CHAPTER – II

REVIEW OF LITERATURE
2.0 REVIEW OF LITERATURE

Classical methods and recent developments in scanning patterns, segmentation, coding schemes, and data security schemes are described in this literature survey.

2.1 SCANNING PATTERNS

Exploring the content of an image depends on the way in which it is scanned. For example, information obtained by scanning the image horizontally differs from those obtained by scanning it vertically. Since there are several ways to scan the image, there are also several possible interpretations of its content. Thus finding the scan that provides more useful and relevant information of the image can be useful for image processing. In the compression context, efficient scanning must be able to explore most redundancy in the image. In GIF image encoder for example, scans the image line by line while projecting the pixel values in a vector, then it applies dictionary coding (LZW) to remove statistic redundancies. Therefore, horizontal patterns (redundant sequences) are effectively compressed but vertical patterns are not. It is evident that the performance of such coding depends partly on the way in which the image was scanned. Thus, searching a way to find the most suitable scan may be useful for image compression.

In view of this fact, Tarek Ouni et al. (2012), has proposed a space filling curves method which scans the image continuously exactly once and results a single dimensional image data in which intra correlation pixels in an image is translated into an autocorrelation within image sequence. In this model, the image is scanned according to its content analysis. The model input is the original image. The outputs consist in linear high correlated pixels sequence and the adapted scan code. The construction of the scan path is carried out while the evolution of the scan itself. The analysis and scanning are combined. The scanning starts from a given pixel. Next, according to considered analysis function, the next pixel is found. Then the sequence of pixels and the code of the path
(additional information) will be updated. It gives an acceptable compression performance especially with medical images. However, it should be noted that this method is very time consuming. Actual test results show that this methodology works very well for the more significant bit planes, but compression becomes impossible for the less significant bit planes due to the random nature of these image patterns.

The use of Hilbert scan is relatively new in image compression. Hilbert curve is one of the space filling curves, published by Hillbert Peano (1890). The author Biswas et al. (2000) has used this curve in the area of image processing. The Hilbert curve has an one-to-one mapping between an n-dimensional space and an one dimensional space which preserves point neighborhoods as much as possible. An Hilbert scanned gray level image of size M x M can be considered as a space curve with length equal to the square of M. The size of the image provides the resolution of the Hilbert curve. Due to the neighborhood property of the Hilbert scan of a gray level image, long homogeneous segments are found frequently in its Hilbert image. This characteristic of the Hilbert scan makes it suitable for image compression. Examination shows better performance due to the correlation of pixels over large segments but it lacks for smaller segments also for color segments.

Kuo-Tien Lee et al. (2011) has proposed generalized Zigzag scanning algorithm to perform coefficient scanning for irregular regions. This paper suggests generalizing the conventional Zigzag scan model since it is improper to use the conventional zigzag of JPEG images to scan the DCT coefficients. For example, for a region whose height is M and width is N, if N is much larger than M, then the DCT coefficient in the location (0, b) is usually larger than that in (b−1, 0). However, when using the conventional zigzag method, (b−1, 0) is scanned before (0, b) which does not match the rule that the AC coefficient with higher energy should be scanned before that with lower energy. Therefore, a generalized scanning is proposed.
According to this, two rules for the new zigzag scanning algorithms for non-square regions are proposed. First, when determining the scanning order, the height and the width of a region should be considered. Second, as the original zigzag, the AC coefficient to be scanned should be neighboring to the previously scanned AC coefficient as possible. With the two rules, a scoring method is proposed that can be used for determining the zigzag scanning order. The simulation results show that, with the generalized zigzag algorithm, the number of bytes of the compressed image can be obviously reduced by 6% to 12%. This scanning algorithm works well only for irregular size of images.

Raster scanning is a technique for generating or recording the elements of a display image by sweeping the screen in a line-by-line manner. More specifically, it scans the whole area, generally from left to right, while progressing from top to bottom of the image. The scanning process basically transforms a 2-D image representation into a 1-D representation.

Nasir Memon et al. (2000) has analyzed performance of raster scan with random scan, progressive scan, and Hilbert scan, for a lossless compression. Progressive scan divides the image into a series of scan layers. The first scan shows the image at the equivalent of a very low quality setting, and therefore it takes very little space. Following scans gradually improve the quality. Each scan adds to the data already provided, so that the total storage requirement is roughly the same as the final scan. It is a common scan that is used in tandem with progressive (multilayer) coding techniques.

According to Nasir Memon et al. (2000) pixels in the grid are visited in layers; within each layer, the relevant pixels are visited in raster order. In a common implementation of this scheme, pixels in alternate rows and columns are visited in a layer. In layer 1, starting from the pixel at location (1,1) it visits pixels at locations (2i+1, 2j+1) in raster order, i.e., alternate pixels in the odd-numbered rows are scanned in raster
order. In layer 2, alternate pixels located at points \((2i, 2j)\) in the even-numbered rows are scanned. Finally, in layers 3 and 4, the remaining pixels in, respectively, the odd-numbered and the even-numbered rows are scanned in raster order as shown in Fig. 2.1

![Figure 2.1: Progressive scan](image)

One can increase the number of layers by visiting pixels spaced more widely apart in any layer. The crucial point, of course, is that it proceeds from one layer to the next, more and more of the local context becomes available for prediction. The performance of statistical, context-based lossless image coding techniques in conjunction with the Hilbert, raster and progressive scans are analyzed. The result shows that under certain reasonable assumptions, progressive scan out-performs the raster scan, mainly due to its larger context. The random scan is seen to have the worst performance among all the scans considered in this paper.

Chien-Pen Chuang et al. (2012) intended a novel image compression method called adaptive arithmetic coding. The algorithm is disunited into two sub parts. Firstly snake scan is used to convert a 2-D signal into 1-D signal and then arithmetic coding is applied. Snake results into residual data by vanishing the correlation between pixels. However, the compression algorithm achieves better results than conventional raster scan methods which does not use snake scan at first step. Authors analyzed the results among scans like raster scan, random scan, progressive scan and Hilbert scan. Though it is well recognized that Hilbert scan commits better result than other methods yet they deduced
their result in terms of predictive gain and discovered that random scan is lowest in the list.

In diagonal scan, the image is scanned along the anti diagonals beginning with the topmost anti diagonal. Each anti diagonal is scanned from left bottom corner to right top corner. Jung-Ah Choi et al. (2000) has proposed diagonal scanning pattern to compress video using Differential Pixel Value Coding lossless scheme. The result is compared with snake vertical and horizontal scanning patterns. In which, the video frames are divided into several segments and then on each segment the proposed encoding scheme is applied. For the relatively small size video frames i.e 4 x 4 and 8 x 8 sizes, and if the frames are not quantized, vertical snake scan often results in better performance but for the quantized relatively small sized video frames, horizontal snake pattern results better performance. On the other hand the diagonal scan outperforms if the video frames are more than 8 x 8 sizes.

Smila Mohandas et al. (2014) has proposed novel scanning patterns namely Z - Vertical Scan (ZV), and Z – Horizontal Scan (ZH), both are variant of Zigzag scan. The scanning patterns are used to compress still images of grayscale where JPEG image compression scheme is used. The experimental results show that the proposed scanning patterns perform equally with conventional Zigzag scanning pattern but the case is not true for color images.

In a spiral scan a single circle or ring of a spiral or helical curve on an image plane winds around a fixed center point at a continuously decreasing distance from the point. Łukasz Błaszak et al. (2005) has used this scanning pattern in video compression. For a given spiral scan aspect, a region of interest technique is used to define the starting point of the spiral scan. In which video frames are divided into number of smaller size frames, and for each frame the centre of the frame is considered as a starting point for the spiral scan. The compression efficiency of the spiral scan is compared with raster scan. The
comparative tests made for single-layer grayscale images prove that this performance is very similar for both scans and the implementation complexity is also very similar for both scans. For spiral scan, the number of mathematical operations performed during data conversion from 2-D to 1-D is 1% less than the raster scan. The code length for full implementation is about 5% greater for the spiral scan solution in compare to the standard raster scan solution.

2.2 SEGMENTATION

Image segmentation is the process of partitioning a input data into multiple parts. If the input data is an image of size N x N then the image is divided into usually several 8 x 8 chunks. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

Batenburg et al. (2009) has proposed a simplest method of image segmentation is called the threshold method. The key of this method is to select the threshold value. Pixels with values less than threshold have been placed in one category, and the rest have been placed in the other category. The boundaries between adjacent pixels in different categories have been superimposed in white on the original image.

In a region-based segmentation algorithms operate iteratively by grouping together pixels which are neighbors and have similar values and splitting groups of pixels which are dissimilar in value. Split-and-merge algorithm is a popular region based algorithm proposed by Horowitz et al. (1976). The algorithm operates in two stages, and requires a limit to be specified for the maximum variance in pixel values in a region. The first stage is the splitting process. Initially, the variance of the whole image is calculated. If this variance exceeds the specified limit, then the image is subdivided into four quadrants. Similarly, if the variance in any of these four quadrants exceeds the limit it is further subdivided into four. This continues until the whole image consists of a set of squares of varying sizes, all of which have variances below the limit.
Barghout et al. (2013) has proposed a clustering method, in which a K-means algorithm is an iterative technique that is used to partition an image into K clusters. According to this, first K cluster centers has been picked, either randomly or based on some heuristic method, after that assign each pixel in the image to the cluster that minimizes the distance between the pixel and the cluster center then re-compute the cluster centers by averaging all of the pixels in the cluster. Finally repeatedly do assigning the pixels to cluster center and recalculating the cluster center until convergence is attained.

Rajesh Gothwal et al. (2014) has proposed a RGB channel based segmentation algorithm. In which split the input RGB image into Red channel, Green channel and Blue channel. Region growing process is applied to Red channel, Green channel and Blue channel separately with seed pixels. Then the region mean value of the region is calculated. At starting step region mean value is set to pixel intensity value. It uses the 8-neighbor connectivity similarity index for region growing process. Regions are grown iteratively until the pixel distance is less than threshold value. Pixel distance is difference of intensity of neighbor pixel and region mean value. The region mean is updated when a pixel is included to region. The threshold value (µ) = 0.2 is used during region growing process. When region growing process stops, construct labeled matrix and then region growing process is repeated for another seed pixel and so on. The above process is repeated for Red, Green and Blue channel and three labeled matrix are obtained. The each labeled matrix are divided into n x n size where n=2^1, 2^2, 2^3, 2^4 and so on, in general n=3. In this thesis RGB color channel based segmentation is used during the compression since its simplicity, fast and suitable for any type of images.

2.3 MATHEMATICAL TRANSFORMS IN DATA COMPRESSION

Ahmed et al. (1974) has proposed a mathematical transform called discrete cosine transform which is a key for many data compression processes in recent years. Liu
Chien-Chih et al. (2005) has used this transform for encoding JPEG 2000 image formats. The DCT is in a class of mathematical operations that includes the well known Fast Fourier Transform (FFT), as well as many others. The Discrete Fourier Transform (DFT) and discrete cosine transform are two variants of FFT and are widely used in digital signal processing. The basic purpose of these transforms is to take a signal and transform it from one type of representation to another. One disadvantage of using DFT for some applications is that the transform is complex valued, even for real data whereas DCT does not have this problem.

The DCT is a separate transform and not the real part of the DFT. It is widely used in image and video compression applications. The DCT can be used to convert the signal (spatial information) into numeric data ("frequency" or "spectral" information) so that the information exists in a quantitative form that can be easily manipulated for compression. The equation 4.21 is a DCT which is used on 2-D data to convert highly correlated data set in a relatively independent data set and results a DC coefficient, which is higher than all other values and a series of AC coefficients where some has zero value at high frequency, represents much redundant information and small values at low frequency can represent the same data much more efficiently. Since DCT is a lossless transformation, the function Inverse DCT (IDCT) which is mentioned in Equation (4.23) results original 2-D data. Since all natural images exhibits spatial redundancy, not all coefficients in the transformed array have significant values. This can be demonstrated by an example. For example consider 8 x 8 block of Lena image, whose pixel intensities are shown in Fig. 2.2. Because DCT is designed to work on pixel values from -128 to +127 each entry in the matrix can be leveled off by subtracting 128 then compute DCT for each entry using Equation (4.21).

The transformed matrix values are shown in Fig. 2.3. It is worth noting that most of the transformed coefficients have very small values and only a few coefficients have
higher magnitudes. This shows the energy compaction capabilities of DCT. Selection of block-sizes in DCT is an important consideration.

<table>
<thead>
<tr>
<th>166</th>
<th>162</th>
<th>162</th>
<th>160</th>
<th>155</th>
<th>163</th>
<th>160</th>
<th>155</th>
</tr>
</thead>
<tbody>
<tr>
<td>166</td>
<td>162</td>
<td>162</td>
<td>160</td>
<td>155</td>
<td>163</td>
<td>160</td>
<td>155</td>
</tr>
<tr>
<td>166</td>
<td>162</td>
<td>162</td>
<td>160</td>
<td>155</td>
<td>163</td>
<td>160</td>
<td>155</td>
</tr>
<tr>
<td>166</td>
<td>162</td>
<td>162</td>
<td>160</td>
<td>155</td>
<td>163</td>
<td>160</td>
<td>155</td>
</tr>
<tr>
<td>166</td>
<td>162</td>
<td>162</td>
<td>160</td>
<td>155</td>
<td>163</td>
<td>160</td>
<td>155</td>
</tr>
<tr>
<td>161</td>
<td>160</td>
<td>155</td>
<td>159</td>
<td>154</td>
<td>154</td>
<td>156</td>
<td>154</td>
</tr>
<tr>
<td>159</td>
<td>163</td>
<td>158</td>
<td>163</td>
<td>155</td>
<td>155</td>
<td>156</td>
<td>152</td>
</tr>
<tr>
<td>159</td>
<td>162</td>
<td>162</td>
<td>160</td>
<td>153</td>
<td>153</td>
<td>153</td>
<td>151</td>
</tr>
</tbody>
</table>

Figure 2.2: 8 x 8 block

<table>
<thead>
<tr>
<th>248</th>
<th>19</th>
<th>3</th>
<th>4</th>
<th>-7</th>
<th>9</th>
<th>1</th>
<th>-7</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>-2</td>
<td>3</td>
<td>6</td>
<td>-3</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>-4</td>
<td>2</td>
<td>-2</td>
<td>-3</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 2.3: DCT Coefficients of 8 x 8 block

The images should be so subdivided that the level of redundancies between the adjacent sub images are reduced to an acceptable level and the dimension of the sub-images should be an integer power of 2. Increasing the block size reduces adjacent block redundancies and reduces mean square reconstruction error using truncated and quantized coefficients, but involves more computations. Most popular block sizes used in image
compression are 8 x 8 and 16 x 16. Since DCT is a lossless transform [Raid et al., 2014], the inverse DCT, equation 4.23, if it is applied on each entry in the Fig. 2.3 results Fig. 2.2.

Wavelet transform (WT) represents an image as a sum of wavelet functions (wavelets) with different locations and scales. Any decomposition of an image into wavelets involves a pair of waveforms: one to represent the high frequencies corresponding to the detailed parts of an image (wavelet function ψ) and one for the low frequencies or smooth parts of an image (scaling function Φ). DWT is a multi-resolution decomposition scheme for input signals. The original signals are first decomposed into two subspaces, low-frequency (lowpass) sub band and high-frequency (high-pass) sub band. For the classical DWT, the forward decomposition of a signal is implemented by a low-pass digital filter H and a high-pass digital filter G. Both digital filters are derived using the scaling function Φ(t) and the corresponding wavelets Ψ(t). The system downsamples the signal to half of the filtered results in the decomposition process. The transform is based on a wavelet matrix, which can be computed more quickly than the analogous Fourier matrix but it has certain accepted loss of quality in image compression [kanika et al., 2012].

Nibouche Mokhtar et al. (2001) has proposed a variant of DWT called Field Programmable Gate Array (FPGA) based Discrete Wavelet Transforms System. The methodology allows a signal/image processing application developer to generate FPGA device configurations for DWT at a high level. This architecture works in a non-separable fashion using a parallel filter structure with distributed control to compute all the DWT resolution levels. The preliminary results are very promising; however, it lacks to minimize time complexity while performing different arithmetic operations, different wavelet analysis and synthesis schemes.
Steve Haynal et al. (2011) has implemented a FFT which is an efficient implementation of DFT and is used in digital image processing. In FFT, it decomposes an image into its real and imaginary components which is a representation of the image in the frequency domain. If the input signal is an image then the number of frequencies in the frequency domain is equal to the number of pixels in the image or spatial domain. The inverse transform re-transforms the frequencies to the image in the spatial domain. The advantage of representing an image in the frequency space is that performing some operations on the frequencies is much more easier than doing the same in the image space. But many of the convolutions operations increase the computational complexities than the other transforms.

To avoid the artifacts generated by block transforms, full-frame methods, in which the transform is applied to the whole image as a single block, have been investigated in medical imaging research. The tradeoff is the increased computational requirements and the appearance of ringing artifacts (a periodic pattern due to the quantization of high frequencies). Bramble et al. (2000) used full-frame Fourier transform compression on 12 bpp digitized hand radiographs at average rates from about 0.75 bpp to 0.1 bpp with no significant degradation in diagnostic quality involving the detection of pathology characterized by a lack of sharpness in a bone edge.

2.4 COMPRESSION METHODS

Huffman coding is a typical statistical compression algorithm proposed by David Huffman (1952). Huffman coding eliminates data redundancy using the statistical properties of data values, or in other words by encoding densely those data values that appear more frequently. After coding, more frequent values have narrower or at least equal length codeword than less frequent values. Huffman coding uses lossless compression and it widely used in data compression in recent years. The idea behind Huffman coding is that certain pieces of information in the input are more likely to appear than others. To conserver space common shorter symbols are assigned to this
information. For compressing images and video type of data JPEG Huffman compression method is used widely.

Transform coding algorithm typically initiate by partitioning the original image addicted to sub images (blocks) of small size (usually 8 x 8). For each block the change coefficients are calculated, successfully converting the original 8 x 8 array of pixel values into an array of coefficients closer to the top left corner usually contain most of the information needed to quantize with encode the image with little perceptual distortion. The resultant coefficients are then quantized and the output of the quantizer is used by a symbol encoding technique(s) to produce the output bit stream representing the encoded image. At the decoder’s side, the reverse process takes place, with the obvious difference that the dequantization stage will only generate an approximated version of the original coefficient values; in other words, whatever loss is introduced by the quantizer in the encoder stage is not reversible. DCT, DWT are some popular examples for transform coding.

A Dictionary-based compression is a kind of static coding which encode common values using pointers to dictionaries which keep those values. A popular variation of dictionary-based compression algorithms is LZ77, also referred as LZ++ was proposed by Ziv et al. (1977). Assuming there is a sequence of data to be compressed. A virtual window (dictionary) is defined with a constant length and added into the head of the sequence. That means the window is part of the sequence and its content changes due to its movement in the sequence. During the compression process, the window moves forward in the data sequence and keeps comparing the sequence data behind the window with its own content for the longest match. A match occurs when a data sequence behind the window is as same as part of its content. The matched data are encoded for compression purpose by using the corresponding pointer information of the window. For example, there is a sequence of character MFABCABCFEDF. Define the length of window to be 4. While the window moves to the second character of the sequence, its
content is FABC. The window searches for the match from the first character behind it which is A. In this case, the first 3 characters behind the window (ABC) is as same as part of its content which is the longest match. Then a combination of (1, 3, E) is used to represent ABC. Where 1 is the pointer of A in the window (as the pointer of F is 0), 3 presents the length of the matched data ABC, E is the first unmatched character behind the window. In the next step, the dictionary moves forwards to be ABCE and searches for a new match from character D. The drawback in this method is it requires additional memory temporally to keep dictionary.

LZW encoding is proposed by Abraham Lempel et al. (1984) which is an improved version of LZ77. It is the algorithm of the widely used Unix file compression utility compress, and is used in the GIF image format. According to this, a dictionary is initialized to contain the single-character strings corresponding to all the possible input characters. The algorithm works by scanning through the input string for successively longer substrings until it finds one that is not in the dictionary. When such a string is found, the index for the string without the last character is retrieved from the dictionary and sent to output, and the new string is added to the dictionary with the next available code. The last input character is then used as the next starting point to scan for substrings. In this way, successively longer strings are registered in the dictionary and made available for subsequent encoding as single output values. The algorithm works best on data with repeated patterns, so the initial parts of a message will see little compression.

gzip is a compression utility designed to be a replacement for compress. Its main advantages over compress are much better compression and freedom from patented algorithms. It has been adopted by the GNU project and is now relatively popular on the Internet. gzip was written by Jean-loup Gailly and Mark Adler (1993) for the decompression code. gzip which is based on the DEFLATE algorithm, which is a combination of LZ77 and Huffman coding. DEFLATE was intended as a replacement for LZW. gzip is normally used to compress just single files. Compressed archives are
typically created by assembling collections of files into a single tar archive, and then
compressing that archive with gzip. The final .tar.gz or .tgz file is usually called a tarball.
gzip is not to be confused with the ZIP archive format, which also uses DEFLATE. The
ZIP format can hold collections of files without an external archiver, but is less compact
than compressed tarballs holding the same data, because it compresses files individually
and cannot take advantage of redundancy between files.

Deflate was designed by Philip Katz (2003) as part of ZIP file format and
implemented in PKZIP software. It finds duplicated strings in the input data. The second
occurrence of a string is replaced by a pointer to the previous string, in the form of a pair
(distance, length). Distances are limited to 32K bytes, and lengths are limited to 258
bytes. When a string does not occur anywhere in the previous 32K bytes, it is emitted as a
sequence of literal bytes. Literals or match lengths are compressed with one Huffman
tree, and match distances are compressed with another tree. The trees are stored in a
compact form at the start of each block. A block is terminated when deflate() determines
that it would be useful to start another block with fresh trees. Duplicated strings are found
using a hash table. All input strings of length 3 are inserted in the hash table. A hash
index is computed for the next 3 bytes. If the hash chain for this index is not empty, all
strings in the chain are compared with the current input string, and the longest match is
selected. The hash chains are searched starting with the most recent strings, to favor small
distances and thus take advantage of the Huffman encoding. The hash chains are singly
linked. There are no deletions from the hash chains; the algorithm simply discards
matches that are too old. To avoid a worst-case situation, very long hash chains are
arbitrarily truncated at a certain length, determined by a runtime option. Therefore,
deflate() does not always find the longest possible match but generally finds a match
which is long enough.

The Lempel–Ziv–Markov Algorithm (LZMA) is an algorithm used to perform
lossless data compression used in the 7z format of the 7-Zip archiver, implemented by
Abraham Lempel *et al.* (1977) but was placed by Igor Pavlov (2008) in the public domain, with the GNU release of 7-zip version 4.62. LZMA uses a dictionary compression algorithm is a variant of LZ77 with huge dictionary sizes and special support for repeatedly used match distances whose output is then encoded with a range encoder, using a complex model to make a probability prediction of each bit. The dictionary compressor finds matches using sophisticated dictionary data structures, and produces a stream of literal symbols and phrase references, which is encoded one bit at a time by the range encoder.

The adaptive Huffman coder operates very similarly to the standard Huffman algorithm; however, instead of a static measurement of the entire input sequence's statistics, a dynamic, and cumulative that is from the first symbol to the current symbol estimate of the sequence's probability distribution is used to encode (and decode) each symbol. In contrast to the standard Huffman coding approach, the adaptive Huffman algorithm requires this statistical analysis at both the encoder and decoder.

Vitter *et al.* (1987) has proposed an algorithm for dynamic Huffman coding. A standard Huffman coder has access to the probability mass function of its input sequence, which it uses to construct efficient encodings for the most probable symbol values. In the prototypical example of file-based data compression, for example, this probability distribution can be calculated by doing histogram the input sequence, counting the number of occurrences of each symbol value. This histogram is used to generate a Huffman tree, example for how to construct Huffman tree is explained in section 4.10. The tree is arranged by decreasing weight or probability of occurrence in the input sequence; leaf nodes at the top represent the most probable symbols, which therefore receive the shortest representations in the compressed data stream. The tree is then saved along with the compressed data and is subsequently used by the decompressor later to regenerate the (uncompressed) input sequence again. The adaptive Huffman coder's structure is quite similar; it uses a similar tree-based representation of the input
sequence's statistics to select efficient encodings for each input symbol value. The main difference is that, as a streaming implementation of the algorithm, no a priori knowledge of the input's probability mass function is available; the sequence's statistics must be estimated on the fly. If one is to use the same Huffman encoding scheme, this means that the tree used to generate each symbol's encoding in the compressed stream must be built and maintained dynamically as the input stream is processed.

The Vitter algorithm is one way of accomplishing this; as each input symbol is processed, the tree is updated, maintaining its characteristic of decreasing probability of symbol occurrence as you move down the tree. The algorithm defines a set of rules for how the tree is updated over time, and how the resulting compressed data is encoded in the output stream. As the input sequence is consumed, the tree's structure should represent a more and more accurate description of the input's probability distribution. In contrast to the standard Huffman coding approach, the decompressor does not have a static tree to use for decoding; it must perform the same tree-maintenance functions continuously during the decompression process.

Entropy encoding, is a coding format that involve transfer codes to cryptogram so as to go with system lengths with the probability of the symbols. Obviously, entropy encoders are used to compress data by replace cryptogram represented by the same length codes by means of symbols represented by codes proportional to the negative logarithm of the probability. Therefore, the most common symbols use the shortest codes. Witten, Ian H., et al. (1987) has proposed a kind of entropy coding called Arithmetic Coding Technique (ACT). It is the mainly authoritative method for statically lossless encoding. The main aim of ACT is to provide code words with a perfect length. ACT is the most efficient method to code symbols according to the probability of their occurrence. The average code length is very close to the possible minimum given by information theory. The ACT assigns an interval to each symbol whose size reflects the probability for the
appearance of this symbol. The code word of a symbol is an arbitrary rational number belonging to the corresponding interval.

Run Length Encoding is a popular coding technique which its origin difficult to trace to any one individual. Several variants exist but a scheme described by Held (1983) that combines both run length values and unencoded source data in the same output stream appears to be the most common. In this scheme a special, reserved character or symbol in the output stream is used to indicate that the next two data items (usually bytes) are a source character followed by a run value indicating the number of consecutive times that source character occurs, a total of three bytes in all. Encoding would not be worthwhile, and would be precluded (i.e., the special character would not appear), unless the number of repeated occurrences of the source character exceeds 3. For an instance, if the special character is "#" then the character sequence AAAAAABBCCCCCCCCC would be encoded as #A5BB#C9, a 50% reduction in size.

A two-dimensional run length encoding scheme for general image compression has been developed by Prachat et al. (1987). This scheme encodes either largest overlapping, or largest non-overlapping, rectangular blocks of pixels of the same color by their corner coordinates. Both schemes, however, rely on some way of marking pixels as either processed or unprocessed as the image is encoded. Since pixels in binary images are represented by bits, which can only assume two values, there is no way to mark a pixel as having already been scanned without introducing a secondary array.

Gopal Lakhani (2004) has proposed a version of JPEG baseline Huffman coding algorithm to compress JPEG still images. According to this, an image is segmented based on RGB color spaced then each color spaces are divided into 8x8 several image chunks. On each chunk a transform coding called DCT is applied which results a DCT matrix. On each DC matrix a quantization is performed which results DC and AC coefficients. The DCPM is performed on DC coefficient whereas zero run length is performed on AC
coefficients. Finally, Huffman coding is performed on output of DPCM and RLE respectively to get compressed image. In this thesis, focuses on using these coding schemes have been given and working principle of them is explained in section 4.8 and 4.9 respectively in detail.

A modified run length encoding was proposed by VidyaSagar et al. (2013). Run length encoding (RLE) of text strings replaces sequences of characters by a number (the number of repetitions of the character) followed by the character that is repeated. Thus, the string AAAAAABAAABBCCEEEEEE would be replaced by 4AB3ABBC5E. Note that BB is not replaced by 2B since that is no shorter. Under some circumstances, Modified Run Length Encoding (MRLE) can do better by replacing repeated adjacent substrings in a similar way. Now, need a pair of numbers associated with a repeated substring: the number of repetitions, and the length of the substring. The string ABABAB (length 6) can be written as 3 2 AB (length 4), being 3 repetitions of a length-2 substring (AB). From longest to shortest substring length, the algorithm will scan the target string from left to right looking for sequences of substrings of that length. When a repetition of a substring is found, it is replaced the two numbers associated with the substring, followed by the substring itself. This process is repeated until finally all length-1 substrings have been considered.

Sarika et al. (2013) has proposed an improved run length encoding. According to this, a bit stuffing is the process of inserting non information bits into data to break up bit patterns to affect the synchronous transmission of information. It is widely used in network and communication protocols, in which bit stuffing is a required part of the transmission process. The location of the stuffed bit is communicated to the receiving end and extra bits are removed from the original data at the receiving end. The receiver end requires the information of maximum allowable consecutive bits Bit stuffing works to limit the number of consecutive bits of the same value included in the transmitted data for run-length limited coding. This procedure includes a bit of the opposite value after the
maximum allowed number of consecutive bits of the same value. Stuffed bit should not
confuse with overhead bits. In improved RLE, 15 consecutive ones are represented by 4
bits and 17 consecutive ones are represented by 5 bits. In this paper the length of the
sequence after which the bit will be stuff is 15 consecutive ones/zeros.

20 consecutive ones are represented by 5 bits of run

```
1111111111111111
```

10100, 1

By Bit Stuffing is used 4 bits are sufficient to represent run

```
111111111111101111
```

|111, 1|

Therefore by bit stuffing it limits the more number of consecutive ones/zeros
which decreases the number of bits to represent the run value and decreases the memory
stack and increase in turn increases the transmission speed.

JPEG baseline Huffman coding uses RLE quite effectively on the coefficients that
remain after transforming and quantizing image blocks. The basic principle is to replace
sequences of successive identical symbols by three elements: a single symbol, a counter,
and an indicator which gives the interpretation of the other two elements. Run length
coding is extremely fast on the other hand and can compress quite well in case of suited
data. Run-length coding chooses to use a series of (run, level) pairs to represent a string
of data. For example: For an input data array: {2, 0, 0, 0, 5, 0, 3, 7, 0, 0, 0, 1….}, the
output (run, level) pairs are: {(0, 2), (3, 5), (1, 3), (0, 7), (3, 1)…}. Run here means how
many zeros come before the next non-zero data. Level is the value of the non-zero data.

2.5 DATA SECURITY

With the rapid development of multimedia and network technologies, the security
of multimedia becomes more and more important, since multimedia data are transmitted
over open networks more and more frequently. Typically, reliable security is necessary to content protection of digital images. Encryption can be defined as the art of converting data into coded form which can be decode by intended receiver only who poses knowledge about the decryption of the ciphered data. Encryption can be applied to text, image, and video for data protection. On the other hand, an Image compression is an application of data compression that encodes the original image with few bits. The objective of image compression is to reduce the redundancy of the image and to store or transmit data in an efficient form. Even though both data compression and encryption are methods to transform data into different representation, the goals tried to achieve by them are different. Data compression is done with the intension of decreasing the size of data, where encryption is done to keep the data secret from third parties. Data compression offers an approach for reducing communication costs, at the same time it is vulnerable to attack during the transmission. If it is compromised then it is not possible to get actual data during the decompression. Therefore security is needed to preserve the compressed data. Compression always relies on high redundant data in order to gain size reduction. Since encryption destroys redundancy, the compression algorithm would not be able to give much size reduction, if it is applied on encrypted data. For that reason, compression before encryption is the highly preferable order. Even though there are several encryption methods are available, non linear chaotic method is getting popular. In this thesis, a focus on how to use Non linear chaotic map for encrypting on compressed data has been given.

The existing chaos based algorithms operate on two stages: the shuffling stage and the substitution stage. In the shuffling stage, the position of the pixels from the original image is changed by chaotic sequences or by some matrix transformation, such as Arnold transformation, magic square transformation, and so forth. These shuffling algorithms can be easily realized. Since these shuffling algorithms just involve changing the position of the pixels but not changing the pixel values it leads to histogram of the encrypted image same as the original image, thus the security of the image is threatened by
statistical analysis such as Arnold transformation, magic square transformation, and so forth. These shuffling algorithms can be easily realized. Since these shuffling algorithms just involve changing the position of the pixels but not changing the pixel values it leads to histogram of the encrypted image same as the original image, thus the security of the image is threatened by statistical analysis. Chaotic Image encryption is a branch of cryptography in which image data has been encrypted with the help of cryptographic tools based on chaos theory.

Matthews (1989) has proposed first chaos-based encryption scheme, and then which is adopted into image encryption by Fridrich (1997). Since then, many chaos-based image encryption algorithms have been designed to realize secure communications. Chaos Theory states that for certain systems, similar initial conditions will end up with very different end conditions. In other words, it is the study of dynamical systems with complex behavior. Chaotic systems have many important properties, such as the sensitive dependence on initial conditions and system parameters, pseudorandom property, no periodicity and topological transitivity, etc. Most properties meet some requirements such as diffusion and mixing in the sense of cryptography. Therefore, chaotic cryptosystems have more useful and practical applications.

Nonlinear is an adjective that applies to a system to indicate that output responses are not linearly related to input changes. In a linear system, doubling an input doubles the change in the output. An example of a linear relation is the amount of work a person can do versus the time that the person works. If the person works twice as long, he or she can accomplish twice as much. Linear systems are always exactly solvable. A nonlinear system may or may not be solvable. If it is solvable then it cannot be chaotic because it would be predictable. Nonlinear system is a chaotic system in which output of the system is totally unpredictable and dynamic since it uses chaotic maps. The chaotic maps are getting more attention recently in cryptanalysis since it is easy to solve but the result is bifurcation where at every point it changes from one functional behavior to another
functional behavior. With more than one control parameters and initial conditions, high dimensional chaotic systems are most complex and have a big key space. However, complex calculations make the encryption algorithm too slow. To overcome these drawbacks, a nonlinear chaotic map is adopted. Encryption method based on nonlinear chaotic algorithm uses tangent and power function to give large key space. The experimental results show that this chaotic map has more complex chaotic behaviors than the linear chaotic map. The section 4.14 explains how to use of the Non linear chaotic map in detail.

Ahmed et al. (2007) has proposed an Efficient Chaos Based Feedback Stream Cipher (ECBFSC) for image cryptosystems. It works using an iterative cipher mechanism that is based on the logistic function. The encryption module encrypts the image pixel-by-pixel, in each iteration, with the values of previously encrypted pixels. This feedback property, combined with the external secret key of 256-bit, makes stream cipher robust against cryptanalytic attacks. This simple implementation of ECBFSC achieves high encryption rates on general-purpose computers. The proposed stream cipher is based on the use of a chaotic logistic map and an external secret key of 256-bit. The initial conditions for the chaotic logistic map are derived using the external secret key by providing weightage to its bits corresponding to their position in the key. Further, new features of the proposed stream cipher include the heavy use of data-dependent iterations, data-dependent inputs, and the inclusion of three independent feedback mechanisms.

The results of several experimental, key space analysis, statistical analysis, and key sensitivity tests show that the proposed ECBFSC for image cryptosystems provides an efficient and secure way for real-time image encryption and transmission from the cryptographic viewpoint.

Recently, a Feedback Chaotic Synchronization (FCS) for designing a real-time secure symmetric encryption scheme has been implemented by Azzaz et al. (2010). The
basic principle of encryption with chaos is based on the ability of some dynamic systems to produce sequence of numbers that are random in nature. This sequence is used to encrypt messages. For decryption, the sequence of random numbers is highly dependent on the initial condition used for generating this sequence. A very minute deviation in the initial condition will result in a totally different sequence. This sensitivity to initial condition makes chaotic systems ideal for encryption.

Sathishkumar et al. (2011) has proposed chaos based circular mapping. The given original images are divided into blocks and zigzag scanning is performed. To encrypt the image, chaos based circular shift mapping procedure and scrambling based on cryptography technique are adopted. The efficiency of the proposed scheme is evaluated in terms of statistical measures such as cross correlation and Peak Signal to Noise Ratio (PSNR). All the simulation and experimental analysis show that the proposed image encryption system has very large key space, high sensitivity to secret keys, has information entropy close to ideal value 8 and has low correlation coefficients close to the ideal value 0, but time to encrypt in not realistic for large size data.

Somaya et al. (2012) has proposed an approach for encrypting images based on chaotic maps and in which a wavelet transform is used for image decomposition and some external encryption keys and one-dimensional chaotic map are used to encrypt the important part. In this work, a 2-D grayscale image is divided into sub bands and sub sampled using a discrete wavelet transform. The most important component of these sub bands is the Low-Low component because it is much more similar to the original image; Low-Low is called the approximate image. To provide security for this component, the second step is to encrypt this component using a chaos-based image-encryption algorithm. The chaotic map parameters are the initial values which are also used to select the external encryption keys. The basic idea of this work is to show the influence of using multiple keys in increasing security by increasing the number of external encryption keys in each time such that the important part of the image is encrypted with several external
encryption keys; the encryption is done by a new 1-D chaotic map and the results show that the correlation coefficients between the original image and the encrypted image are decreased when the number of external encryption keys is increased, and this increases the security.

Ansam Osama et al. (2012) has proposed a block shuffling, pixel shuffling and chaotic map encryption scheme. In natural images the values and position of the neighboring pixels are strongly correlated. The proposed method breaks this correlation increasing entropy of the position and entropy of pixel values using block shuffling, pixel shuffling and encryption by chaotic logistic maps respectively. The image is divided into blocks and then performs block based shuffling using Arnold Cat transformation. Later on the block shuffled image, pixel shuffling is performed using certain number of iterations of Arnold cat map. The Arnold Cat Map takes concepts from linear algebra and uses them to change the positions of the block/pixel values of the original image. The result after applying the Arnold Cat Map will be a shuffled image that contains all of the same pixel values of the original image but with shuffled positions. An external secret key of 80-bit and two chaotic logistic maps are employed. The initial conditions for the both logistic maps are derived using the external secret key by providing different weightage to all its bits. To make the cipher more robust against any attack, the secret key is modified after encrypting each block.

The image is divided into blocks and then performs block based shuffling using Arnold Cat transformation. The Arnold Cat Map (ACM) takes concepts from linear algebra and uses them to change the positions of the pixel values of the original image. The result after applying the ACM will be a shuffled image that contains all of the same pixel values of the original image but with shuffled positions. The ACM is a discrete system that stretches and folds its trajectories in phase space. Let \( X = \begin{bmatrix} x \\ y \end{bmatrix} \) where \( X \) is a vector, then the ACM transformation is shown in Equation (2.1).
\[ \Gamma: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 1 & p \\ q & p + 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \text{mod } n \] (2.1)

Some conditions for the map are that \( p \) and \( q \) are positive integers and \( \det \begin{bmatrix} 1 & p \\ q & p + 1 \end{bmatrix} = 1 \) which makes the map area preserving. The experimental study proved that Chaos theory is an excellent alternative to provide a fast, simple, and reliable image encryption scheme that has a high enough degree of security. Both security analysis and experiments show that, taking into account the trade-off between attack expense and information value as well as other issues such as operational speed, computational cost, and implementation simplicity, this kind of chaos-based image encryption schemes are very practical.

The XOR operator is extremely common as a component in more complex ciphers. The XOR operation is a bit-wise function that maps an element of \( \{0, 1\} \times \{0,1\} \) onto the set \( \{0, 1\} \) as \( 0 + 0 = 0, \ 0 + 1 = 1, \ 1 + 0 = 1, \) and \( 1 + 1 = 0. \) If the second operand to be a key value, the XOR operation can be thought of as being simply a bit-level substitution based upon the bit values of the key. With such assumption, XOR sends a 0 or 1 to itself when the corresponding key bit is 0 and inverts a 0 into a 1 and a 1 into a 0 when the corresponding key bit is 1. XOR exhibits the following properties: \( a + 0 = a, \ a + a = 0, \) and thus \( a + b + b = a. \)

The last property implies that using a fixed key value, the XOR operation can be applied to encipher a plaintext, which can then be recovered by simply applying the XOR operation to the cipher text and the same key value. This property has led to the proliferation of many variants of weak encryption methods that solely rely on the simple XOR operation and thus are easily breakable. A fixed-length key, \( K, \) is used for the XOR of some plaintext blocks. Knowing a block of plaintext, \( P, \) and its XOR transformation directly leads to \( K, \) by way of XORing the plaintext with the corresponding cipher text. C
=P +K that is, P +C =P +P +K =K. Similarly, by knowing two cipher text blocks alone one can XOR them together to yield the XOR of the corresponding plaintext blocks as follows: C1 +C2 =P1 +K +P2 +K =P1 +P2. Examining the bit patterns of P1 + P2 can easily result in recovering one of the plaintexts. The latter can then be XORed with its cipher text to yield the key stream.

By transforming one bit at a time the XOR operation lends itself well to a class of encryption algorithms known as stream ciphers. In contrast, block ciphers divide a plaintext into identical size blocks, generally of length greater or equal to 64 bits, then apply the same encryption transformation to encrypt each block at a time. Stream ciphers are geared for use in situations where memory buffering is limited or when characters are individually transformed as they show up at an endpoint of a transmission medium. Because they generally transform plaintext bits independently from one another, error propagation remains limited in the event of transmission anomaly.

Key iterations are often performed in encryption. It can be used only with symmetric encryption. If a file/text is encrypted using passphrase, HASH is calculated and encrypted with any one encryption method for 'n' number of times. If iterations selected is 100 then this is done 100 times. This is done to defend against brute-force attack. Key iterations have no use with public-key encryption. Some applications by default take 100 iterations. Few applications by default take 2000 iterations. The iteration count is to make the operation deliberately more expensive to defend against attack. The iterations are there to slow down the attacker. Let's suppose that the function is properly salted, so that the best an attacker can do is indeed trying out all potential passwords until a match is found. It is defined the password entropy to be equal to n bits if the best the attacker can do is, on average, to try $2^{n-1}$ passwords: the attacker knows which passwords are more often chosen by users, and tries passwords in the "optimal" order, and this will work with that average success. The use of "bits" comes from the fact that if the password is a sequence of k bits, such that all the possible $2^k$ sequences stand an equal
chance of being selected by the user, then the entropy is, by the definition above, equal to k bits. Then r iterations are applied in an attempt to slow down the attacker. The computational cost for the attacker is then, on average, r·2^{n-1}. Thus, the goal is to get r and n sufficiently high so that this cost is prohibitive, and will not be achievable by the attacker. Traditionally, cryptographers have used "80 bits", i.e. 280 "simple operations", as the limit. This 80-bit limit was already used 20 years ago. Nowadays, "128 bits" is used as the "safe limit" for now and the next decades as well.

Color quantization is the process of reducing the number of distinct colors used in an image. The main reason we may want to perform this kind of compression is to enable the rendering of an image in devices supporting only a limited number of colors (usually due to memory limitations). Obviously all compressions come with a cost. In this case the resulting image may differ too much from the original one. Hence the goal of the color quantization is to obtain a compressed image as similar as possible to the original one. The key factor for achieving this is the selection of the color palette by choosing the colors that most summarizes the original image. The most common techniques reduce the problem of color quantization into a clustering problem of points where each point represents the color of a pixel. It consists in creating the palette by selecting a representative point for each cluster. After that the compression simply remaps all the colors into their cluster representative. A color can be represented as a point in an n-dimensional space called color space. Most commonly the space is 3-dimensional and the coordinates in that space can be used to encode a color. The RGB (abbreviation of red-green-blue) color space is by far the most common and used color space. The idea is that it is possible to create colors by combining red, green and blue. A color in RGB is usually encoded as a 3-tuple of 8 bits each, hence each dimension takes a value within the range [0, 255] where 0 stands for absence of color while 255 stands for full presence of color. The post one common approach to perform color quantization is to use a clustering algorithm. In this case we're going to use k-means that minimizes the within-cluster sum of squared distances between the centroid and the other points of the cluster. Given that
the distance used by the k-means clustering algorithm is the Euclidean distance. It is a natural fit for being applied for color quantization with RGB color space.

The procedure consists in applying the k-means algorithm with a number of centroids that is equal to the number of colors we want the palette to be composed of. Then we use the resulting centroids as the colors that will be part of the palette. After that, each color is remapped to the nearest one among those in the palette as in the random selection. Given the nature of k-means, the nearest color corresponds exactly to the centroid representing the cluster the color is part of. It is important to notice that n pixels of an image having the same color consist in n overlapping points in the color space and not in a single point. Actually in case of random selection it makes no difference, but it does a lot of difference when applying the k-means algorithm.

Binning or discretization is the process of transforming numerical variables into categorical counterparts. An example is to bin values for Age into categories such as 20-39, 40-59, and 60-79. Numerical variables are usually discretized in the modeling methods based on frequency tables (e.g., decision trees). Moreover, binning may improve accuracy of the predictive models by reducing the noise or non-linearity. Unsupervised binning methods transform numerical variables into categorical counterparts but do not use the target (class) information. Equal Width and Equal Frequency are two unsupervised binning methods. Equal Width algorithm divides the data into k intervals of equal size. Equal Frequency algorithm divides the data into k groups which each group contains approximately same number of values. For the both methods, the best way of determining k is by looking at the histogram and try different intervals or groups. Supervised binning methods transform numerical variables into categorical counterparts and refer to the target (class) information when selecting discretization cut points. Entropy-based binning is an example of a supervised binning method. Entropy based method uses a split approach. The entropy (or the information content) is calculated based on the class label. Intuitively, it finds the best split so that the bins are as pure as
possible that is the majority of the values in a bin correspond to have the same class label. Formally, it is characterized by finding the split with the maximal information gain.

2.6. RESEARCH GAP

In the age of digital era information is increasing in an explosive rate. The demand for data transfer today is stretching the original hardware infrastructure of the internet to its limits. Yet, there is a lack of development of reliable compression techniques that can deal effectively with several data types. The following research gap has been identified in this research study.

- Optimal Scanning pattern is required for reordering the input chunks in a single vector
- An universal compression scheme is required for compressing different data format
- It has been identified that compression has to be performed in a multi depth rather a collective approach in order to improve the compression ratio.
- Getting the original data is not possible if the compressed data is corrupted during the transmission via public channel hence a method for secured transmission of compressed data is inevitable.