Simulated Circular Indexing Transform (SCIT) Based Digital Image Watermarking

Gebremichael Girmay[1], D.Lalitha Bhaskari[2]

[1, 2]Department of Computer Science & Systems Engineering, AUCE(A), Andhra University, Visakhapatnam, India

[1]micgirmay@gmail.com

[2] lalithabhaskari@yahoo.co.in

Abstract — Data compression is becoming more essential because the world has entered into the big IT and IOT technology which resulted in a big data universe. Information hiding and secured data communication are also serous issues in this big data word. Cryptography, Steganography and Watermarking are exemplary techniques broadly used to secure and hide information. In this research paper the focus will be to deal on implementing the SCIT, to compress digital images. Specifically the message to embed, as a watermark on a cover data is compressed lossless and then embedded. Then the embedded message is extracted and decompressed which reveal and match exactly to the initial message. By lossless compressing the secret message, it is possible to increase the payload (capacity), fidelity, robustness, and security requirements of digital watermarking. Issues related to compression and watermarking are covered in this work. The SCIT algorithm can be combined with other data compression methods to optimize the compression efficiency. The performance of this proposed algorithm is measured as compared to other related compression algorithms, and it is found that in some cases it can perform better than others.

Keywords — *Cover, Compression, Info-table, LSB, Message, Watermarking, Steganography, Transform*

I. INTRODUCTION

The issues of security, privacy and data encryption and data compression have been gained high attention as the world is becoming more digital and overwhelmed with big data generation and transmission on a highly fast communication network channels. As the data generated is huge in volume which requires voluminous digital hardware storage, data compression is used as a first step of solution to minimize the cost wasted for storage devices and speed up data processing operations as well. Several compression and decompression methods have been developed and implemented with their best and week features. Lossless compression is preferred basically for text and other critical media files such medical images. Lossy compression can be suitably applied with images and videos among others.

A number of techniques have been practiced deeply to deal with digital data security and privacy. Cryptography, steganography, and watermarking are a few well known techniques for data security and information hiding.

As detailed in [1] and other literature texts and articles, Steganography is the art of writing or embedding and hiding secret information in a cover data so that no one apart from the sender and intended recipient can uncover and utilize that hidden information. Here the most important thing is the hidden data not the cover data. Watermarking on the other hand is an art of embedding or inserting a sort of message; visibly, semi visibly or invisibly in a cover data which serves as a logo.

The main task of this research work is to exclusively addressing on lossless compression algorithms, by modeling and implementing a new transform technique similar to MTF method and then encoding using Huffman coding as a final step. That is to test the proposed transform method with images. Here using the proposed compression technique it is to be applied on watermark image before embedding it to a cover image.

The outline of this paper is as follows. Section I contains the introduction part of this paper. The basic concepts of compression, and algorithms required for this work are covered in sections II.A to II.D. A brief review of information hiding focusing on watermarking is covered in section II.E. Section III contains methodology within which modeling and coding is covered. Section IV contains proposed technique and its implementation. The experimental result is discussed in section V, and section VI presents the conclusion and summery of this paper work.

II. RELATED WORK

The research works done in the encoding/decoding or compression/decompression area is wide, and is still continuing in several fields as the data generated is vastly growing at high rate. In this chapter some basic theoretical and practical concepts and approaches of compression, privacy

and copyright assurance via watermarking and related techniques are reviewed.

A. Encoding and Data Compression

The process of data compression deals with techniques for reducing the storage required for storing data, or the bandwidth required to transmit it [2]. That is compression is the process of reducing or eliminating redundant and/or irrelevant data.

The methods of compression can be lossless or lossy. Lossless data compression methods can be classified as Entropy type, Dictionary type, and other types [3]. Lossy compression is useful when compressing pictures (e.g., JPEG) or audio files (e.g., MP3), where small deviations from the original are invisible (or inaudible) to the human senses. Decompression methods are processed in a reverse fashion of that compression steps and actions.

1) Text Compression: Text compression should be lossless, i.e., 100% reversible, otherwise the outcome leads to great loss of information. Most text compression methods are either statistical or dictionary based [7], [10]. Statistical method consists of methods that develop statistical models of the text. The model can be static or dynamic (adaptive). A static model uses fixed probabilities, whereas a dynamic model modifies the probabilities on the fly while text is being input and compressed. Most models are based on one of two approaches: Frequency and Context. In context-based model, the modeller considers the context of a symbol when assigning it a probability.

Some context-based text compression methods perform a transformation on the input data and then apply a statistical model to assign probabilities to the transformed symbols.

Run length encoding (RLE) is an example of transform and/or compression technique that works well on documents that exhibit to contain repetitive symbols. Burrows–Wheeler transform (BWT), move-to-front (MTF), PPM, and Differencing are some examples of predictive/transform techniques used to rearrange characters or symbols within a document after which it is possible to more reduce the size of the document. Encoding methods such as Huffman coding, Arithmetic coding, Elias gamma, Shannon-Fano, Tunstall and other more methods are those used to encode frequent symbols with fewer bits.

2) Image Compression: Compression is applicable for all types of formats of images (binary images, gray scale images, color images, computer graphics). In colored images, the number of bits required to represent a colored pixel is typically three to four times greater that the number of bits used to represent of gray scale images. For example RGB color image is an MxNx3 array of color pixels-thus the data that are the object of compression are the components of each color pixel (e.g., the Red, the Green and the Blue components of the pixels). If each color in an RGB pixel is represented by 8-bits then each pixel in RGB image is represented by 24-bits.

As interestingly explained in [2], image compression is achieved by the removal of one or more of three basic redundancies:

- Coding redundancy - some or most of the bits that represent the code word in images may not too much important.
- Spatial and Temporal redundancy – there is inter-pixel redundancy; means information is unnecessarily replicated in the representations of the correlated pixels.
- Irrelevant information – which regards with psycho-visual redundancy; that is images contain information that is ignored by the human visual system and/or extraneous to the intended use of the image.

Some examples of modeling and coding techniques, used for image compression include, from lossless, RLE, M-T-F, BWT, Huffman coding, predictive coding, TIFF, PNG standards; and from lossy or sub-lossy, JPEG standards (JPEG, JPEG-LS, JPEG-2000), and many others. Coding techniques such as Huffman and predictive (mapping) operate directly on the pixels of an image that is code redundancy and spatial & temporal redundancy respectively, and are classified under the spatial domain methods, while the JPEG (Joint Photographic Experts Group) compression standard is working on modifying the transform of an image and is classified under the transform coding methods which is based on discrete cosine transform (DCT) and/or discrete Fourier transform (DFT) image processing options.

PNG, TIFF, and GIF uses lossless LZW (Lempel–Ziv–Welch) compression algorithm among other options such as RLE, and Huffman encodings. JPEG-2000 uses wavelet –based coding [2].

B. Huffman Coding

Huffman coding is categorized under the entropy coders. An entropy coder is a method that assigns to every symbol from the alphabet a code depending on the probability of symbol occurrence [5]. The symbols that are more probable to occur get shorter codes than the less probable ones. Huffman coding is an optimal coding technique for compression purpose and serves as the basis for several popular programs run on various platforms [3]. The Huffman method is somewhat similar to the Shannon-Fano method.

Though the size of the code assigned to a symbol a_i basically depends on its probability of occurrences p_i it is also affected by the number of unique symbols (the size of the alphabet) in the stream or document. A small alphabet requires just a few codes, so they can all be short; a large alphabet requires many codes, so some must be long. Thus

one critical issue in Huffman coding is that, if those unique symbols (whether they are few or more) that constitute a given data file, do appear with equal probabilities (or frequency of occurrence), it may not considerably change the size of the document. For example, if there are n unique symbols and n=2m, where m is a common code length (e.g. for ASCII code, m=7, n=128) supported by the system when generating each alphanumeric characters and symbols, applying Huffman coding will not affect the size original of the document. Thus in such related cases some transform or predictive techniques are required either to:

- Reduce the number of unique symbols, or
- Vary the frequency of the representative symbols, or
- Rearrange their position in the document so that produce a repetitive form.

What it means is, the Huffman method cannot assign to any symbol a code shorter than one bit, if some form of predictive transform is not done ahead to using Huffman coding. Therefore a need comes to apply a multistep compression process, the last step being Huffman or other statistical coding methods such us Arithmetic, Elias, Golomb, and more others.

Huffman coding is useful in digital image and video compression standards as well. As the elements in digital image are dots or pixels, applying Huffman encoding technique on an image assumes the probability occurrence of the pixel intensities.

Huffman Decoding: During compressing some basic information must be collected in the form of Information Table (Info-table). These points (info) may include: list of the unique symbols and their probability (or frequency); and if required, their variable-size codeword, the size of the original and compressed file. Then the Huffman decoder must first construct (map) the Huffman tree as was constructed by the encoder. Only then it can read and decode the compressed stream.

C. Move-to-Front Transform

Move-to-Front (MTF) is an adaptive scheme and works on the principle that the appearance of a symbol or pixels in the input data makes that symbol more likely to appear in the near future. As detailed in [7] the basic idea of this method is to maintain the alphabet A of symbols or pixels as a list where frequently-occurring symbols are located near the front. A symbol s is encoded as the number of symbols that precede it in this list. The move-to-front method is locally adaptive, since it adapts itself to the frequencies of symbols in local areas of the input stream. The main idea is to move to front the symbols that mostly occur, so those symbols will have smaller output number or results in to more repetitiveness and thus will be coded with short codeword bits. Example, as the input document is processed, each symbol (or word, when used at world level) is looked up in the alphabet list and if it happens to occur as the i^{th} entry, it is coded by the index number i. Then the symbol is moved to the front of the list so that if it occurs soon afterwards, it will be coded by a number smaller than i.

After a stream is transformed using MTF, then follows encoding it using either of the variable length entropy encoders such as Golomb, Huffman, arithmetic, etc.

There are several possible variants to this method:-

- *Move-ahead-k* -The element of A matched by the current symbol is moved ahead k positions instead of all the way to the front of A.
- *Wait-c-and-move* - An element of A is moved to the front only after it has been matched c times to symbols from the input stream (not necessarily c consecutive times).

A critical issue in using MTF is - it gives poor performance if request sequence is in reverse order of the alphabet list [8], [14]. That is if the distribution of the symbols in the stream appears in reverse order at seemingly equal interval, then using MTF will not achieve good compression performance.

D. Entropy

In information theory an entropy encoding is a lossless data compression scheme that is independent of the specific characteristics of the medium. One of the main types of entropy coding creates and assigns a unique prefix-free code to each unique symbol that occurs in the input. Thus Entropy is a theoretical measure of quantity of information [5]–[7], [9]. Theoretically, as demonstrated by Shannon, for a set of possible events with known probabilities, p_1, p_2, p_3, …, p_n, that sum to 1, the Entropy of these events is given as

$$E = -\sum_{i=1}^{n} pi \log_2 pi$$

From this it is clear that more likely symbols or messages having greater probabilities contain less information.

E. Information Hiding and Data Security

In this big data word the information generated using IT and smart devices is huge and structured or unstructured data, which is actively transmitted to or shared among millions of users using internet network or disk devices. Outlaw data beneficiaries, i.e., unauthorized data copiers, and hackers are the obvious threats for privacy of legal user's data and information. Thus, security and privacy of potentially sensitive data, information and IT infrastructure as a whole is always a challenging concern in all level and type of IT and related enterprise systems. Privacy is considered a purely legal issue. Security is the process of actions to make practical the privacy laws and legislations planned to protect user's data and information.

Some of the basic elements of security are availability, utility, integrity, authenticity, and confidentiality. Information or data hiding deals embedding some sort of message or code which protects data from illegally direct use or modification by other parts.

In image processing, as researched and reviewed in [11] and many other articles, the techniques used for hiding information are mainly based on steganography, Cryptography and watermarking.

The core issues considered while hiding information are:

- ***Robustness:*** - The level of immunity against all forms of manipulation, obstruction and hackings or attacks.
- ***Fidelity:*** The degree of perceptual degradation due to embedding operation.
- ***Payload:*** The amount of message signal that can be reliably embedded to a given host signal and then extracted from (subject to perceptual constraints at the designated level of robustness).
- ***Security:*** how securely is communicated, which takes into consideration the issues of cryptographically encryption, authentication and authorization rules and techniques.

1) Steganography and Digital Watermarking:

a) Steganography: As detailed in [1] an important sub-discipline of information hiding is steganography usually interpreted to mean hiding information in other information. The word steganography is of Greek origin and means "covered writing" or "concealed writing". Steganography masks the sensitive data in any cover media (host signal) like images, audio, video, text, over the communicating channels, example internet websites.

b) Digital Watermarking: Watermarks are usually used to verify and protect or preserve the copyrights [12], [13] from unauthorized copying and distribution of digital content. That is the host data are protected by inserting an additional, usually, small data, namely, 'watermark' into the original data. This watermark identifies the host data and makes it unique.

Digital watermarking is classified based on several concerns: example, it is classified as, Visible or Invisible based on perceptivity; Transform domain or Spatial domain based on domain insertion; Robust or Fragile based on type; public or private considering method of detection; blind [15] or in-blind based on target recovery.

Blind watermark recovery needs the original cover media for detection. Semi-blind need some information from the insertion, but not the whole cover media. If the watermark is private then needs the original cover data later for the purpose of extraction and detection. Otherwise if the watermark is public the original document it is not required for the aim of detection.

Digital watermarking is broadly applied to digital images. The requirement of digital image watermarking concerns with variety of applications which includes copyright protection, tamper detection, distribution control, authentication, ownership assertion, privacy annotation, forensics, fingerprinting, web content filtering. Medical field and internet crimes can also benefit from digital image watermarking.

Researches done on digital image watermarking have shown many proposed techniques in the spatial domain or transform domain. With respect to the spatial or pixel domain watermarking, among the broadly used methods are modification of Least Significant bits (LSB), and spread spectrum modulation (SSM) [1].

In this paper it is focused on images as cover media. Fig.1 and 2, shows the generalized structure and steps activated in watermark embedding and watermark extracting operations, respectively. The security key can be generated using any appropriate encryption methods when required. The output of the Embedder and Extractor in watermarking process can be formulated as shown on the figures, respectively as:

Embedder: $d = E_K(d, m)$

Extractor: $R = D_K(d)$, and $D_K(d, d, m) = \{0, 1\}$

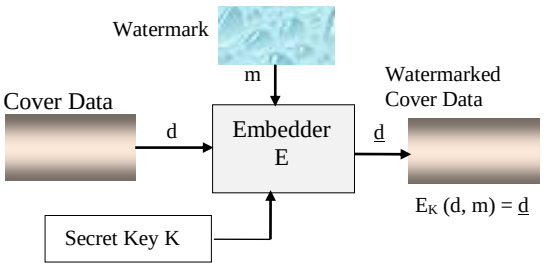

Fig.1. A generalized simple watermarking model

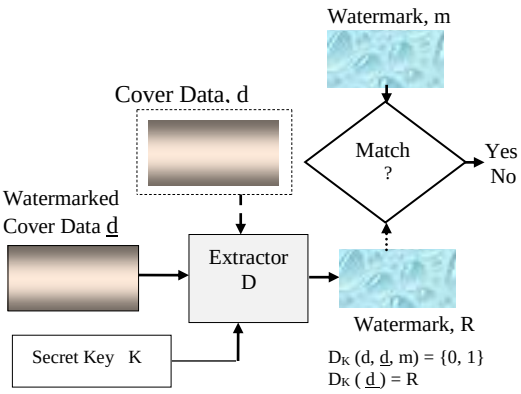

Fig. 2. Generalized simple watermark extractor model

F. Data Embedding Using LSB Method

The spatial domain based digital watermarking method enables simple form of data embedding by directly manipulating the LSBs of the cover-image,

though due to the possible attacks LSB embedding is relatively insecure, at least in its primitive form [1], [16].

A LSB refers to the last or the right-most bit in a binary number. The noise error sensed due to distortion of this single LSB is the minimal error that can be recorded and which may not serous with regard to the selected application. The advantageous of using LSB technique for data hiding is, it achieves invisibility and thus the perceivability of the cover image is preserved as if it is original, because a bit or few bits those having low weight of pixel intensity are replaced.

LSB implies embedding a message signal into the list significant bit(s) of the cover media (image, video, etc.). Logical or arithmetic operations can be used, or simply replace that bit and then extracting it later.

In this paper, LSB based watermark embedding is implemented in a sequential approach.

III. METHODOLOGY

A. *Modeling and Coding*

The task of finding a suitable model for text and image compression is an extremely important problem. As shown in Fig.3, some basic steps are conducted to reach the final compression step. The input message can be transformed first ahead to applying prediction. The separation into modeller and encoder is valuable because modeling and coding are very different sorts of activity [4]. The modeller determines the probability of the unique symbols in the case of text compression and supplies to the encoder. Once prediction is applied, the encoder receives the predicted probability or frequency of occurrences together with the actual input stream and turns them into sequence of bits (binary digits) to be transmitted or saved.

There are three ways that the encoder and the decoder can maintain the same model: static, semi-adaptive, adaptive modeling. In general in statistical compression process the stream can be first transformed so by doing that the size of the stream and/or the number of unique symbols can be minimized or their frequency can be more skewed as is observed in RLE, BWT, MTF, etc.

This paper work is also focusing on the transforming (reshuffling) part of the statistical encoding of digital image. Here the digital water mark is specifically to be compressed before embedding it to the cover digital image. For this to happen, first we have to convert the watermark image (colored or gray scale) into binary form of document. Then the digital (bytes) textual document obtained from the message digital image is to be rearranged (transformed) using the proposed SCIT technique, and then it is to be encoded (compressed) using Huffman coding method.

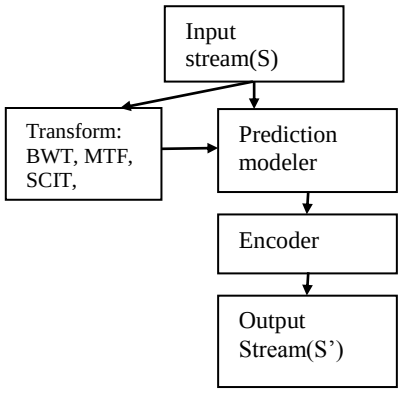

Fig.3. Outline of Basic Steps in Lossless Compression

The reason why is required to apply compression to the watermark is to increase the requirement of payload or capacity in digital image watermarking. Usually the message to be embedded should be smaller than the cover image, especially in visible watermarking, in order to avoid distortion of the main image. The watermarking and extraction processes are diagrammatically shown in Fig.4 and 5 respectively.

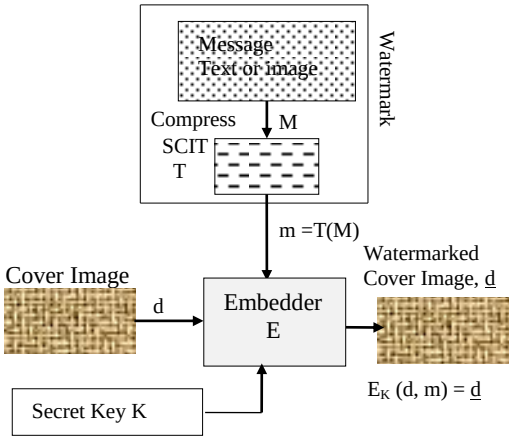

Fig. 4. A proposed watermarking model

Steps: Watermark Embedding using LSB

1. Choose the intended cover image and watermark of m-D array size.
2. Convert the watermark image into binary digits form of document.
3. Transform and compress the binary document form of the message (watermark) image.
4. Organize the compressed version of the watermark in n-D array and format it as that of the cover image
5. Embed the outcome of step 4 into the intended cover image using LSB method.
6. Save or communicate the watermarked image.

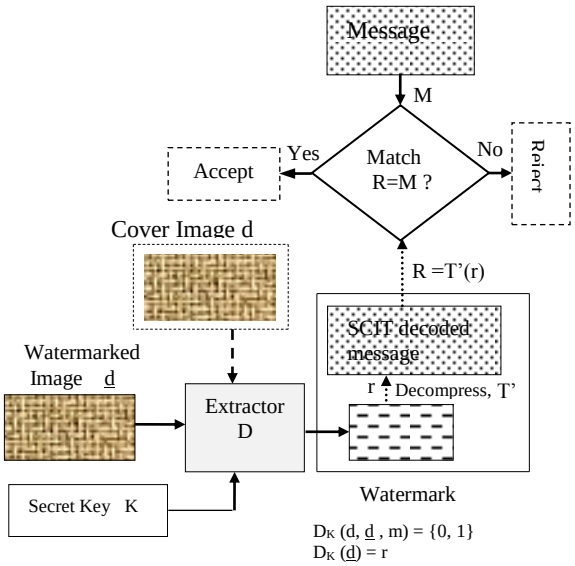

Fig. 5. A proposed watermark extractor model

Steps: Watermark Extraction

1. Get some information if required (e.g., blind watermark) from cover image or watermark.
2. Extract the watermark from the watermarked image.
3. Convert it to binary digits form of document and decompress it.
4. Organize it in m-D array of pixel intensity and convert it to similar image format of the initial watermark.
5. Compare for their exactness or similarity; the recovered watermark with the initial watermark before compressing and embedding.

IV. PROPOSED METHOD

If the probabilities of the symbols or pixel intensities in a document are more skewed, then applying statistical coding methods directly can achieve good results.

The proposed method, Simulated Circular Indexing Transform (SCIT) [14], aims at filling some specific gab observed in the compression or data transforming techniques reviewed above (RLE, MTF, and Huffman) and others, for example BWT. The approach is similar to the Move-to-Front transforming techniques. But instead of moving the symbol to any location in the list of alphabets, is just to play with index , moving or stepping next or back circularly if a different symbol is coming otherwise don't move if the current symbol happens to occur repetitively. Conceptually all the moves (next/back or up/down) are mapped or simulated circularly to the position index value of the symbols in the alphabet, except that don't move or 'stay there' case is given a 0 index value for a repetitive symbol.

In move-to-front and some other transforming methods, the transformed data is represented by the indexes (numerical values). In the proposed SCIT method, the index is translated to the symbol or pixel currently indexed, thus the transformed data is represented by the digit or pixel value themselves though it can also be represented by the corresponding indexes as it is usually done. As this translated (virtual) indexing works right, for comparison and compatibility purpose, the MTF is also implemented using this concept in this paper. For example take a row scan line of an input image with pixel intensity value of 8-bits:

I: 12 33 45 24 56 56 75 99 85 85 45 24 33 56

Here, I=14 pixels and the unique pixel elements are n=8 which to be listed and saved in the info-table for later use in the reconstructing and decompression operation:

P0, Ia →

0	1	2	3	4	5	6	7
12	33	45	24	56	75	99	85

Info-table basically contains unique symbols or pixels in I, and may include their codeword and some additional information, such as frequencies of each unique symbol.

Let us use these variables: n, Ia, Pv, P0. Where,

n - total number of unique pixels in I.

Ia - holds the index value of a next symbol/pixel to be accessed, in the info-table.

Pv - virtual(simulated) index, in terms of numeric or the pixel intensity themselves

P0 -actual index value of immediate previously accessed symbol/pixel.

Now, let's formulate the proposed method as follows. Initially Ia, Pv, and P0 are assigned to a specific value, example 0 index values.

Steps: During encoding*:* For each byte or pixel in the image find corresponding Pv as follows. Let, initially, P0=Ia=0.

1. Scan the next pixel 'p' from the image, I.
2. Ia← index of 'P' as is in the Info-Table,
   if Ia ≥ P0, then Pv = Ia – P0,
   else, Pv = (n + Ia) - P0
   P0 = (Pv + P0)mod(n)
3. Save the value Pv in sequence as a transformed list of pixels, I', and repeat from step (1) until the intended pixels are finished.
4. Compress I' using Huffman or other encoding methods.
5. Embed as a watermarking into a proposed cover image.

Example,

I : 12 33 45 24 56 56 75 99 85 85 45 24 33 56 = 14(8)

Ia → 0 1 2 3 4 4 5 6 7 7 2 3 1 4

Pv: 0 1 1 1 1 0 1 1 1 0 3 1 6 3 = 14(4)

P0: 0 1 2 3 4 4 5 6 7 7 2 3 1 4

Or, Pv is represented by the name of the pixel value themselves instead of using the index values.

Pv: 12 33 33 33 33 12 33 33 33 12 24 33 99 24 = 14(4)

The successive list of Pv gives the transformed variant of that I. The value 0 in the Pv list is to mean don't move, that means take the current pixel, rather. Since, a pointer at starting time, in most cases, points to the first byte/word in an input stream or document, the first 0 in Pv tells to consider the first pixel intensity in I which in this illustration is, pixel value 12.

Now, as a last step is to, code Pv list (transformed image, I') using either Huffman or other statistical coding techniques.

Note that the number inside the parenthesis, e.g., 8 in 14(8), and 4 in 14(4) represent the number of unique symbols/pixels in the original and transformed images respectively.

Steps: During decoding: Huffman or other coded image is decoded or decompressed to obtain I' back. Then retransform Pv list from I', to get back I, as follows.

1. Scan the next pv value from I'; and translate to corresponding numeric (index) value in the Info-Table.
2. Determine the corresponding actual index, P0 = (Pv + P0)mod(n), Ia←P0
3. Copy the pixel/symbol at location P0 or Ia in the info-table, I← Info-Table[Ia].

Example: (if the second optional list of Pv is taken),
Pv: 12 33 33 33 33 12 33 33 33 12 24 33 99 24 = 14(4)

Pv: 0 1 1 …

P0: 0 1 2 …

Ia→
I: 12 33 45 24 56 56 75 99 85 85 45 24 33 56 = 14(8)

For the above transforming and retransforming example, Move-to-Front, will do as follows.
Ia →
I : 12 33 45 24 56 56 75 99 85 85 45 24 33 56 = 14(8)
Pv: 0 1 2 3 4 0 5 6 7 0 5 5 6 6 = 14(8)

i.e., using the symbols or pixels instead,

Pv: 12 33 45 24 56 12 75 99 85 12 75 75 99 99 = 14(8)

It is clear that the number of unique symbols/pixels after transforming using the SCIT method is 4 while after using MTF is 8, same as in the original document (image) for this arbitrary example.

As is observed in the info-table, Table 1, the transformed string may have less number of unique symbols (Pxl.) as compared to the original string. This phenomenon holds true also for MTF in other examples. That is some of the symbols will gain highest frequency (Freq.) though in some cases, the SCIT technique propagates and equalizes the frequency for many pixels in which case the performance will decrease.

Now, if a statistical encoding technique, Huffman coding, for example, is applied further, then,
I → 41 bits ≈ 6 bytes/14 ; if Huffman is solely used.
MTF: I'→ 40 bits = 5 bytes/14 ; if MTF→Huffman multistep is used.
SCIT: I'→ 23 bits ≈ 3 bytes/14 ; if SCIT→Huffman multistep is used.

TABLE 1. COMBINED INFO-TABLE OF HM, MTF AND SCIT

P0	Input		MTF		SCIT	
Ia	Pxl.	Freq.	Pxl.	Freq.	Pxl.	Freq.
0	12	1	12	**3**	12	3
1	33	2	33	1	33	**8**
2	45	2	45	1		
3	24	2	24	1	24	2
4	56	**3**	56	1		
5	75	1	75	**3**		
6	99	1	99	**3**	99	1
7	85	2	85	1		
	n=8, I =14		n=8, I' =14		n=4, I' =14	

V. EXPERIMENTAL RESULTS AND DISCUSSION

Table 2, summarizes the compression performance of the SCIT technique, comparing with respect to Huffman alone, and with MTF; that means, Huffman; MTF then Huffman; and SCIT then Huffman.

Commonly used benchmark test image files can be found from the net to evaluate lossless compression methods. Thus using some of this files and additional files, more than 60 images have been tested, out of which very few indicated that using either of MTF or SCIT is not effective. The rest showed improved compression result if either MTF or SCIT transforms and then Huffman coding is performed.

It is demonstrated that the SCIT techniques work better for some special arrangement or distribution of symbols or pixel intensities in particular files.

For this multistep compression process, C program has been implemented for the compression task and MATLAB for the watermarking purpose.

Table 3 shows compression ratio (CR) and compression factor (CF) for each the three techniques.

$$CR = \frac{Size\ of\ the\ compressed\ \ stream}{Size\ of\ the\ input\ \ stream}$$

$$CF = \frac{1}{CR}$$

TABLE 2. IMAGE COMPRESSION RESULTS AND COMPARISON

Color Image	Size, Bytes	After compression, in Bytes		
		Huffman	MTF	SCIT
Lena	786,432	764,865	675,355	**671,081**
Pepers	547,344	509,841	390,483	**385,402**
Img49	1,125,000	1,113,759	999,352	**874,907**
Img60	1,125,000	1,054,180	960,315	**939,179**
Img26	1,125,000	1,103,558	**882,214**	901,763
Img65	1,000,500	930,167	**781,285**	808,629
Img25	1,000,500	**835,268**	883,791	947,431
Img56	421,500	**398,621**	403,343	417,634

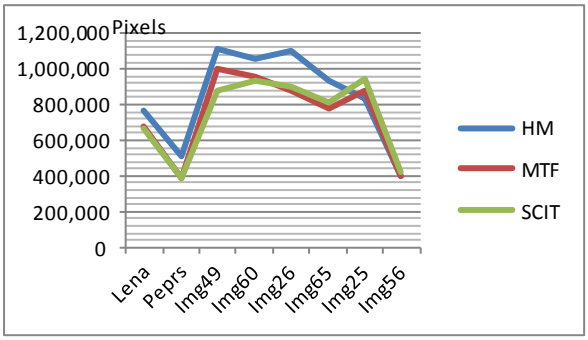

Fig. 6. Comparing Compression results graphically

TABLE 3. COMPRESSION PERFORMANCE MEASURING METRICS

CR			*CF*		
HM	***MTF***	***SCIT***	***HM***	***MTF***	***SCIT***
0.97	0.86	**0.85**	1.03	1.16	**1.17**
0.93	0.71	**0.70**	1.07	1.40	**1.42**
0.99	0.89	**0.78**	1.01	1.13	**1.29**
0.94	0.85	**0.83**	1.07	1.17	**1.20**
0.98	**0.78**	0.80	1.02	**1.28**	1.25
0.93	**0.78**	0.81	1.08	**1.28l**	1.24
0.83	0.88	0.95	**1.20**	1.13	1.07
0.95	0.97	0.99	**1.06**	1.05	1.01

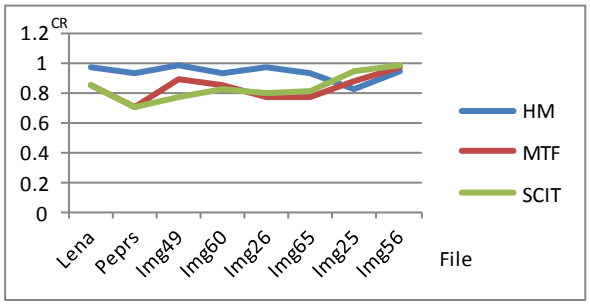

Fig.7. Performance Comparisons based on Compression Ratio (CR).

From Table 2 and 3, it is clear that the highlighted results of compression and the performance for the addressed methods, suggest that all or most compression algorithms have their own positive and weak sides although there are superior techniques, of course.

Fig. 6 and 7 shows charts that demonstrate graphically the compression performance of the Huffman (HM), MTF, and SCIT methods.

TABLE 4. INPUT MESSAGE AND COVER DATA DETAILS

Images	**Size, bytes**	**SCIT→Huffman encoding**
W.mark	*170x154x3 = 78,540*	*70,722 ≅153x155x3=71,145*
Cover	*512x512x3 = 786,432*	

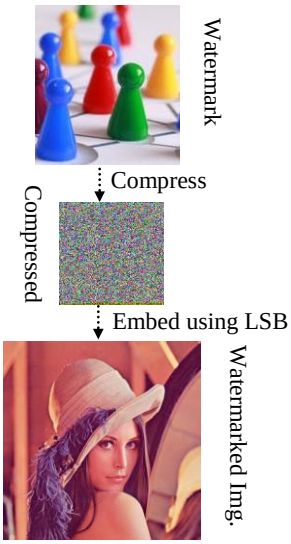

Figure 8. Screenshot of watermarking process

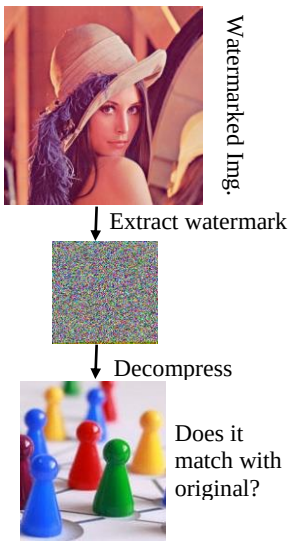

Fig. 9. Screenshot of Watermark Extraction Process

SCIT is preferable for streams that repeat periodically or sub-periodically in a reverse fashion. That is, the SCIT method performs best for media

files that are composed in reversed and equally well distributed manner at several intervals.

Table 4, contains information regarding digital watermarking. As stated above, the message is first transformed using SCIT and then encoded using Huffman code. By doing so, it is observed that high payload and embedding capacity are achieved. Here only the message to be embedded is compressed. If only least single bit (LSB) is to be used, then a total of 170x154x3=78,540 pixels cannot fully inserted into a cover data whose size is less than or equal to this value or otherwise it has to be compressed prior to embedding it. So, in this example, it is safe to embed the compressed version of the message i.e., 71,145 pixels into 512x512x3=786,432 pixels cover data. The outcomes for watermark embedding and extraction processes are shown in Fig.8 and 9. In this work, the main concentration is on data compression rather than security aspects. Visually there is no difference between the original message and the extracted one as observed from the experimental values which is given in the figures showed. The same is quantitatively depicted in Table 4, after extracting the watermark.

VI. CONCLUSION

As data is generated at large volume and data transfer rate is becoming essential in this digital and IOT world, some means of mechanism is required to solve the storage, latency, speed, bandwidth, security and energy problems. Data compression is becoming an integral part of the modern information storage and retrieval systems and researches on data compression methods are still continuing as there are gaps to be covered. The need for data hiding is even more essential than compression. Watermarking is one example of data hiding mechanism which mainly useful to protect data from illegal use. In this paper, using the proposed SCIT technique, it has been tried to fill some gaps identified in data compression transforming techniques for some special cases at byte/character/pixel level and implemented in image watermarking as well. As a conclusion, future works can be carried on an efficient and optimal coding technique (though difficult to avoid the multistep coding techniques approach) for all file types (structured, unstructured) so that it can solve one or all of the Big Data issues (speed, volume, variety and security).

REFERENCES

[1] Stefan Katzenbeisser, Fabien A.P. Petitcolas, "Information Hiding Techniques for Steganography and Digital Watermarking", Artech House publishers, Boston.London, pp. 1-149, 2000.

[2] Rafael C.Gonzalez, Rechard E. Woods,"Digital Image processing, 3rd Edition", Pearson Prentice Hall, USA, pp.548-563, 2008.

[3] David Salmon, "Data compression, The Complete Reference, 4th Edition", Springer, Verlag London Limited, pp.74-89, 2007.

[4] Timothy C. Bell, John G. Cleary, Ian H. Witten, "Text Compression", Prentice Hall, Englewood cliffs new jersey, pp.12-22, 1990.

[5] Sebastian Deorowicz, "Universal lossless data compression algorithms", Doctor of Philosophy Dissertation, pp.12-15, 2003.

[6] Colton McAnlis, Aleks Haecky, "Understanding Compression, Data Compression for Modern Developers", O'Reilly Media, USA, pp.19-21, 2016.

[7] David Salomon, Giovanni Motta, "Handbook of Data Compression, 5th Edition", Springer, Verlag London Limited, pp.45-48, 2010.

[8] Rakesh Mohanty, Sasmita Tripathy, "An Improved Move-To-Front (IMTF) Off-line Algorithm for the List Accessing Problem", RsearchGate ,2011.

[9] Khalid Sayood, "Introduction to Data Compression, 3rd Edition", Elsevier, USA, pp.16-17, 2006.

[10] Arup Kumar Bhattacharjee, Tanumon Bej, Saheb Agarwal, "Comparison Study of Lossless Data Compression Algorithms for Text Data", IOSR Journal of Computer Engineering (IOSR-JCE), Vol.11, Issue 6, pp.16-19, 2013.

[11] Aditi Kurapa, Mahendra Sahare, Umesh Lilhore, "A Robust Fractal Code and LSB based Image Watermarking", International Journal of Computer Applications (IJCA), vol.160, No.9, 2017.

[12] Kapil Kumar Kaswan, Dr. Roshan Lal, "Data Protection Using Digital Watermarking", International Journal Of Engineering And Computer Science(ijecs), vol.2, Issue 12, pp.3405-3410, 2013.

[13] Mehmet Utku Celik, Gaurav Sharma, A. Murat Tekalp, "Lossless Watermarking for Image Authentication: A New Framework and an Implementation", IEEE TRANSACTIONS ON IMAGE PROCESSING, vol.15, no.4,pp.1042-1049, 2006

[14] G. Gebremichael Girmay, D. Lalitha Bhaskari, "Data Compression Using Simulated Circular Indexing Transform(SCIT)", International Journal of Computer Application(IJCA), vol.179, no.43, pp.1-9, 2018.

[15] Stefanos Zafeiriou, Anastasios Tefas, Ioannis Pitas, "Blind robust watermarking schemes for copyright protection of 3D mesh objects", IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 11, NO. 5, pp.596-607, 2005.

[16] Sherin Sugathan, "An improved LSB embedding technique for image steganography", 2nd International Conference, IEEE 2016.