algorithm provides the same color to an area of connected pixels. The connectivity of two pixels is decided based on spatial adjacency and pixel brightness values. This algorithm is used to fill colors for shapes in programs like Microsoft Paint.

Drawing a parallel from the floodfill algorithm, the proposed approximation scheme checks for pixels that share an edge with the pixel under consideration and replaces their values if they are within the distance of approximation.

To keep the structure of the image intact, using Canny edge detection, the edges in the pixels are identified beforehand and are left untouched by the approximation. This algorithm results in an image where the pixels of same value are localized. The ability to lower the entropy with increasing distance of approximation is contingent on the presence of regions of connected pixels with similar values in the original image.

An insight into the working of pixel modulating approximation schemes

Consider the following pixel data extracted from an image.

10 30 160 240 5 30 48

160 198 251 91 129 61 91

The working of the different approximation schemes are shown in Figure 3.3 for a distance of 32.

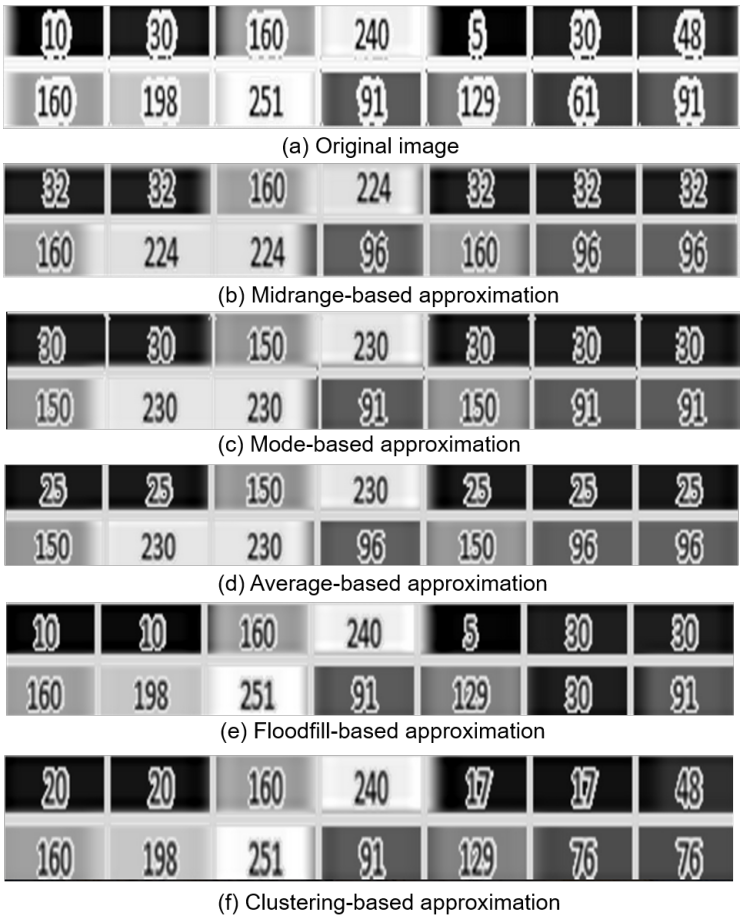


Figure 3.3: Demonstration of working of pixelvalue modulation based approximation techniques. From the top to bottom, the figures represent (a) Original image (b) Midrange-based approximation (c) Mode-based approximation (d) Average-based approximation (e) Floodfill-based approximation (f) Clustering-based approximation

For Midrange-based, mode-based and average-based approximation schemes, the representative values are chosen, as shown in Table 3.1.

Range of pixel values	**Midrange**	**Mode**	**Average**
0-64	32	30	25
65-128	96	91	96
129-192	160	150	150
193-255	224	230	230

Table 3.1: Representative values from different ranges for midrange, mode and average based approximation, for Example 3.1

3.2 Quadtree-based pixel representation

The pixel-modulation based approximation schemes discussed in the previous sections identify regions with uniform values of pixels and replace them with a single value. With the division of the images into blocks for more accurate approximation of the original image, this section is motivated by the idea of developing an alternate representation for the image by conveying the dimension of the blocks and information about the constituent pixels. This necessitates further processing after decompression of the compressed file, to obtain the pixel values. The size of the approximated file is typically smaller than the size of the original file, due to the alternate representation of data. Unlike the techniques that modulate pixel data aim to reduce entropy, these algorithms aim to reduce file size to be encoded by compact representation of the pixel data.

The developed techniques in this classification make use of a tree data structure called quadtree, which can efficiently represent large sections in the image with uniform pixel values. The requirements to reconstruct the image at the receiver are the dimension of each block along with the information about the pixel values within the block. A quadtree representation of an image recursively subdivides the image into quadrants, as long as a threshold on the minimum size or the variance of the resulting block is met. If the block has a variance exceeding the variance threshold and is at the minimum block size, it is not subdivided further. The process of quadtree formation and the approximation applied to each block can be made parallel, reducing the time

taken to approximate. A tradeoff between output quality and compression ratio could be obtained by sweeping the threshold value that determines whether to subdivide a quadrant.

Variance of a dataset is defined as the averaged sum of squares of differences from the mean. It is a measure of how spread out the numbers in the dataset are from the mean value. The variance threshold is used to identify regions with drastic changes in pixel values, so that their subdivision into quadrants could result in regions with more uniform pixel values.

Quadtree-based average approximation

For quadtree-based average approximation, the minimum permissible block size in the quadtree is set to 2 and the variance threshold of the block is defined. The resulting blocks of pixels after quadtree representation could then be replaced using the dimension of the block and the average value of pixels present in the block. A condition on the minimum block size is put forth to be able to achieve significant compression. For example, the smallest pixel block (2x2 pixels) is replaced by two numbers - the dimension of the block and its mean. This ensures that the compressed file is at most half the size of the original file.

Quadtree-based polynomial approximation

The quadtree based polynomial representation is an adaptive scheme that works based on the variance of the pixel block. The conditions for recursive quadrant subdivisions are that the minimum block size is set to 8x8 and the variance threshold of a pixel block is predefined. If the block is found to have a variance that is within the limits set by the threshold, a polynomial function that describes the pixels in the pixel block in terms of their positions is computed. The pixel values can be replaced by the degree and the coefficients of the polynomial for reconstruction at the receiver. Else, if the block has reached the minimum block size and cannot be split further, and yet the variance is higher than the threshold, the block of pixel values is transmitted without any approximation.

Polynomial-based approximation is implemented in MATLAB using the curve-fitting toolbox, which provides the method of Least Absolute Residuals (LAR). The LAR

method of curve fitting computes a curve that minimizes the absolute differences of the residuals. The outliers are treated equivalent to other data points. Hence, this method is a good choice for fitting a polynomial to the pixel values, since all the pixel values in a block are equally important.

The polynomial-based approximation method aims to reduce the number of data points to be encoded by transmitting only the degree of the polynomial and the corresponding coefficients. The minimum size of the block is 8x8 so that 64 pixels are considered at once. Since the polynomial coefficients are floating point numbers, for the worst case of having to fit a polynomial with degree 9 (giving rise to 10 coefficients), we replace the information obtained from 64 bytes of pixels with 41 bytes. (The 10 floating point polynomial coefficients requiring 10 * 4 bytes = 40 bytes and one byte for the degree of the polynomial). For the case where the pixel block is above the pre-determined variance threshold, the block remains unapproximated and is transmitted. This leads to inefficiency in compression, since there has to be an additional byte of information sent to indicate the length of the following uncompressed data.

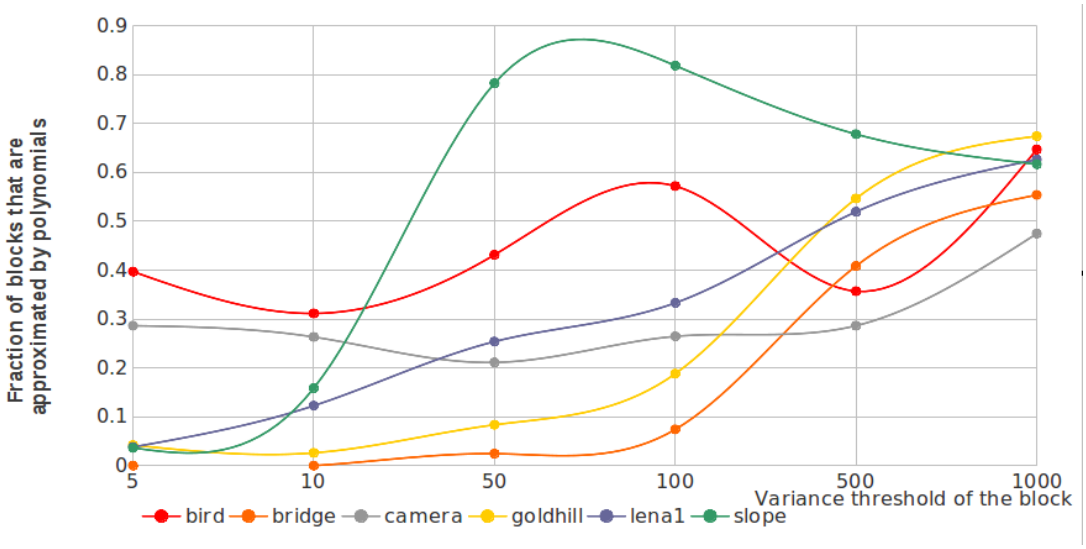


Figure 3.4: Images bird.tif and camera.tif permit polynomial-based approximation at a variance threshold of 5, implying they have regions with uniform pixel values.

Figure 3.4 shows a plot of fraction of blocks that are approximated using polynomials based on the variance threshold set for recursive subdivision of the blocks. The images used for analysis are described in Table 4.1. This plot helps make conclusions on the characteristics of the different images in the benchmark.

Quadtree based approximation with a variance threshold of 5 repeatedly subdivides the block until it reaches the dimensions of 8x8, or attains a variance lesser than 5. The observation that 0.4 and 0.3 percentage of the blocks are polynomial approximated implies the presence of blocks with uniform pixel distributions.

The other extreme, the images of bridge and goldhill allow polynomial approximation to take place only beyond the variance threshold of 50. This indicates the presence of regions which are dense, with many details that cannot tolerate approximations to meet the conditions on variance threshold.

With the increase in variance threshold, there is a decrease in the total number of blocks present in an image and hence, there is an increased number of blocks that undergo approximation, which can be concluded from the dip in the fraction of blocks that are not approximated.

The steepness of the plot of the image slope.tif is because of the increase in approximation of the blocks up to the variance threshold of 50, beyond which the reduction in total number of blocks the image gets divided into makes the fraction bigger. The image lena shows a proportional increase in the number of blocks which are below the threshold with the reduction in the total number of blocks in the image.

Chapter 4

Methodology

This chapter describes the methodology used in our evaluations.

4.1 Benchmarks

The natural images from GreySet1 of the Waterloo image repository [3] are used for testing. The images are of uncompressed tif format, with the resolution of 256 x 256 and 8 bit representation for pixels. The characteristics of the images used for analysis are summarized in Table 4.1.

The tests are performed as sequential processes on a node with a single Intel Xeon CPU X5675, running at 3.07 GHz.

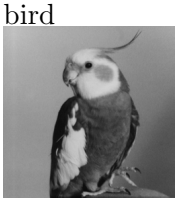
Image	First order Entropy	Characteristics
bird 	6.77	• Still image. • Composed of smooth textured regions – the background of the bird, the head, and mildly coarse textured regions like the feathers on the body of the bird.

Image	First order Entropy	Characteristics
bridge 	7.66	• Still image. • Highest entropy in the benchmark set. Compared to other images, the compression ratio would be the least. • The presence of various types of textures in different portions of the image help provide a camouflage effect to the distortions introduced by approximation techniques.
camera 	7.00	• Still image. • Image is composed of equal amounts of smooth textured regions – the coat, the sky and mildly coarse textured regions – the grass.
goldhill 	7.47	• Still image. • Image contains many edges, which serve as well-defined boundaries between the textured regions. • Image comprises of a lot of complex textures, which are localized to small regions.
Lena 	7.56	• Still image. • Image dominated by smooth sections – her face, body, and portions of the background. There are also mildly coarse textured regions – her hat and complex textured regions – the accessory on her hat.

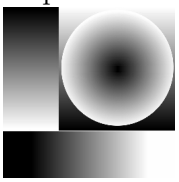
Image	First order Entropy	Characteristics
Slope 	7.51	• Autokinetic image – Image which appears differently when viewed from different angles. Limitations in user's perception could be capitalized by introducing losses while compressing autokinetic images. • This image has been artificially generated but the qualities of blur and noise are similar to natural images.

Table 4.1: Characterization of benchmark images

4.2 Lossless Compression Algorithms

This section discusses the lossless compression algorithms that are used to compress the approximated data generated from the various approximation algorithms.

The lossless algorithms considered are 7zip 15.09 [24], BZIP2 1.0.6 [25], CMIX-v7 [26], GZIP 1.6 [27], Infozip [28], LHA 1.14i [29], LZ4 [30], Nanozip 0.09 [31], LZIP 1.17 [32], LZOP 1.03 [33], PAQ 8o6 [34], RAR 5.21 [35], XZ 5.2.1 [36], ZLIB 1.2.8 [37], ZPAQ 7.05 [38], ZZIP v0.36c [39], PNG and GIF using NConvert [40]. The best compression ratio for all the images in the benchmark is given by CMIX – an open-source variant of the PAQ family of algorithms.

Table 4.2 categorizes the compression algorithms based on the base algorithm they have been derived from.

Table 4.3 shows the compression ratios provided by the different lossless compression algorithms on the original, unapproximated images from Table 4.1.

4.3 Metrics for Evaluation

The goal of this thesis is to introduce controlled approximations to the image to trade off quality for improved compression. This gives rise to the need for quantification of loss of quality and the compression performance to be able to evaluate the impact of

Base Algorithm	Description	Compressors that use the base algorithm
PAQ	Context mixing (modified PPM) and Arithmetic Coding	PAQ, ZPAQ, CMIX
BWT	Divides input stream into blocks, permutes within each block to group same symbols together, followed by RLE and Arithmetic coding	BZIP2, ZZIP
LZ77	Dictionary-based, uses pointers to refer to previous occurrences of the string	LZOP, LZ4
LZMA	LZ77 and Modified Arithmetic encoder	LZIP, XZ, 7ZIP
DEFLATE	LZ77 and Modified Huffman encoder	ZLIB, GZIP, ZZIP, 7ZIP, PNG
LHA	LZSS (Modified LZ77) followed by Huffman encoding	LHA
RAR	General mode uses Deflate, special mode uses PPM	RAR
LZW	Improvized LZ78, LZ78 assigns codes to repeating sequences, instead of back pointers to previous occurrences as in LZ78	GIF

Table 4.2: Categorization of lossless compression algorithms

Compression Algorithm	Switches	Test Images					
		Bird	Bridge	Camera	Goldhill	Lena	Slope
7zip	-m0=LZMA2	1.898	1.273	1.593	1.362	1.358	4.747
7zip	-m0=PPMd	2.041	1.193	1.629	1.316	1.325	4.923
BZIP2	–best	1.952	1.212	1.562	1.308	1.349	4.854
CMIX		2.716	1.395	1.984	1.535	1.741	8.587
GZIP	–best	1.570	1.069	1.353	1.130	1.113	3.672
INFOZIP	–best	1.562	1.065	1.347	1.126	1.110	3.630
LHA		1.588	1.079	1.388	1.144	1.141	3.668
LZ4	-9	1.279	1.005	1.175	1.009	1.022	3.107
NANOZIP(LZHD)	-cd	2.207	1.365	1.730	1.490	1.678	4.795
LZIP	–best	1.907	1.275	1.599	1.365	1.358	4.838
LZOP	-9	1.434	1.005	1.253	1.031	1.039	3.394
NANOZIP(NZCM)	-cc	2.027	1.338	1.614	1.437	1.539	3.079
PAQ 8o6	-7	2.624	1.372	1.943	1.511	1.711	7.566
RAR	-m5	1.818	1.266	1.443	1.388	1.384	3.652
XZ		1.902	1.275	1.595	1.365	1.357	4.792
ZLIB		1.580	1.077	1.387	1.154	1.117	3.624
ZPAQ	–best	2.218	1.289	1.756	1.409	1.451	6.738
ZZIP	-mx	2.001	1.221	1.586	1.316	1.345	5.288
NConvert(GIF)		1.382	0.858	1.184	0.959	0.923	2.229
NConvert(PNG)		1.541	1.057	1.355	1.131	1.096	3.434

Table 4.3: Compression of original, unapproximated images by different lossless algorithms

approximations. The metrics used for the evaluation of output quality and compression are discussed in the following subsections.

4.3.1 Output Quality

The metric used to assess the output quality of the approximated image should quantify of the degradation of the image due to approximations, with respect to the original image. In image processing, Mean Squared Error (MSE) and Peak Signal to Noise Ratio (PSNR) are the two widely used metrics to measure the quality of recovered images after lossy compression.

MSE is defined as the averaged sum of squared differences between the pixels of the original and the reconstructed image.

$$MSE = 1/n * \sum_{i=1}^{n} (p_i^2 - q_i^2)$$

where p and q represent the pixels from the original and reconstructed images and n is the total number of pixels in the image.

PSNR is defined as follows.

$$PSNR = 20 * log_{10} \frac{max(p_i)}{sqrt(MSE)}$$

where $\max(p_i)$ is the maximum permissible value for a pixel in the image. For a grayscale image using 16 bits to represent each pixel, $\max(p_i)$ equals $2^{16} = 65535$.

The closer the reconstructed image is to the original image, the smaller the MSE and the higher the PSNR. MSE and PSNR work with the absolute differences between the pixels in the images and hold no relationship with image perception by humans.

From the comparison of metrics shown in Figure 4.1 , all the images have a PSNR of around 20 dB and yet, appear visually different. The contrast-stretched image (second image), visually appears to be a good approximation to the original image, yet gets a lower PSNR metric compared to the images affected by noise (fourth, fifth and sixth images).

Structural Similarity (SSIM) is a metric that has been developed to quantify the visually perceived quality of images [41]. The human visual system perceives images by recognizing their structural information, which is provided by the pixels' inter-dependency

when they are spatially close. The SSIM algorithm processes the reference and distorted image in terms of pixel blocks of dimensions 8x8. Separate functional measures of luminance – l(x,y), chrominance – c(x,y), and structural similarity – s(x,y) are defined over the local, overlapping windows of the two signals x and y, and are weighted within each window. A general form of SSIM index is obtained by combining the three measures.

$$SSIM(x, y) = [l(x, y)]^{\alpha} * [c(x, y)]^{\beta} * [s(x, y)]^{\gamma}$$

where α, β, and γ are used to assign the relative importance to luminance, chrominance, and structural similarity measures. The SSIM rates the quality of images on a scale between -1 to 1. An SSIM value of 1 indicates identical images, whereas a value of -1 indicates complete uncorrelation between the reference and distorted image. Two signals are said to be uncorrelated, when their covariance is 0. Covariance is a measure of how much two random variables change together.

The quality score provided by SSIM can distinguish between the images, as shown in Figure 4.1. Images that are affected by noise (Fourth, fifth and sixth images) get a quality score close to 0, even though the bird in these images is discernible. In the third image of the series, inspite of the visibility of the distortions introduced by approximations, the quality score is higher than what is assigned for the images affected by noise. Hence, SSIM does not correlate well with human perception for these set of images.

Multi Scale Structural Similarity index (MS-SSIM) [42] extends the SSIM technique by evaluating the image at multiple resolutions and viewing distances. MS-SSIM works by computing the SSIM over many iterations by successively down-sampling the reference and distorted images by a factor of two. The results obtained from the different scales of evaluation are weighted differently to give an overall score. MS-SSIM has been shown to align more closely with subjective evaluations than SSIM and PSNR [42].

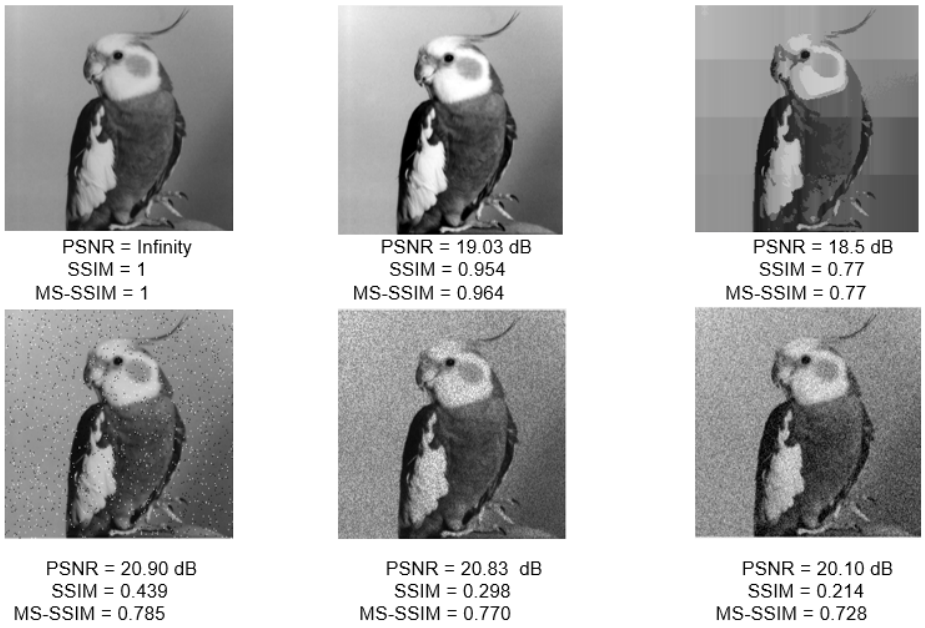


Figure 4.1: MS-SSIM, as a quality metric, is closer to subjective evaluation than PSNR and SSIM. From left-to-right starting across the top row, these images are 1. the original version [3] 2.contrast-stretched 3.mode 8x8 approximated 4.salt and pepper noise 5.Speckle noise 6. Gaussian noise

Autokinetic images are still images, that appear differently when viewed from different angles or distances. To quantify the quality of the autokinetic image evaluated in Figure 4.2, MS-SSIM provides a metric close to subjective evaluation than SSIM, since MS-SSIM is obtained by evaluating and weighting SSIM at different resolutions. Although the second, third and fourth images appear the same, they have dissimilar SSIM values, but have similar MS-SSIM values.

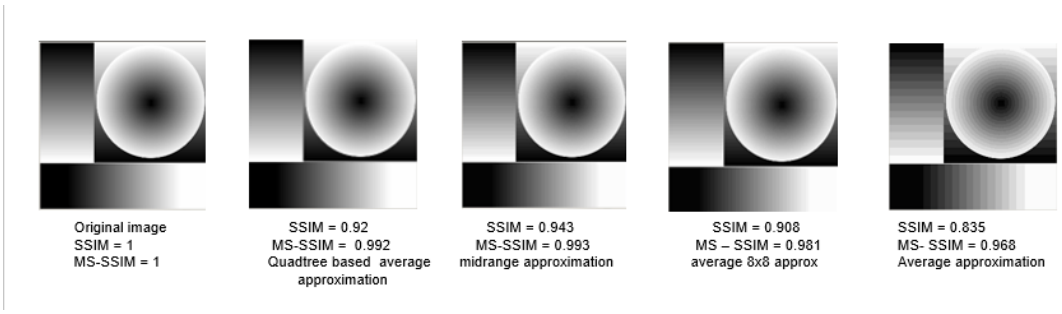


Figure 4.2: MS-SSIM is close to subjective evaluation than SSIM for autokinetic images

4.3.2 Compression

The motivation behind data compression is to enable efficient data transfer through a transmission link. The amount of data that can be transferred per unit time is restricted by the capacity of the transmission link, which is referred to as its bandwidth. The size of the compressed file is a direct indication of the reduction in bandwidth usage of the transmission link.

Compression algorithms demand more CPU processing power for lower bandwidth usage. The compressed data can now be sent faster on the transmission link, at the cost of increased processing at the sender and receiver for compressing and decompressing the data, respectively.

The evaluation of the performance of compression algorithms is based on the following two aspects:

- The size of the compressed file: The compression ratio is a metric that is used to evaluate the size of the compressed file with reference to the original uncompressed file.

$$CompressionRatio = \frac{Size\ of\ the\ uncompressed\ file}{Size\ of\ the\ compressed\ file}$$

  For Lossy compression, the compression ratio is defined at a particular value of the output quality.

- The time taken to approximate and compress the file at the sender and decompress the file at the receiver: The time command in Linux is used to measure the time

taken for approximation, compression, and decompression, expressed as an average over ten runs.

The choice of an algorithm for compression is based on the priority between the resulting file size and the speed of approximation and compression, or decompression, as dictated by the applications.A third factor – the memory requirement of the compression algorithm – also influences the choice.

Data Compression finds applications in storage and transmission. While compressing data for storage, the typical priority ladder from the top to the bottom is the compression ratio, time to decompress, and the time to approximate and compress. While generating data for storage, for example, backing up data on a hard drive, the primary concern is the size of the compressed file to meet the available storage capacity. This application involves a one-time compression and a many time decompression. Therefore, a longer approximation or compression time wouldn't affect as much as a longer decompression time would. In case of using lossy compression algorithms for storage, for example, to write a movie onto a disk, the goal is to work with a pre-specified fidelity and maximize the achievable compression.

Data compression for transmission is dictated by the bandwidth limitation of the transmission link. The goal is to meet constraints on the transmitted data rate, providing the highest possible fidelity. Video conferencing, mobile, and radio communications are examples of applications limited by the transmission rate of communication links.

Chapter 5

Results and Analysis

This chapter analyzes the results obtained from the lossless compression of the approximated test images. The proposed approximation techniques are presented in Chapter 3, the characteristics of the test images are discussed in Section 4.1 and the lossless compression algorithms considered for evaluation are presented in Section 4.2.

5.1 The tradeoff between compression ratio and output quality

The tradeoff space between MS-SSIM and the compression ratio is discussed in this section for each continuous-tone image in the benchmark. The classification of the images into the following categories is made based on the observations from Table 4.1.

- Autokinetic images
- Still images dominated by smooth textures
- Still images dominated by complex textures

From the comparison of the compression ratios provided by the different lossless compression algorithms in Table 4.3, it is observed that the CMIX algorithm, from the family of PAQ algorithms, gives the best compression ratio for all the images. Hence the analysis of the trade off space between MS-SSIM, entropy, and Compression Ratio is confined to the results of the CMIX algorithm.

To quantify the reduction in quality introduced by approximation, we compare quality against the original un-approximated image, which has a perfect MS-SSIM score of 1. To quantify the improvement in compression achieved by approximation, we compare the compression ratio obtained against that of the lossless compression of the original images using the CMIX algorithm, shown in Table 4.3.

Comparison of the impact of approximation on the compression ratio and quality of the approximated image is also performed against a simple sampling of the image as well as the widely-used lossy image compression technique – JPEG[43].

- Sampling of the image at different distances is implemented by dividing the image into groups of pixels, the number of pixels in each group being equal to the distance. From each of the groups, the first pixel is selected. Hence, the original file is compressed by a factor equal to the distance of sampling. The approximation techniques are expected to achieve better compression / quality tradeoffs than crude sampling of the image, they consider the characteristics of the images and the values of the pixels, unlike sampling which chooses representative values based on their positions.

- JPEG is a lossy image compression technique. The first step in the compression process, as shown in 5.1, is the conversion of the pixel information from the original image into frequency domain components by Discrete Cosine Transformation (DCT). A DCT coefficient with a high value represents a low frequency component, whereas a low valued DCT coefficient refers to a high frequency component. Low frequency components in an image are the smooth sections, the high frequency components refer to details such as the edges. The human visual system is sensitive to distortions in the low frequency regions, unlike those in the high frequency regions. The losses introduced in JPEG are in the quantization phase, which approximates the DCT coefficients corresponding to high frequency components to 0 and preserves the low frequency information. The extent to which the high frequency components are approximated can be adjusted by specifying the required output quality in JPEG.

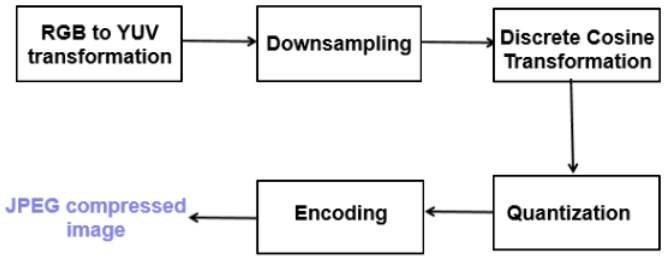


Figure 5.1: JPEG approximates the DCT coefficients corresponding to high frequency components in an image, due to the resulting imperceptibility of distortions.

5.1.1 Autokinetic Images

Autokinetic images, though synthetically generated, can be classified as continuous-tone images due to their qualities of blur and noise. Autokinetic images appear differently when viewed from different angles or distances. Hence, the introduction of approximations to these images could take advantage of the inability of the user to identify distortions in the image, based on the viewing distance.

Image-1: slope.tif

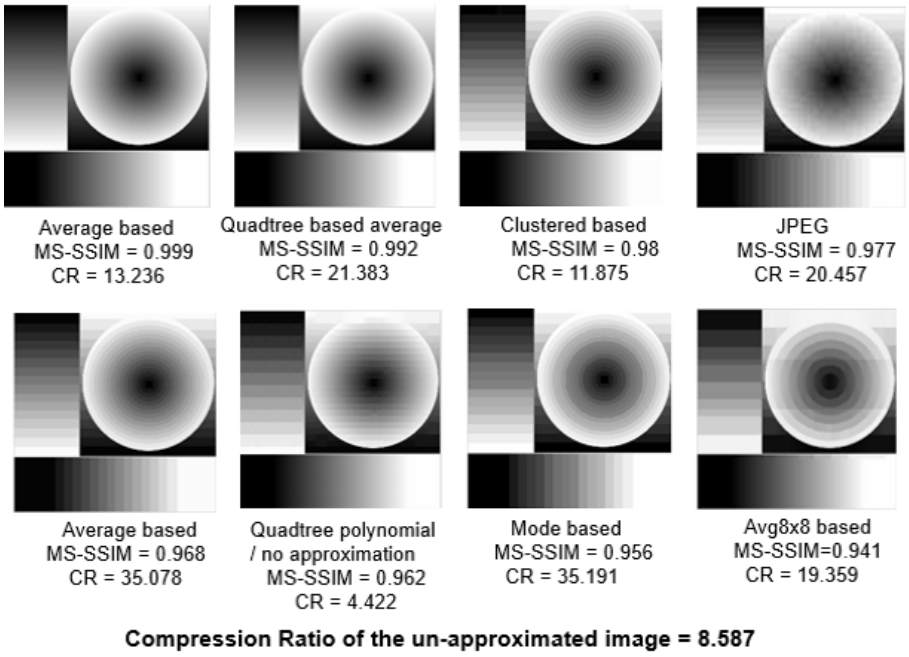
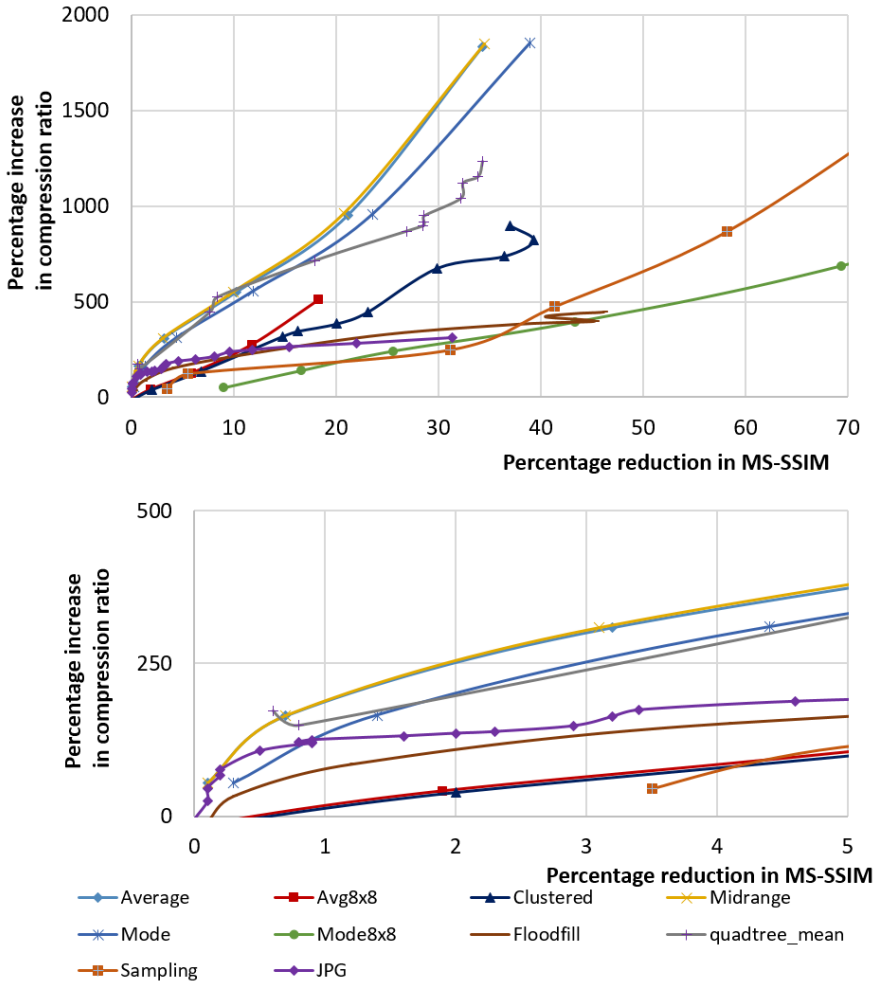


Figure 5.2: Data points from different approximation schemes with their MS-SSIM and CR for slope.tif, showing the output quality and compression rate.

The plot showing the percentage reduction in MS-SSIM with the percentage improvement in compression ratio for the different approximation techniques, sampling, and JPEG, is shown in Figure 5.3. The best compression ratio is provided by CMIX compression of mean- and midrange-based approximations through most of the range of MS-SSIM. This is closely followed by mode- and quadtree-based average approximation.

The image has regions of white, gray and black pixels, which are varying in gradients along different directions. The presence of regions with uniform pixel values helps the quadtree-based average technique. The clustering-based approximation performs well because it captures the uniformity in pixels along the horizontal scan-line and improves the compression ratio. Mode8x8 performs worse than Average8x8, since it loses quality at a faster rate.

For this image, the increase in JPEG's compression ratio is not significant even with a lot of quality lost. For example, in Figure 5.3(a), the improvement in compression ratio corresponding to a reduction in MS-SSIM from 10% to 30% is 1.3x. The analysis of JPEG's performance is discussed in the section 5.1.1, concluding auto-kinetic images.



]

Figure 5.3: The improvement in JPEG's compression ratio is not significant even with a lot of quality lost. Top: Plot of improvement in compression ratio vs reduction in MS-SSIM for different approximation schemes compressed with CMIX, JPEG and Sampling for slope.tif. Bottom: Zoomed plot showing up to 5% reduction in MS-SSIM.

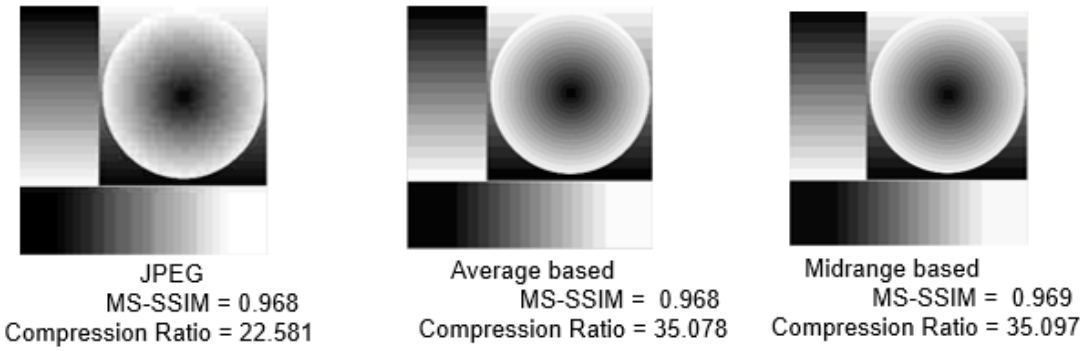


Figure 5.4: Comparisons between JPEG and approximation-based CMIX compression at the MS-SSIM value of 0.96 shows that average- and midrange-based approximate compression achieve significantly higher compression ratios for the same quality rating.

Image-2: cosinepattern.tif

To ensure that the results from the autokinetic image slope.tif are not image-specific, the image – cosine test pattern [44] is also analyzed. A few data points of interest to understand the quality rating of MS-SSIM are presented in Figure 5.5.

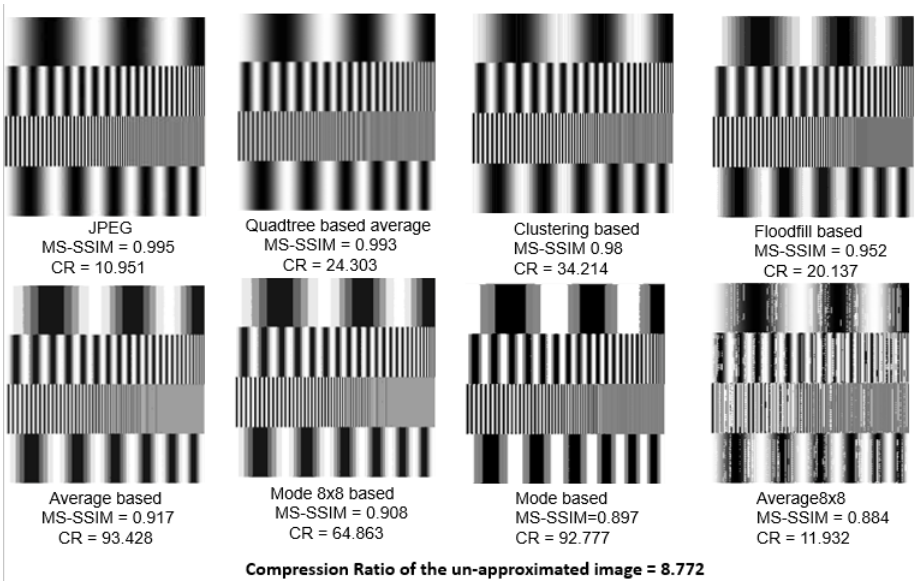


Figure 5.5: Data points from different approximation schemes with their MS-SSIM and CR for cosinepattern.tif

The plot showing the percentage reduction in MS-SSIM with the percentage improvement in compression ratio for the different approximation techniques – sampling and JPEG, is shown in Figure 5.6. Similar to the results obtained from the image slope.tif, the best compression ratio is provided by average-based and midrange-based approximations, closely followed by mode-based approximation.

This image has bands of white and black, with varying widths in different rows. Quadtree-based average approximation does not perform well here compared to slope.tif, because of the presence of fine details in the image, which minimizes the number of large regions with uniform pixels. Clustering and floodfill-based approximation techniques begin to saturate, since the regions with uniform pixels are limited in size and constrained by the presence of bands in the image. This image has regions dominated by black and white pixels and the gradient from black to white contains gray pixels. Since average8x8 weights all the pixels within each 8x8 block, the approximated image has equal contributions from white, gray and black pixels, even though our perception is mainly affected by black and white pixels. This is the reason why average8x8 approximation sacrifices a lot of quality to attain higher compression. Since mode8x8 does not weight

all the pixels in a region, the resulting image is a closer approximation to the original image.

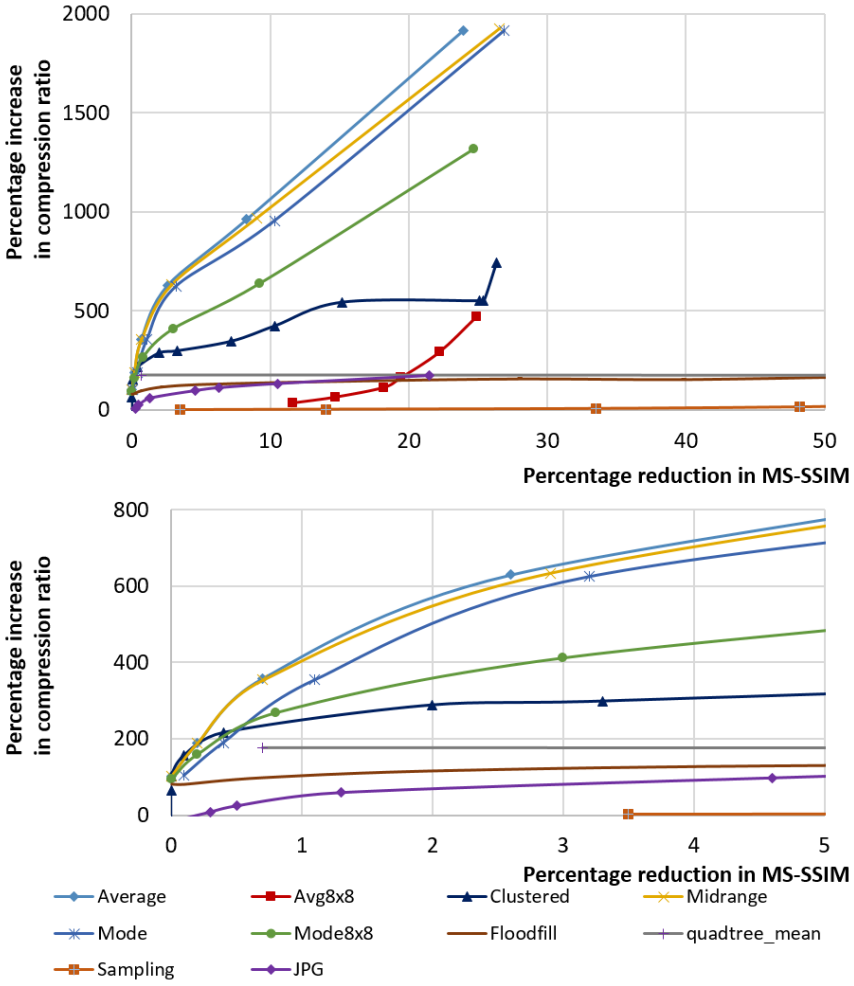


Figure 5.6: Average-based approximation performs better than JPEG due to the dominance of low-frequency components in the image Top: Plot of improvement in compression ratio vs reduction in MS-SSIM for different approximation schemes compressed with CMIX, JPEG and Sampling for cosinepattern.tif. Bottom: Zoomed plot showing up to 5% reduction in MS-SSIM.

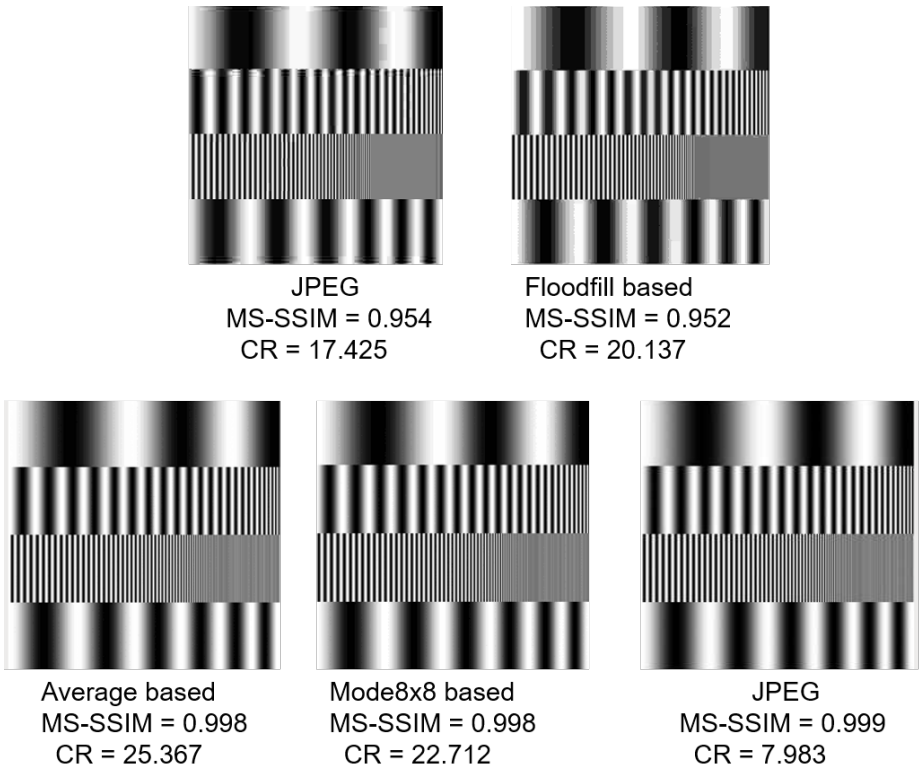


Figure 5.7: Comparisons between different lossy compression techniques at the MS-SSIM value of 0.95 and 0.99 for cosinepattern.tif demonstrates a significant benefit for approximation-based compression over state-of-the-art lossy compression (JPEG).

From Figure 5.7, it is observed that at the MS-SSIM value of 0.95, floodfill approximation provides 1.17x better compression than JPEG. At a MS-SSIM value of 0.99, averagebased approximation provides about 1.1x and 3x better compression than mode 8x8-based and JPEG compression techniques, respectively.

Conclusion For autokinetic images, average- and midrange-based approximation techniques provide superior performance to JPEG throughout the range of MS-SSIM. JPEG does not provide good compression for autokinetic images. JPEG introduces losses to the original image by approximating the DCT coefficients corresponding to high frequency components, or equivalently, approximating the DCT coefficients which have values close to 0. By approximating coefficients close to 0, there is an improvement in compression since the repetition of 0 coefficients increases.

The graph showing the distribution of DCT coefficients for the images slope.tif and cosinepattern.tif is shown in Figure 5.17 (a) (b). For autokinetic images, there are many coefficients already at 0 and very few coefficients close to 0. Hence, the opportunity for JPEG to provide improved compression, without introducing distortions in the image, is reduced. This causes the JPEG tradeoff curve to saturate with greater sacrifice in output quality, as can be observed from Figures 5.3 and 5.6. The quantization phase in JPEG starts to affect the low-frequency regions with further reduction in quality constraints, which produces visible distortions in the output. Hence, it is not possible for JPEG to achieve improvements in compression, without sacrificing a lot of quality.

While the existing still image compression standard JPEG does not achieve good improvements in compression with loss of quality, the proposed average-based and midrange-based approximation techniques provide drastic improvements in compression ratios with every unit reduction in output quality.

5.1.2 Still images dominated by smooth textures

The still images bird.tif, lena.tif, and camera.tif are composed of small and large smooth textures. The minor presence of coarse textured regions provide a camouflage to the approximations introduced to the image – the feathers and body of the bird in bird.tif, the hat and background in lena.tif, and the ground in camera.tif. Since the images are dominated by smooth sections, quadtree-based average representation can be expected to provide good compression.

With the increase in the presence of high frequency components in the image, JPEG can be expected to perform well. The images arranged in increasing order of presence of high frequency components are bird.tif, camera.tif and lena.tif, deduced from the presence of edges. Edges denote sharp transitions of the intensity values in an image.

Image-1: bird.tif

The data points of interest to understand quality and compression tradeoffs are presented in Figure 5.8.

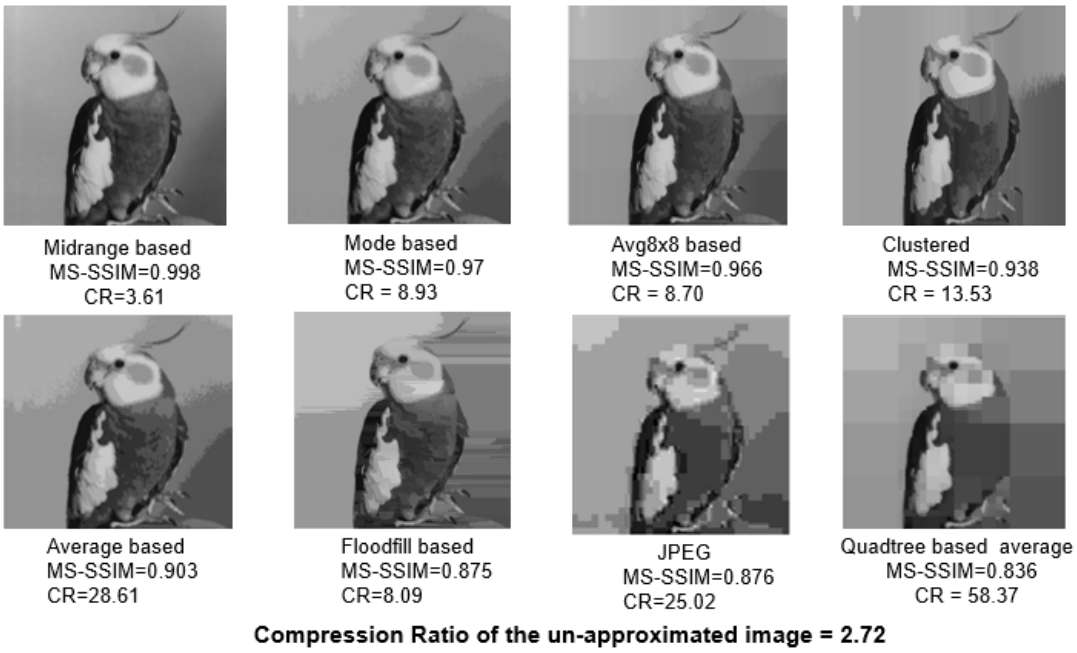


Figure 5.8: Data points from different approximation schemes for bird.tif

The comparison among the different approximation techniques followed by CMIX compression, Sampling, and the JPEG compression standard for the image bird.tif is presented in Figure 5.9.

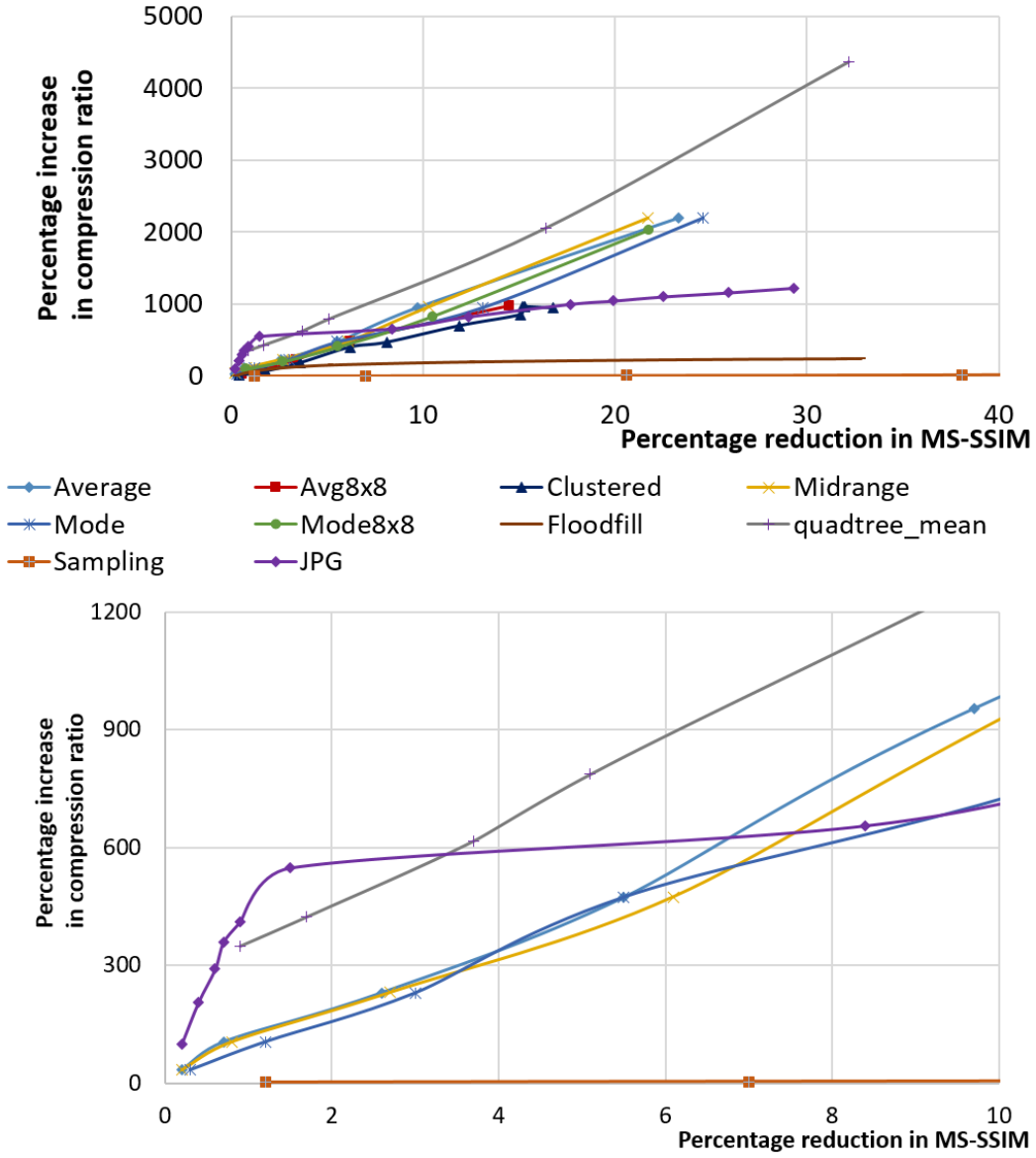


Figure 5.9: The high-frequency components in the image allow JPEG to introduce imperceptible losses and achieve good compression upto 1.5% reduction in MS-SSIM, beyond which the distortions start to affect our perception. Top: Plot of reduction in MS-SSIM vs improvement in compression Ratio for different approximation schemes compressed with CMIX, JPEG and Sampling for bird.tif Bottom: Showing region of interest, up to 10% reduction in MS-SSIM

The improvements in JPEG's compression ratio for every unit quality lost is steep for percentage reduction in MS-SSIM of up to 1.5%, as shown in Figure 5.9. From the distribution of DCT coefficients for the image bird.tif in Figure 5.17(c), the number of coefficients around 0 is more than that in autokinetic images (Figure 5.17(a)(b)), which implies that JPEG can provide improvements in compression by introducing less perceptible losses, working within a quality constraint. Beyond this region, JPEG must sacrifice a lot of quality to achieve better compression. For example, from Figure 5.9, the improvement in compression obtained with the reduction in MS-SSIM from 1.5% to 8.4% is only 1.19x.

The image bird.tif has large, contiguous regions with uniform pixel values, as in the background of the image. This helps quadtree-based representations since the division of the images into blocks of uniform pixels will result in large blocks being replaced with the dimension and the mean value of the block, providing benefits in compression. The quadtree-based average approximation technique followed by CMIX compression starts to perform better than JPEG for MS-SSIM values less than about 0.97 (more than 3% reduction in quality).

The average-based and midrange-based approximation methods perform better than JPEG for MS-SSIM values below 0.92 (more than 8% reduction in quality).

The compression gains achieved by clustering-based, floodfill-based and average 8x8 approximation schemes cannot be expected to match average-based and midrange-based approximation schemes, since these techniques generate many unique values of pixels to achieve higher accuracy.

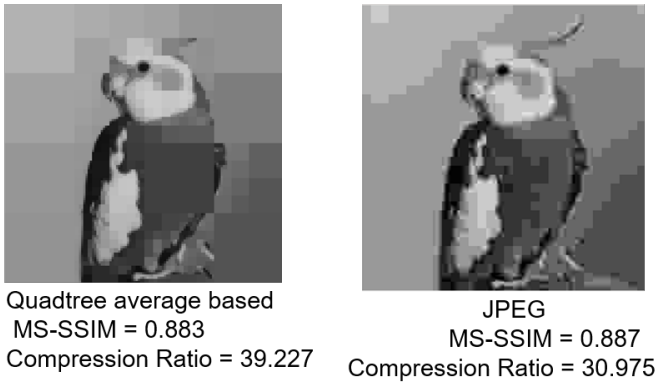


Figure 5.10: Comparisons between JPEG and approximation-based CMIX compression at the MS-SSIM value of 0.88, for bird.tif shows that quadtree-based average approximation achieves 1.25x higher compression than JPEG.

Image-2: lena.tif

Data points of interest to understand the quality and compression tradeoffs for lena.tif are presented in Figure 5.11.

The comparison among the different approximation techniques followed by CMIX compression, Sampling, and JPEG compression for the image lena.tif is represented in Figure 5.11.

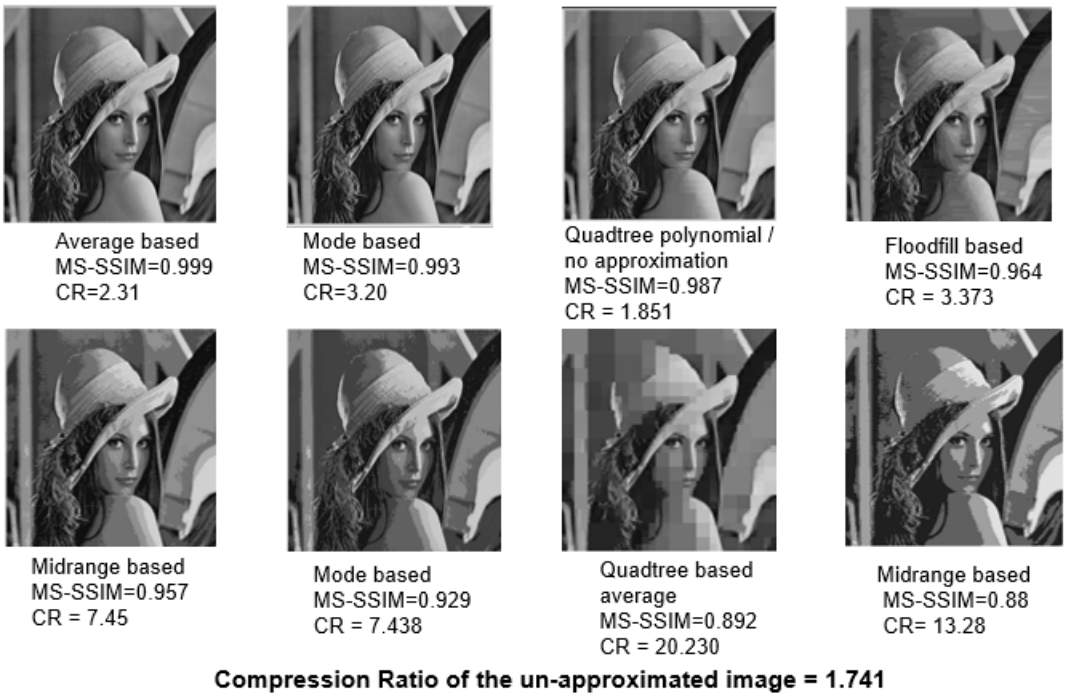


Figure 5.11: Data points from different approximation schemes with their MS-SSIM and CR for lena.tif

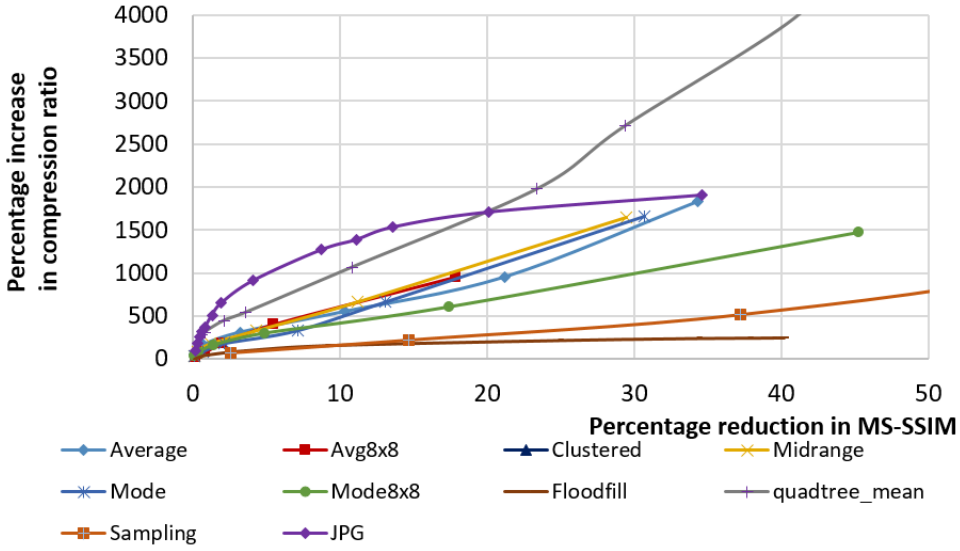


Figure 5.12: Presence of sharp variations in the intensity of the image enables JPEG to provide good compression upto 20% reduction in MS-SSIM for lena.tif.

Figure 5.12 shows a plot of MS-SSIM vs Compression Ratio for different approximation schemes compressed with CMIX, JPEG, and Sampling for lena.tif. It can be observed that crude sampling of the image provides higher compression than using floodfill-based approximation beyond a 15% reduction in output quality.

Floodfill-based approximation does not perform well since it aims for better accuracy, providing a lot of unique pixel values, which do not help improve compression. The quadtree-based average approximation starts to perform better than JPEG for MS-SSIM values less than about 0.80 (more than 20% degradation in quality). Unlike the large, smooth textured regions in bird.tif, the image lena.tif has smaller smooth textured regions, negatively impacting the performance of quadtree based average approximation, pushing the cross-over point between JPEG and quadtree-based approximation to 20% reduction in MS-SSIM.

Figure 5.12 shows that JPEG's compression ratio improves with loss of quality. This is due to the presence of high-frequency components in the image, as can be inferred from the distribution of DCT coefficients from Figure 5.17(e). There is a potential to achieve higher compression due to the increased presence of DCT coefficients close to 0, compared to the autokinetic images and the non-autokinetic images bird.tif and camera.tif. This is reflected in the dominance of JPEG for up to 20% reduction in MS-SSIM, compared to only 1.5% in bird.tif.

Another observation that can be made from Figure 5.12 is that the performance of average8x8 approximation meets that of average and mode-based approximations but does not extend beyond a reduction in MS-SSIM of 18%. Since average8x8 approximation weights all the pixels within each 8x8 block to obtain a representative value, it does not lose as much quality as average-, mode- or midrange-based approximations. The inability of 8x8 approximation to reach higher compression is due to the increase in the number of unique values that increase the number of codes to be assigned by the encoder.

Figure 5.13 shows the loss of significant detail from the image at the MS-SSIM value of around 0.78. JPEG provides the best compression ratio throughout the range of MS-SSIM where there is no significant quality degradation in the image, up to a quality reduction of 20%. If the application can tolerate lower qualities than MS-SSIM of 0.80, quadtree-based average approximation is an attractive option, as it achieves drastic

gains in compression ratio per unit quality lost. Below the reduction in MS-SSIM of about 20%, the compression ratio of JPEG does not see much improvement with further losses in quality, as it begins to approximate the DCT coefficients corresponding to low-frequency regions in the image.

Figure 5.13: Comparisons between JPEG and quadtree-based average approximation followed by CMIX compression for MS-SSIM around 0.78 shows that quadtree-based average appoximation gives 1.15x better compression than JPEG, but with the loss of significant detail.

Image-3: camera.tif

Figure 5.14 presents several sample points in the quality / compression tradeoff space for camera.tif.

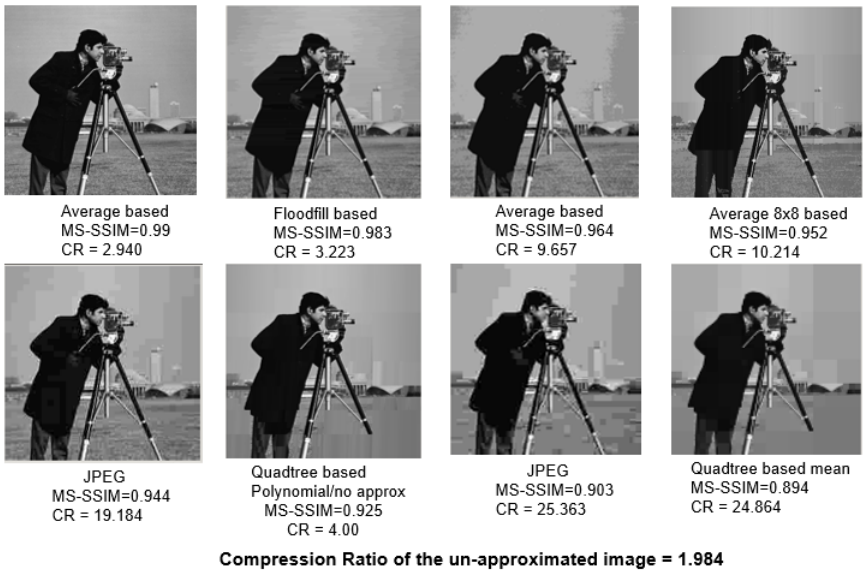


Figure 5.14: Sample data points from different approximation schemes with their MS-SSIM and CR for camera.tif

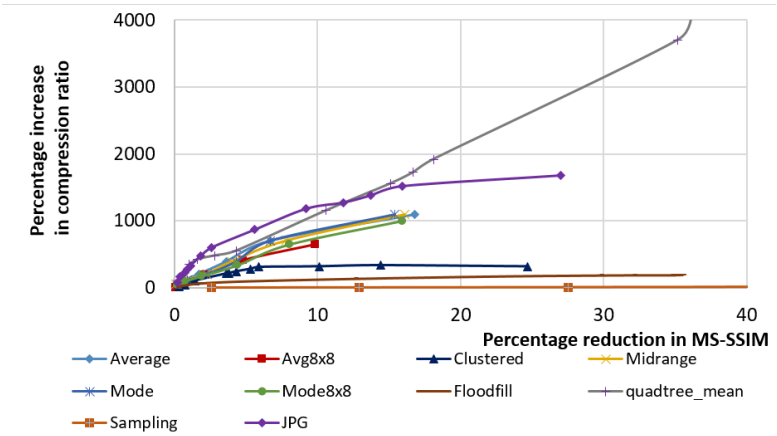


Figure 5.15: The cross-over point between JPEG and quadtree-based average compression shifts to the left compared to the observation from Figure 5.12, implying fewer high-frequency components.

Figure 5.15 shows a plot of the trade-off between MS-SSIM and Compression Ratio for different approximation schemes compressed with CMIX; JPEG, and Sampling for

camera.tif. It is observed that JPEG compression performs the best through the range of MS-SSIM values from 1 to 0.88. This is due to the presence of high-frequency components in the image, as can be inferred from the distribution of DCT coefficients from Figure 5.17 (d), which is greater than that in bird.tif and lesser than lena.tif. This is reflected in the cross-over point between JPEG and quadtree-based average approximation which is now at 12%, compared to 1.5% in bird.tif and 20% in lena.tif. Quadtree-based average approximation starts to perform better than JPEG for MS-SSIM values less than about 0.88 (more than 12% degradation in quality). While bird.tif has large smooth textured regions and lena.tif has smaller smooth textured regions, the characteristics of this image assume a middle ground between the two.

The image camera.tif is comprised of a fair balance between coarse and smooth textured regions compared to lena.tif and bird.tif. The coarse textured regions can hide distortions introduced by the approximations, for example, the grass in camera.tif. Hence, the minimum MS-SSIM value that can be reached by techniques like average-, midrange-, and mode-based approximations is only between 0.90 and 0.80 (see Figure 5.15). In contrast to camera.tif, the presence of coarse textures in images lena.tif and bird.tif is localized, hence the reduction in quality introduced by average-, midrange-, and mode-based techniques is about 30% and 20%, respectively, even when the distance of approximation is at its maximum.

Clustering-based approximation begins to saturate with reduction in quality because with greater distance, the clusters that are already formed have minimal addition/deletion of elements. With changes in the number of elements in a cluster, the mean value of the cluster changes – this affects the output quality. But there is not a significant change in the number of clusters which are formed, resulting in the same number of codes to be assigned and no improvement in the compression ratio.

Figure 5.16: Comparisons between JPEG and quadtree-based average approximation followed by CMIX compression at the MS-SSIM value around 0.84, for camera.tif

Figure 5.16 compares the distortions introduced by quadtree-based average approximation and JPEG at the MS-SSIM value of 0.84. At this quality level, both techniques achieve a similar compression ratio. The approximations introduced by quadtree-based average cause a loss of the legs of the tripod of the camera, and the boundaries of the quadtree regions are visible in certain areas. JPEG approximation introduces noise at the edges of the man and the tripod and creates a hazy background.

Conclusion For still images dominated by smooth textures, the choice between approximation schemes should be made based on the proportion of high-frequency components in the image and the required output quality from the application. If only minimal output quality degradation is permissible and the image has high frequency components, JPEG is the best choice. If higher compression ratio is more important than output quality and the image has high frequency components, quadtree-based approximation schemes perform better than JPEG.

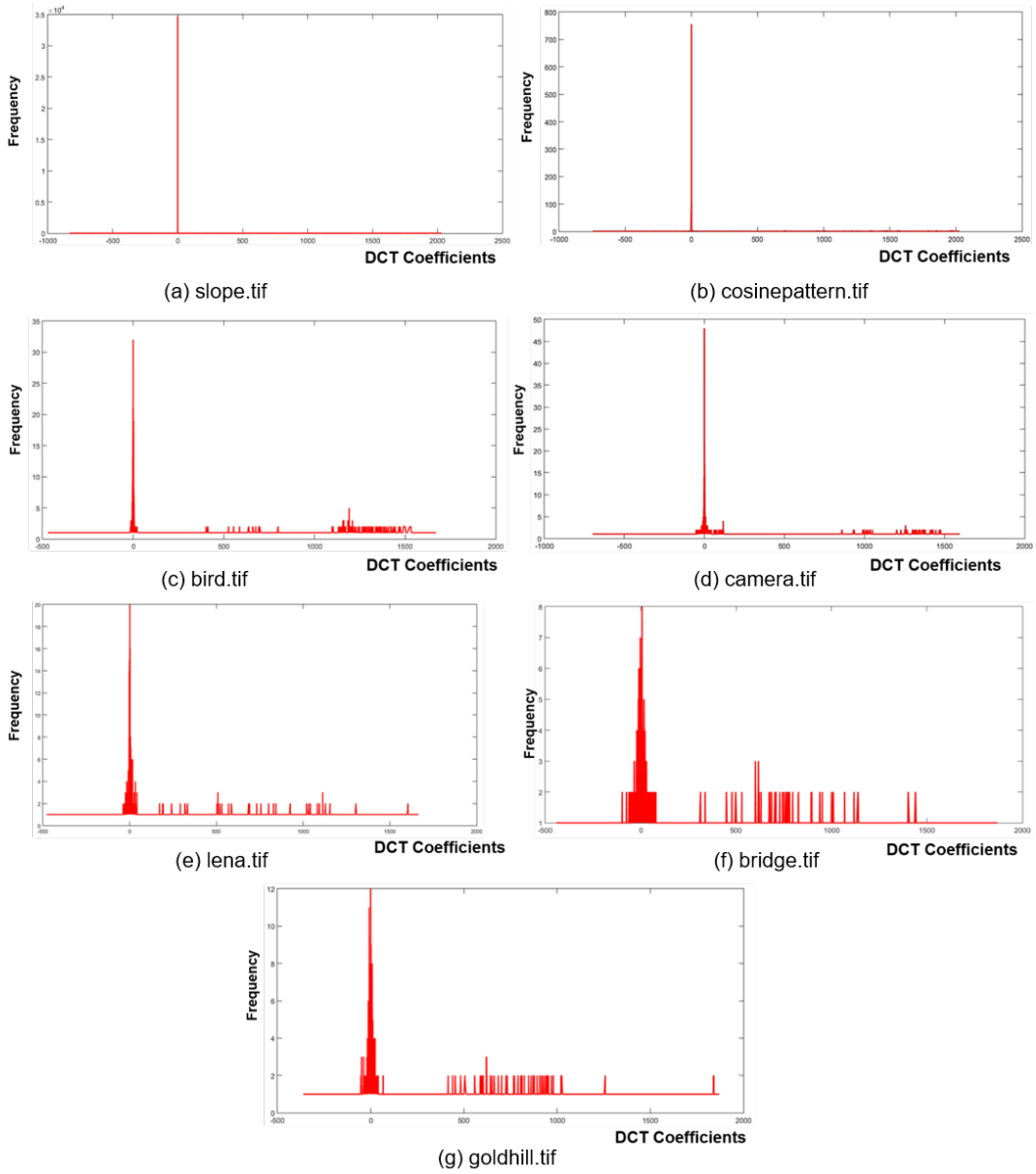


Figure 5.17: DCT coefficients with values close to 0 correspond to high-frequency components in the image. Distribution of DCT coefficients for the following images. From top left: (a) slope.tif (b) cosinepattern.tif (c) bird.tif (d) camera.tif (e) lena.tif (f) bridge.tif (g) goldhill.tif

5.1.3 Still images dominated by complex textures

The still images bridge.tif and goldhill.tif are composed of complex textures. The presence of complex textured regions provide a camouflage to the approximations introduced to the image. With such complex textures, quadtree-based approximation may not perform well at high quality requirements since it may not be feasible to identify large regions with contiguous regions, providing benefits in compression. With relaxation in quality requirements, quadtree-based average approximation may begin to perform well. Since these images have sharp changes in intensity indicating the presence of high frequency components, JPEG can be expected to provide good compression.

Image-1: bridge.tif

Figure 5.18 presents several sample points in the quality / compression tradeoff space for bridge.tif.

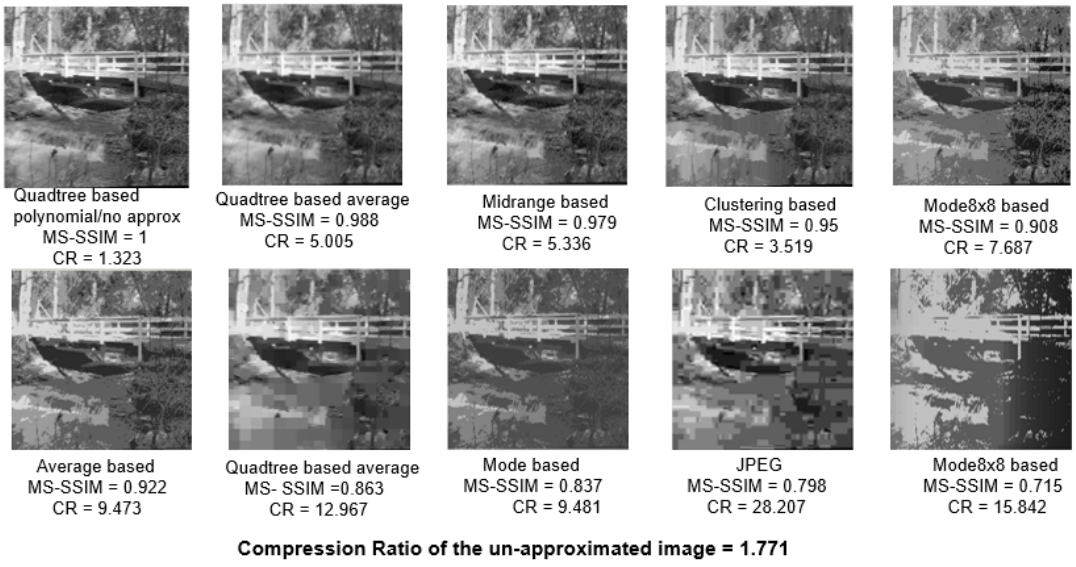


Figure 5.18: Data points from different approximation schemes with their MS-SSIM and CR for bridge.tif

JPEG provides the best compression ratio up to the MS-SSIM value of 0.68 beyond

which, quadtree-based average approximation starts to perform better as shown in Figure 5.19. JPEG's dominance for up to 32% reduction in MS-SSIM is due to the presence of high-frequency components in the image. On transforming the pixel information to DCT coefficients, these manifest as values close to 0, that can be approximated to 0 for higher compression, as can be seen from Figure 5.17 (f).

Figure 5.19 shows the trade-off space between MS-SSIM and compression ratio for different approximation schemes compressed with CMIX, JPEG, and Sampling for the image, bridge.tif. The dominance of complex textures in the image helps hide distortions effectively, allowing JPEG to provide higher compression with less perceptible losses. There is a significant drop in quality between successive approximation points, as we move towards the lower quality region. For example, from Figure 5.19, corresponding to the JPEG trendline, a quality setting of 3 gives an MS-SSIM value of 0.8, whereas the MS-SSIM value corresponding to the quality setting 2 is 0.35.

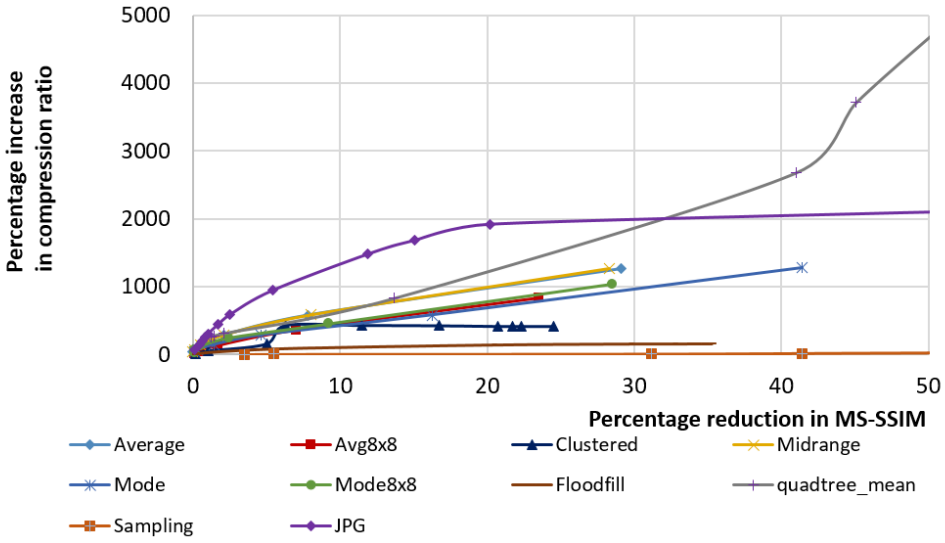


Figure 5.19: The presence of details in the image helps JPEG hide distortions effectively and achieve good compression without introducing perceptible losses.

The proposed approximation techniques like clustering and floodfill-based approximations do not perform well for bridge.tif because they are unable to identify large, contiguous regions with uniform pixel values, owing to the presence of details in the

image. Techniques like average, midrange and mode-based approximations work well for images which are dominated by low frequency components. These pixel modulation techniques which aim to increase the repetition of certain pixel values do not provide improvements in compression for images with complex textures without a significant reduction in MS-SSIM. They occupy the middle ground between JPEG and crude sampling of the image. The only exception to the above observation is the quadtree-based average approximation, which achieves drastic improvements in compression ratio, if the permissible loss of quality is below 32%. However, the image becomes unrecognizable when it approaches the reduction in quality of 30%, as can be observed from Figure 5.18.

Image-2: goldhill.tif

The image goldhill.tif also behaves similar to bridge.tif in response to the different approximation schemes. This image, as shown in Figure 5.18, has well defined boundaries between complex textured and small, smooth textured regions. A comparison of the MS-SSIM rating and CR for different points in the tradeoff space is shown in Figure 5.20.

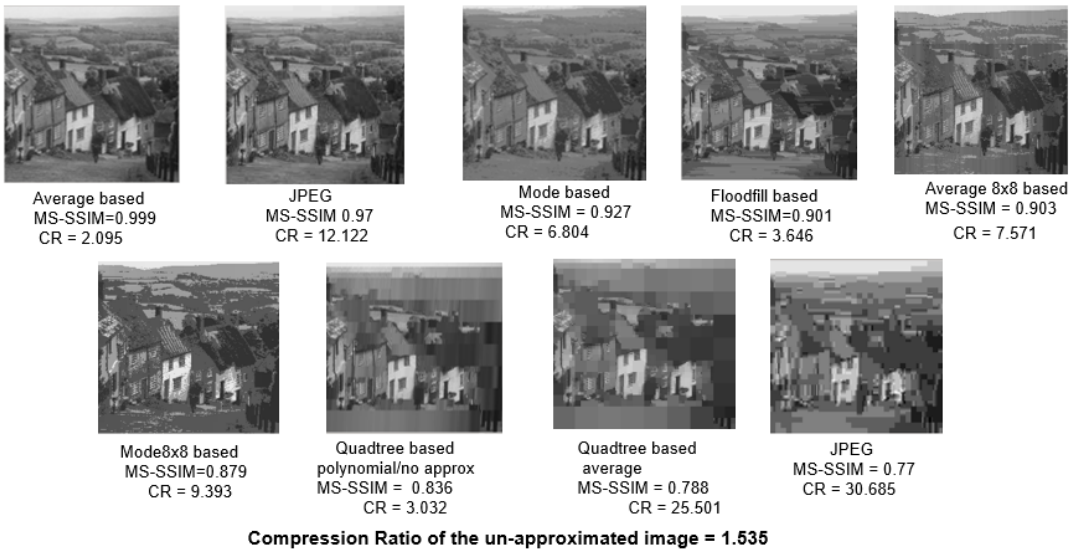


Figure 5.20: Data points from different approximation schemes with their MS-SSIM and CR for goldhill.tif

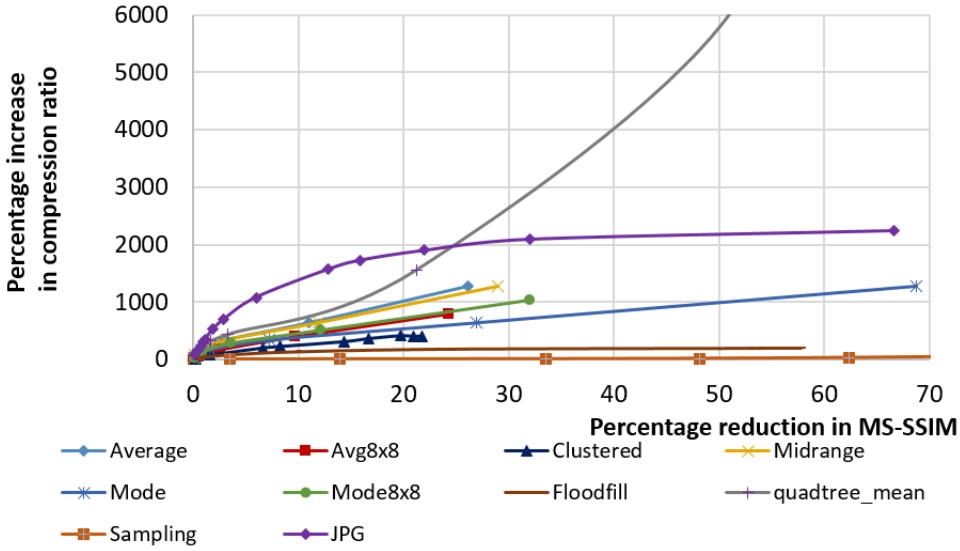


Figure 5.21: The cross-over point between JPEG and quadtree-based average compression shifts to the right compared to the observation from Figure 5.12, implying greater high-frequency components.

Figure 5.21 shows the trade-off space between MS-SSIM and compression ratio for different approximation schemes compressed with CMIX; JPEG, and Sampling for the image, goldhill.tif. It can be inferred that JPEG provides the best compression ratio for output quality reduction up to 25%. Applications that can tolerate a loss of quality beyond the MS-SSIM value of 0.75 can employ quadtree-based average approximation to achieve greater benefits in compression for every percentage of additional quality loss.

Quadtree-based average approximation meets the compression ratio of JPEG at a MS-SSIM value of about 0.75, whereas for bridge.tif, the cross-over point occurred at the MS-SSIM value of about 0.67. This indicates the increased presence of high-frequency components in bridge.tif as comapred to goldhill.tif, which JPEG can approximate and achieve higher compression with less perceptible losses, as can be observed from Figure 5.17 (g). With a relaxation in quality constraints, quadtree-based approximation is able to identify large blocks with pixel values that are within the variance threshold, providing better compression.

Similar to the observation made in bridge.tif, due to the presence of details in the

image, the pixel modulation based approximation techniques do not provide good compression for the image goldhill.tif.

Conclusion

For images that have complex textures in them with a lot of edges, JPEG gives the best compression ratio while maintaining a recognizable output. The increased presence of edges corresponds to an increase in the high-frequency components in an image, which can be quantized to produce higher compression without significantly visible distortions. If the application permits further loss in quality, quadtree-based average representation would present an attractive alternative to JPEG, since it provides drastic improvements in compression for every percentage of quality sacrificed. With relaxation in quality constraints, quadtree-based approximation identifies larger blocks of pixels that can be represented by just two numbers – the dimension of the block and the mean of the block, enabling greater benefits in compression.

The quadtree-based polynomial approximation technique does not provide a good compression ratio for any of the images from the benchmark. The coefficients of the polynomial to be transmitted are floating point values that occupy 4 bytes, along with the degree of the polynomial and the dimension of the block, which contribute 2 bytes. Consider the worst case of approximating an 8x8 block of pixels with a polynomial of degree 9; the uncompressed stream consists of 64 bytes. The compressed stream, when fit with a polynomial would require 40 bytes for the coefficients and 2 bytes for the degree of the polynomial and the dimension of the block, totaling 42 bytes. In the event of not being able to approximate due to the variance of the block exceeding the variance threshold, the uncompressed stream of 64 bytes has to be sent along with the dimension of the block and the number 64 (indicating that the stream is uncompressed), requiring 66 bytes. This approximation technique results in an 8x8 block of pixels being replaced by either 42 bytes or 66 bytes, depending on whether approximation is done or not. The compression that can be afforded is not competitive enough with the other proposed techniques.

5.2 Recommendation of approximation techniques based on image characteristics

5.2.1 Autokinetic images

For the autokinetic images shown in Figures 5.3 and 5.6, the best performance throughout the range of the MS-SSIM is obtained by average- and midrange-based approximation techniques followed by CMIX compression. By increasing the distance of approximation while using the proposed approximation techniques, we are able to obtain significant and continuous improvements in compression with every unit of quality lost. The absence of high-frequency components denies JPEG the opportunity to obtain higher compression, without introducing visible distortions to the image, as was discussed in Section 5.1.1.

5.2.2 Regular still images

To compress regular still images with high quality, JPEG offers the best compression ratio. When the application permits relaxation on the limits of output quality, quadtree-based average approximation followed by CMIX compression gives the best compression ratio.

The slope of the compression ratio vs quality reduction curve varies with the relative composition of edges and regions with uniform pixels in the image. When the image has sharp transitions in the intensity of the image indicating the presence of edges, the range of MS-SSIM where JPEG is the best approximation method increases. JPEG approximates the high-frequency components in the image, providing higher compression without affecting our perception significantly, hence providing the best means to compress regular images with high constraints on output quality. With relaxation in output quality, quadtree-based approximation is able to identify large regions with uniform pixel values, achieving better compression, but with the introduction of distortions. The conditions for dominance of JPEG and quadtree-based approximation are illustrated in Figure 5.22.

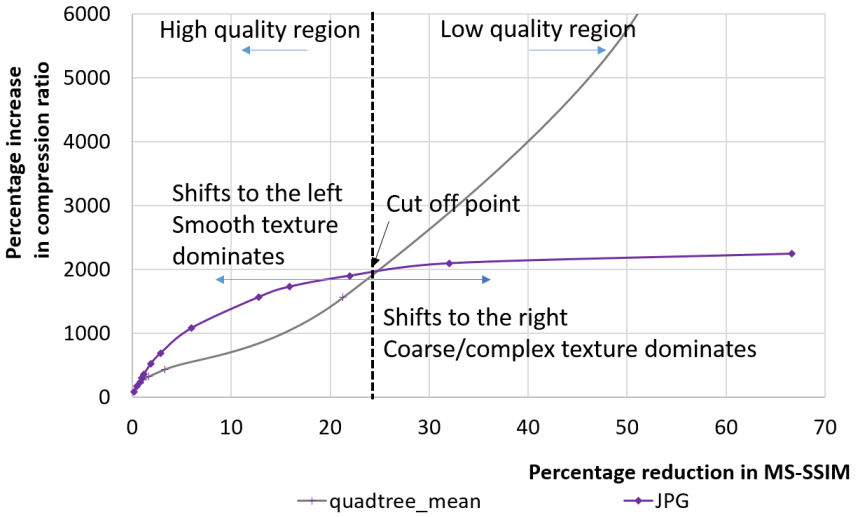


Figure 5.22: With the increase in the presence of high-frequency components in the input image, JPEG provides better compression with imperceptible losses and dominates quadtree-based approximation for a greater range of MS-SSIM.

5.3 The Relationship between Entropy, Compression Ratio, and Quality

The approximation techniques described in Chapter 3 – modulating the pixel values (Section 3.1) and quadtree-based average representation (Section 3.2) – aim to reduce the entropy of the image by constraining the pixel values to a smaller set of values. This section analyses the correlation between the reduction in the entropy of the approximated image, the MS-SSIM between the original and approximated image, and the gain in compression ratio of the CMIX compression algorithm afforded by the different approximation techniques.

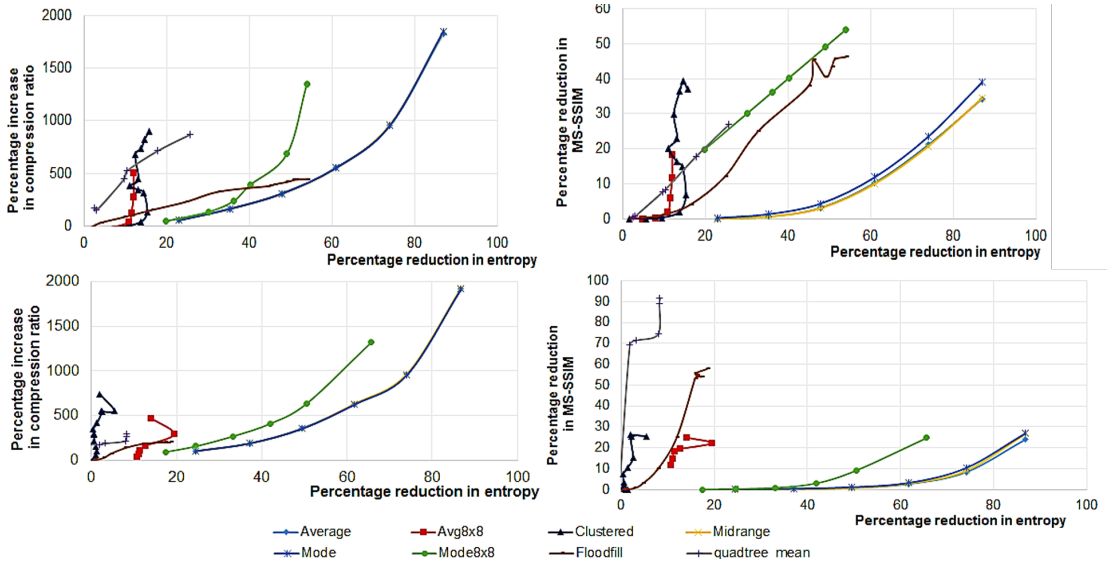


Figure 5.23: Plot showing the impact of reduction in entropy on the compression ratio and MS-SSIM for autokinetic images (a) slope.tif (b) cosinepattern.tif

The following inferences are made from the plots in Figures 5.23 5.24 5.25:

- Given an approximation technique, as the entropy of the approximated image decreases, an increase in compression ratio is observed. Entropy, a measure of information content in a message, can be reduced by increasing the frequency of symbols. The principle behind data compression by removal of redundancy is the assignment of short codes to frequent events and long codes to rare events. The approximation techniques increase the redundancy of data in the message, which in turn increases the number of short codewords in the compressed stream, implying more compression.

- Different approximation schemes affect the entropy differently. The observations that can be made on entropy reduction for a particular image are specific to an approximation technique. For instance, in Figure 5.24, mode-based compression, at the maximum distance of approximation, reduces the entropy of the image by about 85%, whereas the quadtree-based mean reduces the entropy only by about 45% but still achieves higher compression than mode-based approximation.

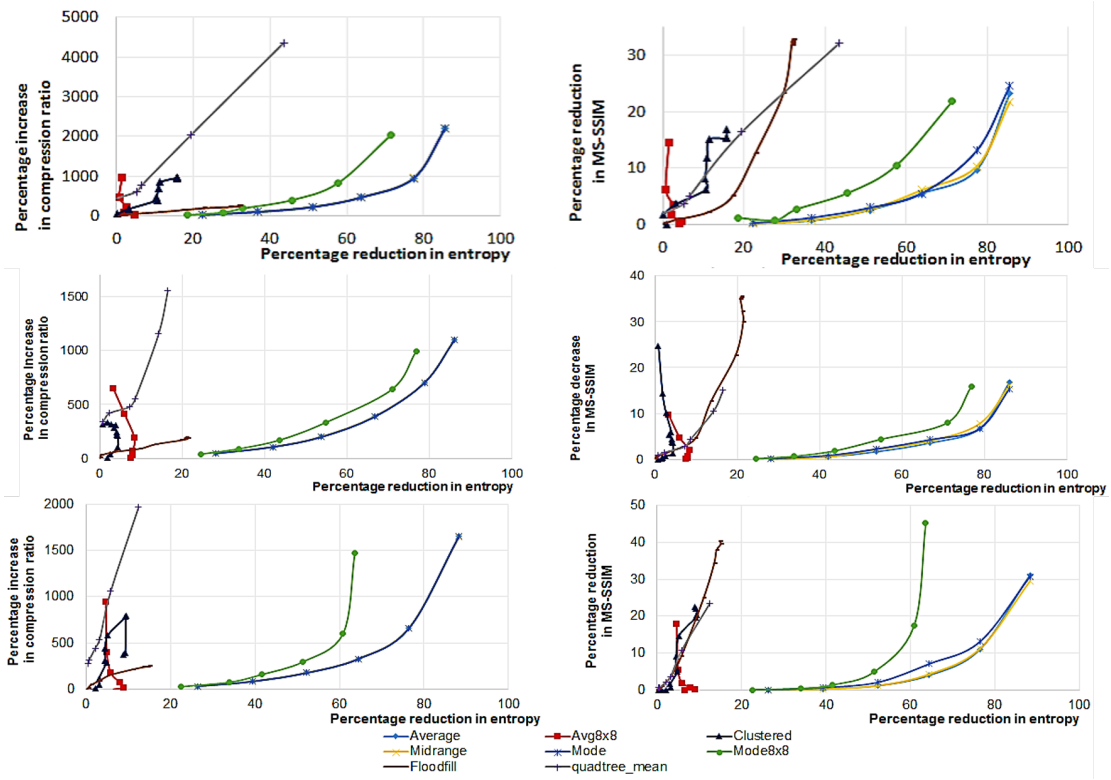


Figure 5.24: Plot showing the effect of entropy on the compression ratio and the MS-SSIM for smooth-texture dominated images (a) bird.tif (b) camera.tif (c) lena.tif

- For all the images from the benchmark set, the entropy of mode-, average-, and midrange-based approximated images are the same. As illustrated by the example in Section 3.1, these approximations result in different set of symbols, which all have the same frequencies of occurrences. Hence, the resulting entropy values and compression ratios from these methods would be the same. These methods differ in the set of the approximated pixels to which the pixels from the original image are mapped, resulting in variations in their output quality metrics.

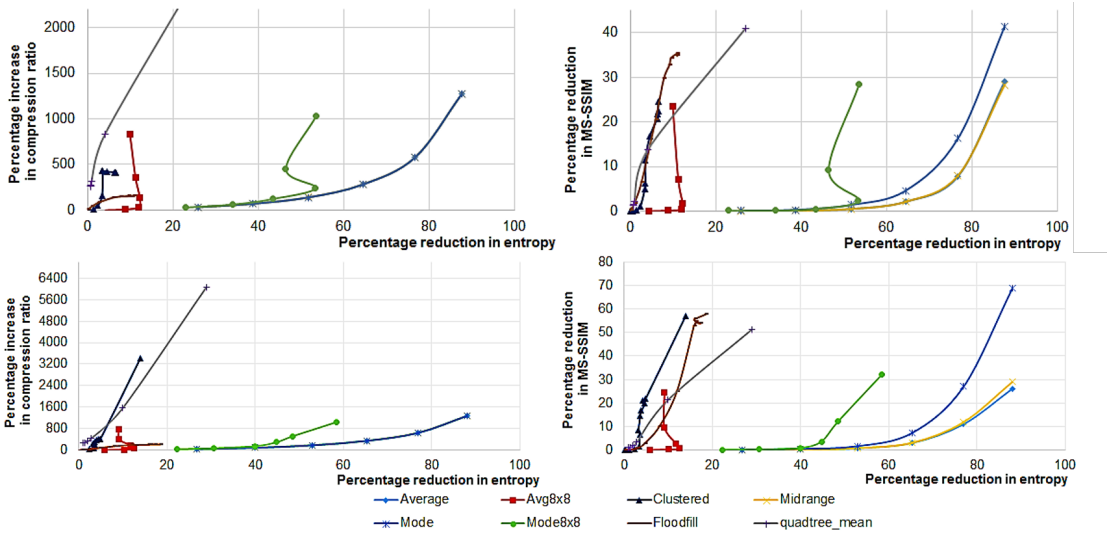


Figure 5.25: Plot showing the effect of entropy on the compression ratio and the MS-SSIM for complex-texture dominated images (a) bridge.tif (b) goldhill.tif

- The entropy calculation for images performed in this section is the first order entropy. First order entropy is as defined in Shannon's Information theory, explained in Section 2.1. The symbols emitted from a source are viewed as independent and the probabilities assumed for coding are for each symbol. The compression algorithm used in this thesis – CMIX – takes into account the relationship between the symbols and calculates conditional probabilities based on the context for encoding. To obtain the exact correlation between entropy and compression ratio, a higher order entropy needs to be calculated based on the context models considered by the CMIX algorithm (from the PAQ family of compression algorithms) to assign probabilities.

5.4 Time Analysis

This section analyses the performance of different approximation and compression algorithms based on the time taken to approximate, compress and decompress the images. The tests are performed as sequential processes on a node with a single Intel Xeon CPU

X5675, running at 3.07 GHz.

5.4.1 Time to approximate

The approximation schemes are implemented in MATLAB and the time taken is measured using the elapsed time from the stopwatch by running the tic and toc commands. The time measurement is averaged over ten runs of the program.

Table 5.1 shows the time taken to approximate for different pixel modulation schemes and the compression ratio obtained by compressing the approximate images using CMIX compression algorithm. The time taken for approximation is averaged over all the images from the benchmark set.

Approximation Scheme	Compression Ratio	Time to approximate(s)	Time to recover(s)*
Average	19.75	10.10	
Mode	19.80	15.90	
Average8x8	9.43	10.75	
Mode8x8	16.63	23.40	
Midrange	19.82	1.19	
Clustered	5.17	4.26	
Floodfill	6.18	5.98	
Quadtree-based average	22.84	1.08	0.527
Quadtree-based polynomial	5.16	271.68	1.54

Table 5.1: Average time taken to approximate the images in the benchmark

* Time to recover is the time taken by the receiver to reconstruct the pixels from the decompressed file. This is defined only for approximation techniques that represent pixels using quadtrees. The other approximation techniques implement modulation of pixel values and require no recovery.

The time taken to approximate by the proposed approximation schemes can be reduced by parallelization.

5.4.2 Time to compress and decompress

The CMIX compression algorithm achieves the highest compression ratio for all the images from the benchmark set. It also has the highest runtime, as it estimates the

probability of occurrence of a symbol by combining the results from different modeling schemes.

For applications like accessing a file from a local storage, a higher priority is placed on the time elapsed for processing compared to the compression that can be obtained. A better option would be to opt for other compression techniques that compress and decompress faster than CMIX, but provide a slightly lower compression ratio.

The time taken to compress and decompress are measured using the time command in linux, which is averaged over ten repeated runs. Table 5.2 shows the values of the time taken to compress and decompress against the corresponding compression ratios provided by the different lossless compression algorithms. The measurements of time and compression ratios are averaged over all the approximated images in the benchmark set.

Compression Algorithm	Switches	Compression Ratio	Time to compress(s)	Time to decompress(s)
7zip	–m0=LZMA2	7.916	0.035	0.034
7zip	–m0=PPMd	9.031	0.031	0.047
BZIP2	–best	9.474	0.025	0.019
CMIX		16.175	35.216	35.030
GZIP	–best	9.313	0.017	0.016
INFOZIP	–best	8.497	0.028	0.020
LHA		8.967	0.016	0.013
LZ4	–9	4.371	0.010	0.008
NANOZIP(LZHD)	–cd	9.175	0.025	0.042
LZIP	–best	10.963	0.039	0.019
LZOP	–9	4.762	0.011	0.073
NANOZIP(NZCM)	–cc	11.858	0.205	0.203
PAQ 8o6	–7	14.103	2.736	2.718
RAR	–m5	7.143	0.053	0.024
XZ		10.659	0.035	0.023
ZLIB		8.645	0.015	0.039
ZPAQ	–best	8.626	0.178	0.192
ZZIP	–mx	10.002	0.019	0.016

Table 5.2: Average time taken to compress and decompress the images in the benchmark set using lossless compression algorithms

Chapter 6

Conclusions

The ability of future RMS applications to permit inaccuracies in the output values can be exploited to provide application-aware benefits in compression ratio. Improved compression ratio can be translated into reduction in communication bandwidth utilization, storage costs, and data transmission latency on a communication link by partially transforming the latency of communication into increased processing time at the sender and the receiver.

As trends in future systems move towards parallel computing, there is an increase in the number of data transfers to be made. This indicates that it is possible to make the time to approximate, compress and decompress infinitesimally small by parallelism, yet, the crux of the problem remains unsolved - the data transfers being limited by the bandwidth of the communication link. The emergence and ubiquity of mobile computing, sensor networks, parallel computing, cloud computing, etc. have increased the stress on the communication bandwidth. This thesis explores the introduction of approximation techniques to enhance the performance of existing lossless compression techniques to overcome the restrictions imposed by the bandwidth of the channel.

By making the approximation techniques input-aware, it is possible to achieve improvements in compression. The motivation behind the development of different types of approximation techniques stems from the differing characteristics of images, e.g., presence of smooth and textured regions, variation in the perception of the image with viewing distances and resolutions, and the density of details in the image.

The performance of the proposed approximation techniques has been compared

with the widely used lossy compression technique, JPEG. JPEG does not provide good compression for images with scarce high-frequency components. The average-, mode-, midrange- and quadtree-based average approximation techniques provide significantly higher compression ratios than JPEG throughout the range of the MS-SSIM quality metric for the autokinetic images analyzed in subsection 5.1.1. They also achieve higher compression rates at lower quality ranges for images with smooth regions, which are dominated by low-frequency components, as was observed for bird.tif in subsection 5.1.2. The comparisons made below are with regard to JPEG.

- The improvement in compression ratio is about 7x at 5% tolerance in MS-SSIM and up to 10x at 23% reduction of MS-SSIM for autokinetic images, for average- and midrange-based approximation followed by CMIX compression.
- For images dominated by smooth regions, the improvement in compression is 0.6x to 3.5x throughout the range of the MS-SSIM, for quadtree-based average approximation followed by CMIX compression.
- From the high quality to the very low quality region for images dominated by coarse regions, the improvement in compression in comparison to JPEG is drastic, ranging from 0.75x to 1.5x, for quadtree-based average approximation followed by CMIX compression.

The impact of approximations on the output quality vary with the characteristics of the image. The introduced distortions show through easily in the smooth regions of an image, while the regions that are dense with details provide a camouflage. The requirements imposed by the application dictate the lower limit on the permissible degradation in output quality. For example, in medical applications, it is not possible to allow any losses in the output quality, while applications like streaming movies can tolerate comparatively greater degradation in quality.

The requirements from the application typically fall in one of the following two categories:

- Given the constraints on the transmission rate, the image must be compressed with the highest achievable fidelity.

- Given a pre-determined fidelity, the maximum achievable compression should be aimed for.

The improvement in compression ratio with the introduction of approximation brings a compromise on the time taken to approximate, compress, and decompress. The priority among the compression ratio, image quality, and processing times can be resolved by the choice of an appropriate approximation scheme and compression algorithm, considering the characteristics of the image. This thesis lays out a trade off space among the variables influencing approximate image compression.

6.1 Future Work

- Based on the different factors influencing image compression considered in this thesis, an adaptive scheme could be developed which would identify the characteristics of the input image and based on the constraints on the output quality, processing time, or the storage capacity specified by the user, the image could be compressed with the most appropriate approximation and compression schemes.

- This thesis can be extended from grayscale, continuous-tone images to approximate RGB color images. RGB color images are usually split into luminance, chrominance-red and chrominance-blue components (YCbCr conversion) to remove the correlations between red, green and blue components of a color image. The luminance component of an RGB color image is the grayscale image, for which the approximation techniques developed in this thesis can be applied. Typically, the chrominance components are sub-sampled for every luminance component as the human visual system has higher sensitivity to variations in brightness compared to colors. Similar work can be performed in other applications that can tolerate losses in output quality, like audio and video compression.

- Quadtree-based image representation was accomplished by recursive subdivision of the image space into quadrants based on the variance constraints of the blocks. By extending pixel blocks to rectangles, a better compression could be obtained.

- The use of Graphical Processing Units (GPUs) to accelerate approximation schemes

that can be parallelized will offer an improvement in time taken to perform approximation.

References

[1] Responsive web design: Why image optimization is crucial for a mobile friendly experience http://www.trilibis.com/files_/Trilibis_RWD_survey_APR_2014.pdf.

[2] RaymondW Yeung. Rate-Distortion theory. In *Information Theory and Network Coding*, Information Technology Transmission Processing and Storage, pages 183–210. Springer US, 2008.

[3] http://links.uwaterloo.ca/Repository.html.

[4] Yen-Kuang Chen, Jatin Chhugani, Pradeep Dubey, Christopher J. Hughes, Daehyun Kim, Sanjeev Kumar, Victor W. Lee, Anthony D. Nguyen, and Mikhail Smelyanskiy. Convergence of recognition, mining, and synthesis workloads and its implications. *Proceedings of the IEEE*, 96(5):790–807, May 2008.

[5] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal, The*, 27(3):379–423, July 1948.

[6] Peter Elias. Predictive coding–I. *Information Theory, IRE Transactions on*, 1(1):16–24, March 1955.

[7] David Salomon. *Data Compression: The Complete Reference*. Springer, 4th edition, December 2006.

[8] Khalid Sayood. *Mathematical Preliminaries for Lossy Coding*, pages 217–250. Elsevier, 2012.

[9] John G. Cleary and Ian H. Witten. Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on*, 32(4):396–402, April 1984.

[10] Alistair Moffat. Implementing the PPM data compression scheme. *Communications, IEEE Transactions on*, 38(11):1917–1921, November 1990.

[11] P. G. Howard. *"The design and analysis of efficient lossless data compression systems, Phd. dissertation, Dept. Comput. Sci., Brown Univ., Providence.* PhD thesis.

[12] Ian H. Witten and Timothy C. Bell. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *Information Theory, IEEE Transactions on*, 37(4):1085–1094, July 1991.

[13] Yong Zhang and Donald A. Adjeroh. Prediction by partial approximate matching for lossless image compression. *Image Processing, IEEE Transactions on*, 17(6):924–935, June 2008.

[14] Krisda Lengwehasatit and Antonio Ortega. Scalable variable complexity approximate forward DCT. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(11):1236–1248, November 2004.

[15] R. J. Cintra and F. M. Bayer. A DCT approximation for image compression. *IEEE Signal Processing Letters*, 18(10):579–582, February 2014, 1402.6034.pdf.

[16] Soontorn Oraintara, Ying-jui Chen, and Truong Nguyen. Integer fast fourier transform (INTFFT). In *IEEE Trans. on Signal Processing*, 2001.

[17] R. J. Cintra. An integer approximation method for discrete sinusoidal transforms. *Circuits, Systems, and Signal Processing*, 30(6):1481–1501, May 2011.

[18] Vaibhav Gupta, Debabrata Mohapatra, Sang P. Park, Anand Raghunathan, and Kaushik Roy. IMPACT: IMPrecise adders for low-power approximate computing. In *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, pages 409–414. IEEE, August 2011.

[19] Jiawei Huang, John Lach, and Gabriel Robins. A methodology for energy-quality tradeoff using imprecise hardware. In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, pages 504–509, New York, NY, USA, 2012. ACM.

[20] Amir Momeni, Jie Han, Paolo Montuschi, and Fabrizio Lombardi. Design and analysis of approximate compressors for multiplication. *Computers, IEEE Transactions on*, 64(4):984–994, April 2015.

[21] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, VLSID '11, pages 346–351, Washington, DC, USA, January 2011. IEEE.

[22] Jeffrey M. Gilbert and R. W. Brodersen. A lossless 2-D image compression technique for synthetic discrete-tone images. In *Data Compression Conference, 1998. DCC '98. Proceedings*, pages 359–368. IEEE, March 1998.

[23] M. Mahoney. Adaptive weighing of context models for lossless data compression, florida tech. technical report CS-2005-16, 2005. Technical report.

[24] 7ZIP 15.09 http://www.7-zip.org/download.html.

[25] http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz.

[26] CMIX -v7 http://www.byronknoll.com/cmix.html.

[27] GZIP 1.6 http://ftp.gnu.org/gnu/gzip/.

[28] INFOZIP http://www.info-zip.org/pub/infozip/Zip.html.

[29] LHA 1.14i http://www2m.biglobe.ne.jp/~dolphin/lha/lha-download.htm.

[30] https://code.google.com/p/lz4/.

[31] Nanozip 0.09 http://nanozip.net/.

[32] LZIP 1.17 lossless data compressor http://www.nongnu.org/lzip/lzip.html.

[33] LZOP 1.03 www.lzop.org/download/lzop-1.03.tar.gz.

[34] PAQ8o6 http://www.mattmahoney.net/dc/.

[35] RAR 5.21 http://www.rarlab.com/download.htm.

[36] XZ 5.2.1 http://tukaani.org/xz/.

[37] ZLIB 1.2.8 compression library http://zlib.net/.

[38] ZPAQ 7.05 http://www.mattmahoney.net/dc/zpaq.html.

[39] ZZIP v0.36c http://archives.damiendebin.net/zzip/download.php.

[40] NConvert for png and gif http://www.xnview.com/en/nconvert/.

[41] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, April 2004.

[42] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402 Vol.2. IEEE, November 2003.

[43] Libjpeg-turbo http://libjpeg-turbo.virtualgl.org/.

[44] Carsten Glasenapp and Hans Zappe. Frequency analysis of maskless lithography system for generation of continuous phase patterns with homogeneous structure depth. *Optical Engineering*, 47(2):023002–023002–7, 2008.