An improved symbol reduction technique based Huffman coder for efficient entropy coding in the transform coders

Vikrant Singh Thakur | Kavita Thakur | Shubhrata Gupta

1 Department of Electrical Engineering, National Institute of Technology, Raipur, India
2 S.O.S. in Electronics and Photonic, Pt. Ravishankar Shukla University, Raipur, India

Abstract

Entropy coding is the essential block of transform coders that losslessly converts the quantized transform coefficients into the bit-stream suitable for transmission or storage. Usually, the entropy coders exhibit less compression capability than the lossy coding techniques. Hence, in the past decade, several efforts have been made to improve the compression capability of the entropy coding technique. Recently, a symbol reduction technique (SRT) based Huffman coder is developed to achieve higher compression than the existing entropy coders at similar complexity of the regular Huffman coder. However, the SRT-based Huffman coding is not popular for the real-time applications due to the improper negative symbol handling and the additional indexing issues, which restrict its compression gain at most 10–20% over the regular Huffman coder. Hence, in this paper, an improved SRT (ISRT) based Huffman coding technique is proposed to properly alleviate the deficiencies of the recent SRT-based Huffman coder and to achieve higher compression gains. The proposed entropy coder is extensively evaluated on the ground of compression gain and the time complexity. The results show that the proposed ISRT-based Huffman coder provides significant compression gain against the existing entropy coders with lower time consumptions.

1 INTRODUCTION

Entropy coding is one of the essential steps in the transform coders, which maps the quantized transform coefficients into the compressed bit stream [1]. Principally, the entropy coder is a lossless coder and therefore provides very less compression of input data as compared to the lossy compression techniques [2, 3]. Hence, improved entropy coding techniques have been developed from time to time to achieve higher compression gain. The modern entropy coding is mostly dominated by the popular Huffman coding [4, 5], Golomb coding [6–9], Arithmetic coding [10–12] and their advance variations such as context-adaptive variable length coding (CAVLC) [9–12] and context-adaptive binary arithmetic coding (CABAC) [13–16]. By nature, all these existing entropy coders are constrained between the coding efficiency and the computational complexity. The popular entropy coders such as Huffman coder and arithmetic coder are less efficient but highly useful due to their low complexity. While the advance entropy coders like CAVLC and CABAC are more efficient but highly complex in nature. Hence, some improved entropy coders are essentially required that could provide higher encoding efficiency with low complexity. Recently, one important attempt has been made by Saravanan et al. [17] and Bheshaj et al. [18] in the development of a symbol reduction technique (SRT) based Huffman coding for the JPEG image coder. In this entropy coding technique, the input symbols are grouped prior to the Huffman coding to achieve a higher compression ratio as compared to the regular Huffman coding with a slight increase in the complexity. This technique has opened a new direction to achieve better entropy coding efficiency than the existing entropy coders but restricted in the practical applications due to the inefficiency of handling the negative symbols of source sequence and the symbol digit indexing problems. Hence, in this paper, an improved SRT (ISRT) based Huffman coding technique is proposed to efficiently alleviate the negative symbol handling and symbol digit indexing problems of the existing SRT-based Huffman coding technique, and to achieve higher compression ratios with...
2 | ENTROPY CODING: DEFINITION AND FUNDAMENTALS

In a very basic sense, the term entropy coding refers to the process of lossless mapping of an input symbol into a codeword usually in the binary stream. If the codeword generated from the entropy coder is fixed in length for all the input symbols then the entropy coder is known as fixed length or fixed rate entropy coder. Whereas, if the generated codeword is variable in length for all the input symbols then the entropy coder is termed as the variable length of variable rate entropy coder. The entropy decoding is the exact inverse process of the entropy encoding, which restores the original source symbol from the encoded binary codeword. Now, we first define the basic mathematical preliminaries that are essential for analysing the fundamentals of the entropy coding or also known as source coding.

Let \( X = x_1, x_2, x_3, \ldots, x_n \) be a sequence of input symbols, where each symbol takes values from a finite universe of alphabet \( A = a_1, a_2, a_3, \ldots, a_m \). Let \( p_{ij} \) denotes the probability that \( x_i \), which is the \( i^{th} \) symbol in the source sequence will take the value \( a_j \). Then the self-information \( I_i \) linked with the \( i^{th} \) symbol can be defined as,

\[
I_i = \log_2 \frac{1}{p_{ij}} = -\log_2(p_{ij}) \text{ (bits)}
\]  

Next, we can also define the entropy of the sequence as,

\[
H(X) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}\log_2 p_{ij}
\]  

According to Shannon’s theorem for source coding [19], the entropy of the source sequence is the lower bound, which indicates the average number of bits needed to encode the source symbols without loss of information [20]. Hence, the efficiency of entropy coders is always measured in terms of closeness to the source entropy. Now, the average number of bits \( L_{avg} \) required to represent a symbol is defined as,

\[
L_{avg} = \sum_{k=1}^{L} f(r_k) p_r(r_k)
\]  

where, \( r_k \) is the discrete random variable or symbol for \( k = 1, 2, 3 \ldots L \) with associated probabilities \( p_r(r_k) \), and \( f(r_k) \) is the number of bits used to represent each value of \( r_k \). The total number of bits required to represent a complete symbol sequence is then calculated by the total number of symbols multiplied by \( L_{avg} [20] \).

Furthermore, usually, the existing entropy coders are compared in the view of gain in the compression and reduction in the time complexity. The parameter time complexity is simply a measure of the total time required to encode and decode the input source sequence. Whereas, the compression gain of entropy coders signify the increase in compression ratio (CR), which is given by,

\[
\text{Compression ratio (CR)} = \frac{B_i \text{ (in bytes)}}{B_c \text{ (in bytes)}}
\]  

where, \( B_i \) and \( B_c \) are the size of the input and compressed data, respectively [21]. For a good entropy coder, the compression gain (CR) should be higher and the time complexity must be as minimum as possible.

3 | REGULAR HUFFMAN CODING TECHNIQUE

Huffman coding is a popular variable length entropy coding technique, which usually assigns shorter codeword to a higher probable symbol and longer codewords to the lower probable symbols. The Huffman coding technique constructs a binary tree for assigning binary codewords to the source symbols. Every source symbol from the sequence becomes a leaf of the Huffman tree. The corresponding codeword for a source symbol is obtained by moving in a unique path leading from the root of the tree to that of the respective leaf node, with each edge is labelled by 0 or 1, that are added to the respective codeword for the occurrence of the edge along the path based on whether the left or right child of a given node occurs next along the path. For the proper understanding of the Huffman algorithm, let us consider a source sequence \( X \) that contains \( n \) symbols with the respective probabilities \( p_{ij} \) as described in the previous section, then the basic steps of the Huffman coding technique are as follows:

**Step 1:** Initially, sort the probabilities \( p_{ij} \) of the source symbols in descending order, and represent the corresponding symbols as a leaf node of the Huffman tree.

**Step 2:** Next, merge the two lowest probable symbols; usually, they are the last two symbols in the sorted list, to make a new combined node and calculate its probability by adding the respective probabilities of the two merged symbols. Now, assign a “1” to the top branch and a “0” to the bottom branch. However, this branch labelling can be interchanged but the final codeword generated is unique in both the ways.
Step 3: Now, repeat the procedure of step 2, until a single node is left with the respective probability of 1. This node forms the root node of the designed Huffman tree.

Step 4: Finally, obtained the respective codewords for each of the source symbols by sequentially reading the branch labels or digits from the root node to the corresponding terminal node.

After encoding all of the source symbols, the Huffman coding technique generates a Huffman table for the reconstruction of the original symbols. The generated Huffman table must be available to the decoder side for the proper reconstruction of the source symbols. In order to reconstruct the original source sequence $X$, the Huffman decoder simply takes inverse steps as the codeword generation process and remaps the code stream to the individual symbols by sequentially moving in the developed Huffman tree node to node from a leaf node to the respective root node. The Huffman coding technique usually offers good coding efficiency; however, it is not able to provide the fractional bitrates as the other entropy coders such as arithmetic, CA VLC and CABAC coders. Hence, in order to improve the coding efficiency of the regular Huffman coding technique, the SRT-based Huffman coders have been proposed in past few years [17, 18]. A detailed description of the recent SRT-based Huffman coding technique along with its advantages and practical limitations has been given in the following section.

4 | EXISTING SRT-BASED HUFFMAN CODING TECHNIQUE

Usually, in the entropy encoding, the number of symbols presented in a source sequence plays a crucial role in achieving a good compression ratio. If the numbers of source symbols are decreased by any possible mapping process prior to the entropy encoding, then the compression ratio immediately increases. This is the basic concept behind the existing SRT-based entropy coders. In 2009, Saravanan et al. [17] proposed the first SRT-based Huffman coding technique to attain higher compression ratios as compared to the simple Huffman coding for the JPEG image compression standard. This entropy coder groups the input symbols in a group of four symbols prior to the Huffman coding and hence provide 10–20% gain in the compression ratio. However, this technique is not able to handle the negative symbols of the source sequence and also requires additional indexing to enlist the digits for each grouped symbols. The index list generated after encoding must be transmitted to the receiver for the proper decoding, which reduces the compression ratio depending on the size of the index list. Later on, in 2012, Bheshaj et al. [18] proposed a modified SRT (MSRT) based Huffman coder to address the negative symbol handling issue of the previous SRT-based Huffman coding technique proposed by Saravanan et al. [17]. The basic algorithm of this recent MSRT-based Huffman coding technique is given in Algorithm 1.

Algorithm 1. MSRT based Huffman coding technique proposed by Bheshaj et al. [18].

Start:

(i) The recent MSRT-based Huffman coding technique adds 128 to the negative symbols to shift them towards the positive side, which is not a good strategy to handle the negative symbols. Because, the negative symbols could have higher values than the 128, hence the direct addition of 128 to the negative symbols may generate negative symbols with lower values but they still present in the sequence of the source symbol.

(ii) The second and most important problem related to the recent MSRT-based Huffman coding technique is that the two indices index and indexdigit that are required to transmit along with the encoded bit-stream to the receiver for the proper decoding of the source symbols. This additional data transmission greatly reduces the compression ratio of the encoder, and hence this technique offers 20–25% compression gain over the regular Huffman coding, which is a lower gain in the compression ratio than the expectation.

5 | PROPOSED ISRT-BASED HUFFMAN CODING TECHNIQUE

The practical limitations of the recent MSRT-based Huffman coding technique discussed in the previous section; restrict it
for the real-time entropy coding applications, and hence leads to a strong motivation for the development of some improved SRT-based entropy coding techniques. Therefore, to properly alleviate the issues related to the existing SRT-based entropy coders, this section presents a new ISRT-based Huffman coding technique. The important contribution of the proposed ISRT-based Huffman coding technique is that it can efficiently handle the negative symbols exist in the source sequence and does not require additional indexing to enlist the digits for each of the grouped symbols. The encoding process of the proposed ISRT-based Huffman coding technique is given in Algorithm 2. The decoding process of the proposed ISRT-based Huffman coding technique is given in Algorithm 3, which is the exact opposite of the encoding procedure shown in Algorithm 2.

Algorithm 2. Encoding process of the proposed ISRT based Huffman coding technique.

Start: 

\( X \rightarrow \) Sequence of source symbols.

\( X_{\text{abs}} = \text{abs}(\text{min}(X)) \); Represents absolute value of maximum negative symbol presented in the source sequence.

\( X_{\text{min}} = X + X_{\text{abs}} \); Shift the entire source sequence in the positive integer side.

\( NSG = 2 \); Number of source symbols to be grouped.

\( X_{\text{binary}} = \text{dec2bin}(X_{\text{abs}},8) \); Convert each source symbol into 8-bit binary number.

\( j = 1 \); Counter for grouped symbols.

\( \text{for} \ i = 1 : \text{NSG} : \text{length}(X_{\text{binary}}) \) 

\( \text{temp} = [X_{\text{binary}}(1:8) X_{\text{binary}}(i+1:8)] \); Group NSG symbols in binary domain by concatenation of the symbols.

\( GX(j) = \text{bin2dec}(\text{temp}) \); Convert the grouped symbol from binary to a positive integer.

\( j = j + 1 \); Increment the counter for grouped symbols.

\( \text{end for} \)

\( \text{Hu f f} = \text{Hu f fman} \_\text{Coding}(GX) \); Encode the grouped symbol sequence \( GX \) using Huffman coding technique described in section 3.

End of Algorithm:

The advantages of the proposed ISRT-based Huffman coding technique over the exciting MSRT-based Huffman coder are as follows:

(i) The first advantage of the proposed ISRT-based Huffman coding technique is that, in order to handle the negative symbols of the source sequence, the whole sequence is shifted to the positive integer side by the addition of minimum value (i.e. maximum value of negative symbol) of the source sequence to all the symbols of the sequence. Shifting of the whole sequence towards the positive side ensures that all the new symbols are positive and it also reduces the requirement of indexing for the negative symbols. Hence, the proposed ISRT-based entropy coding technique properly alleviates the negative symbol handling and its indexing problem related to the MSRT-based entropy coding technique.

(ii) The second advantage of the proposed entropy coder is that it does not require additional indexing to indicate the digits for each grouped symbols. Because, the proposed entropy coder, groups the source symbols in the binary domain with 8-bit representation for each symbol, hence at the decoder, when the combined 16-bit symbol is divided into two parts, the initially combined symbols can be easily isolated.

(iii) The third advantage of the proposed entropy coder is that, to reduce the computation complexity, it groups two source symbols instead of four symbols as used in the MSRT-based entropy coder [18].

Furthermore, to properly elaborate the encoding and decoding processes of the proposed ISRT-based Huffman coding technique and its advantages over the existing MSRT-based Huffman coding technique, we now take a real-time practical example of entropy coding and decoding for a random symbol sequence \( X \) defined in Equation (5).

\[
X = [-130 \ 5 \ 10 \ -135 \ 8 \ 15 \ 0 \ 2]
\] (5)

Algorithm 3. Decoding process of the proposed ISRT based Huffman coding technique.

Start:

\( X_{\text{abs}} \leftarrow \) Minimum value of the original source sequence transmitted by the proposed ISRT based Huffman encoder (1 ~ 2 bytes of size).

\( NSG = 2 \); Number of source symbols to be grouped.

\( Rec \_GX \rightarrow \) Huffman_Decoding(Huffman_Code); Decode the blocks of grouped symbol sequence from the encoded Huffman code Huffman_Code received.

\( Rec \_GX_{\text{binary}} = \text{dec2bin}(X_{\text{abs}}, 16) \); Convert each grouped symbol into 16-bit binary number.

\( nn = 1 \); Counter for original symbols separation or extraction from the grouped symbols.

\( \text{for} \ i = 1 : \text{length}(Rec \_GX_{\text{binary}}) \) 

\( \hat{X}_{\text{abs}}(1, nn) = \text{bin2dec}(Rec \_GX_{\text{binary}}(i, 1:8)) \); Extract first source symbol from the \( i^{th} \) grouped symbol and convert it into a decimal number.

\( \hat{X}_{\text{abs}}(1, nn + 1) = \text{bin2dec}(Rec \_GX_{\text{binary}}(i, 9:16)) \); Extract second source symbol from the \( i^{th} \) grouped symbol and convert it into a decimal number.

\( nn = nn + NSG \); Increment the counter.

\( \text{end for} \)

\( X = \hat{X}_{\text{dec}} - X_{\text{abs}} \); Restore the shifted source sequence in the original integer values and obtain the original source sequence.

End of Algorithm:

The matrix \( X \) shown in Equation (5) is the input to the entropy encoding problem that contains a total of eight symbols to be encoded and has a positive maximum value of 15 and a negative maximum value of -135. For this input matrix \( X \), we can easily analyse the improper negative value handling problem of the existing MSRT-based Huffman coding technique. According to the existing MSRT-based Huffman coding technique (Algorithm 1, line 8), a predefined number 128 has to be added in the vector \( X \) to shift the entire vector to the positive direction. Applying, line 8 of the MSRT algorithm to the input
vector $X$ the resultant vector $X_{\text{MSRT}}$ is shown in Equation (6).

$$X_{\text{Shift}} = [-2 133 138 -7 136 143 128 130] \quad (6)$$

From Equation (6), we can easily observe that there are two negative elements -2 and -7 still present in the shifted vector $X_{\text{Shift}}$, that is the addition process of 128 to the input sequence is not able to completely shift the entire vector to the positive side. These two negative values -2 and -7, lead to the requirements of two indexes e.g. index and indexdigit (Algorithm 1, line numbers 2, and 3) when encoded and decoded using MSRT-based Huffman coding technique. Hence, the two important problems discussed in Section 4 related to the existing MSRT-based Huffman coding technique are clearly visible in the practical example considered in this section.

Now, we explain the step by step encoding and decoding processes of the input symbol sequence vector $X$ using the proposed ISRT-based Huffman coding technique and also validate its advantages over the existing MSRT-based Huffman coding technique.

### 5.1 Entropy encoding of source sequence X using proposed ISRT-based Huffman coding technique

Starting from the first line of the proposed entropy encoding technique shown in Algorithm 2, let us store the source sequence in a row vector $X$. Now in the second line, the absolute minimum value of the source symbols has been obtained using MATLAB syntax $X_{\text{min}} = \text{abs}(\min(X))$, which comes out as $X_{\text{min}} = 135$. The next task in the proposed entropy encoder is to shift the entire source sequence $X$ in positive direction. To perform this task we follow the MATLAB syntax $X_{\text{new}} = X + X_{\text{min}}$ shown in line number three of Algorithm 2, which yields the value of the resultant shifted vector $X_{\text{new}}$ as shown in Equation (7).

$$X_{\text{New}} = [5 140 145 0 143 150 135 137] \quad (7)$$

From Equation (7), we can easily analyse that the proposed technique properly shifts the entire source vector $X$ in the positive direction by adding the value of $X_{\text{min}}$ in the source vector $X$. Next, line number four of Algorithm 2 indicates that the proposed entropy coding technique combines NSG = 2 symbols during the symbol grouping process. Moving further, a MATLAB syntax $X_{\text{newbinary}} = \text{dec2bin}(X_{\text{new}}, 8)$ shown in line number five of Algorithm 2, converts each source symbol into an 8-bit binary number and stores it to a new vector named as $X_{\text{newbinary}}$. Following this syntax we obtain a character set vector $X_{\text{newbinary}}$ of size $8 \times 8$, as shown in Equation (8) that contains 8-bit binary equivalent for each of the symbols presented in vector $X_{\text{new}}$.

$$X_{\text{newbinary}} = \begin{bmatrix} 00000101 & 00011000 & 10001100 & 10010001 & 10010110 & 10001111 & 10001111 & 10001001 \\ 10010001 & 10011010 & 10011011 & 10011010 & 10011011 & 10011011 & 10011010 & 10011011 \end{bmatrix}$$

Next, line number six in Algorithm 2, initializes a counter variable $j$ by 1 to count the grouped symbols during the grouping process of source symbols in binary domain using the obtained vector $X_{\text{newbinary}}$ in the previous step.

Further, as discussed in the previous section that the proposed entropy coder, groups the two adjacent source symbols in the binary domain by the concatenation process. In this context, from line number seven to line number 11 in Algorithm 2, performs a loop for 1 to the complete length of vector $X_{\text{newbinary}}$. For the above example, for loop performs the looping operation from 1 to 8. Inside this for loop (in line 8), a temporary variable temp has been calculated in each loop that contains the binary value of a grouped symbol which is obtained by concatenation of the two adjacent binary values presented in vector $X_{\text{newbinary}}$. Next in line 9, the obtained grouped symbol in each loop is converted into an equivalent decimal number and then saved in a new vector $GX$, which is actually the grouped symbol vector whose length is simply half of the length of source vector $X$. After the execution of each loop counter, variable $j$ is incremented by 1 as shown in line 10 of Algorithm 2. Finally, after completion of for loop, the obtained grouped symbol vector $GX$ is shown in Equation (9).

$$GX = [1420 37120 36758 34697] \quad (9)$$

Vector $GX$ shown in Equation (9) is the final grouped symbol source vector to be encoded by the regular Huffman coding technique to get binary codeword vector Huff_Code as described in line number 10 of Algorithm 2. After regular Huffman coding, the codewords obtained for each of the symbols of vector $GX$ is shown in Table 1.

With the generation of codeword for each of the symbols presented in vector $GX$, the encoding process of the proposed ISRT-based Huffman coding technique is now finished and the number of bits required for the transmission is 8 bits. However, as mentioned in Algorithm 2, we have to transmit one additional information that is the value of $X_{\text{min}}$ (here in the above example $X_{\text{min}} = 135$) along with the obtained codewords for the proper decoding process. This additional information consumes 8-bit binary overload for the transmission. So the total number of bits to be transmitted after encoding using the proposed entropy
In this section, we will present the decoding process of the encoded codeword Huff_Code obtained in the previous section using the proposed entropy decoder described in Algorithm 3.

Let us first store the minimum value (i.e. maximum negative value) of the source vector \( X \) transmitted by the proposed encoder in a vector \( X_{\text{min}} \) as shown in line 1 of Algorithm 3. Here in the case of the above example the received value of \( X_{\text{min}} = 135 \). Moving to the next line, unlike line 4 of Algorithm 2, in line 2 of Algorithm 3, the constant NSG = 2 here in Algorithm 3 represents the number of symbols in which each grouped symbols to be divided during the decoding process. Next in line 3, the received codeword Huff_Code is decoded using the regular Huffman decoding and stored in vector Rec_GX. The decoded vector Rec_GX is shown in Equation (10).

\[
\text{Rec}_G\text{X} = \begin{bmatrix} 1420 & 37120 & 36758 & 34697 \end{bmatrix}
\]

Equation (10)

Now in the next part of the proposed decoding process is the conversion of Rec_GX vector from a decimal number to binary number to perform the ungrouping process as shown in line 4 of Algorithm 3. The \( \text{Rec}_G\text{X}_{\text{binary}} \) vector obtained after execution of line number 4 is shown in Equation (11).

\[
\text{Rec}_G\text{X}_{\text{binary}} = \\
0000010110001100 \text{ for 1st element 1420} \\
1001000100000000 \text{ for 2nd element 37120} \\
1000111110010110 \text{ for 3rd element 36758} \\
1000011110001001 \text{ for 4th element 34697}
\]

Equation (11)

Next, in line number 5 of Algorithm 3, a new constant \( nn \) is initialized by one to count the number of ungrouped symbols. Moving further, line number 6 to 10 contains a for loop which runs for variable \( i = 1 \) to the length of vector \( \text{Rec}_G\text{X}_{\text{binary}} \) (four times for the above example). In \( i^{\text{th}} \) iteration of the for loop, the first eight binary digits from the \( i^{\text{th}} \) symbol of \( \text{Rec}_G\text{X}_{\text{binary}} \) vector are extracted and their decimal equivalent is stored in \( nn^{\text{th}} \) position of vector \( \hat{x}_{\text{dec}} \) using line number 7. Thereafter the remaining eight to sixteen binary digits are then extracted from the same \( i^{\text{th}} \) symbol of the vector \( \text{Rec}_G\text{X}_{\text{binary}} \) and its decimal equivalent is stored in \( nn + 1^{\text{st}} \) position of vector \( \hat{x}_{\text{dec}} \) using line number 8. After completion of each loop, the counter variable \( nn \) is incremented by NSG to identify the remaining ungrouped symbols as shown in line number 9. On the completion of the for loop, we get the vector \( \hat{x}_{\text{dec}} \) which is the shifted version of original ungrouped source symbols as shown in Equation (12).

\[
\hat{x}_{\text{dec}} = [-2 133 138 -7 136 143 128 130]
\]

Equation (12)

Finally, in order to get the original source symbols show in Equation (5), we have to shift the \( \hat{x}_{\text{dec}} \) vector by utilizing the value of \( X_{\text{min}} \) transmitted by the encoder using line number 11 of Algorithm 3. The obtained value of finally decoded vector \( X \) is shown in Equation (13).

\[
X = \begin{bmatrix} -130 & 5 & 10 & -135 & 8 & 15 & 0 & 2 \end{bmatrix}
\]

Equation (13)

The final decoded vector \( X \) is same as the original input source symbol vector shown in Equation (5). Consequently, we can conclude that the proposed ISRT-based Huffman coding technique fairly able to encode and decode the practical source symbols, and hence highly feasible for the real time coding applications.

6 | PERFORMANCE EVALUATION OF THE PROPOSED ISRT-BASED HUFFMAN CODING TECHNIQUE

This section presents an extensive evaluation for the entropy coding performance of the proposed ISRT-based Huffman coding technique against the existing entropy coders. In the first part of this section, a comparative entropy coding performance of the proposed entropy coder is presented with a developed test system commonly used in the video encoders. The next part of this section presents a comparative real-time practical entropy coding performance of the proposed entropy coder using JPEG image coder used in the Apple iPhone 4/4S models.

In order to present fair testing of the entropy coding performances for the proposed entropy coder and the other existing entropy coders, we need a common testing platform that suits the encoding process of all the important entropy coding techniques. Hence, in the first part of this section, a proper test system has been developed that comprises DCT transformation, scalar quantization, and different entropy coding techniques as shown in Figure 1. Note that the developed test system is commonly used for image and video compression; hence the testing

| S. no. | Source symbol | Codeword |
|-------|---------------|----------|
| 1     | 1420          | 01       |
| 2     | 37120         | 00       |
| 3     | 36758         | 11       |
| 4     | 34697         | 10       |

Number of bits needed to transmit 8

TABLE 1 Codewords obtained for each source symbol of vector GX using the proposed ISRT-based Huffman coding technique
of entropy coders with this test system leads us towards their realistic performance.

The performance of the proposed entropy coder has been compared with the recent MSRT-based Huffman coder, regular Huffman coding technique, arithmetic coding, and the advanced entropy coders such as Exp-Golomb coding, CA VLC, and CABAC coders. Particularly, since the advanced entropy coders like Exp-Golomb coding, CA VLC, and CABAC coders support block DCT transformation of size $4 \times 4$, hence the test system shown in Figure 1, has been designed to satisfy that requirement. Further, the encoding performances of all these entropy coders have been analysed using the developed test system for the four standard greyscale test images of different sizes. The name of test images with their specifications are enlisted in Table 2 and all the test images are also shown in Figure 2. The reason behind the utilization of test images with different sizes is to properly analyse the compression gain and the time complexity of the proposed entropy coder on different levels of input complexities.

Moreover, for the realistic performance validation of the tested entropy coders, all the experiments have been performed in MATLAB (version: R2014b) software platform and Intel(R) Core (TM), i5, 3.60 GHz CPU-based computer system. After compression and reconstruction of all the four test images shown in Figure 2, with the developed test system on different quantization factors (QFs), the obtained values of compression ratio and the PSNR index for all the tested entropy coders are tabulated from Tables 3 to 6 respectively.

Note that, the bold highlighted values in all the tables represent the highest compression ratio on the respective compression or quality levels. Meanwhile, in order to properly analyse the compression gain of the proposed entropy coder against the existing entropy coders, the compression ratios tabulated in Tables 3–6 are also plotted with respect to the reconstruction qualities i.e. PSNR values. The compression ratio versus PSNR plots for the four test images are shown from Figures 3 to 6.

The compression ratio vs PSNR analysis is an important feature that represents the ability of entropy coders to provide compression gain on the same reconstruction quality.

From Tables 3–6, it is clearly evident that, on lower values of QF (i.e. on lower compression side), Huffman and arithmetic coders provide better compression ratios than the advanced entropy coders Exp-Golomb, CA VLC, and CABAC. However,
TABLE 3  Compression ratio and PSNR values obtained from all the entropy coders for the first test image Birds.png

| Quantization factor (QF) | PSNR in dB | Regular Huffman coding | MSRT-based Huffman coding | Arithmetic coding | Exp-Golomb coding | CAVLC coding | CABAC coding | Proposed ISRT-based Huffman coding |
|-------------------------|------------|------------------------|---------------------------|------------------|------------------|-------------|-------------|-----------------------------------|
| 10                      | 41.963     | 3.713                  | 4.567                     | 5.100            | 3.542            | 2.756       | 3.197       | 10.709                            |
| 50                      | 31.860     | 3.983                  | 4.899                     | 7.287            | 5.220            | 5.634       | 6.535       | 14.401                            |
| 100                     | 27.476     | 4.002                  | 4.923                     | 7.773            | 5.860            | 9.451       | 10.963      | 15.413                            |
| 150                     | 24.843     | 4.105                  | 5.049                     | 7.919            | 6.232            | 12.731      | 14.768      | 15.756                            |

TABLE 4  Compression ratio and PSNR values obtained from all the entropy coders for the second test image Lena.png

| Quantization factor (QF) | PSNR in dB | Regular Huffman coding | MSRT-based Huffman coding | Arithmetic coding | Exp-Golomb coding | CAVLC coding | CABAC coding | Proposed ISRT-based Huffman coding |
|-------------------------|------------|------------------------|---------------------------|------------------|------------------|-------------|-------------|-----------------------------------|
| 10                      | 40.046     | 3.679                  | 4.414                     | 4.725            | 3.418            | 2.611       | 2.872       | 9.554                             |
| 50                      | 31.508     | 3.998                  | 4.798                     | 7.365            | 5.173            | 5.308       | 5.839       | 14.614                            |
| 100                     | 27.700     | 4.005                  | 4.806                     | 7.841            | 5.800            | 8.798       | 9.678       | 15.521                            |
| 150                     | 24.900     | 4.247                  | 5.696                     | 7.958            | 6.139            | 12.095      | 13.305      | 15.798                            |

On the higher values of QF (i.e. on higher compression side), the advance entropy coders offer significant gains in the compression ratios as compared to Huffman and arithmetic coders. Moreover, from the resultant tables, it is also observable that the recent MSRT-based Huffman coder provides higher compression ratios than the regular Huffman coding on all the values of QF, for all the four test images, with the average gain of about 22.5% in the compression ratio. But the gain in compression ratio is very less as the expectation from the recent MSRT-based Huffman coder, because of the

TABLE 5  Compression ratio and PSNR values obtained from all the entropy coders for third test image Pentagon_city.tiff

| Quantization factor (QF) | PSNR in dB | Regular Huffman coding | MSRT-based Huffman coding | Arithmetic coding | Exp-Golomb coding | CAVLC coding | CABAC coding | Proposed ISRT-based Huffman coding |
|-------------------------|------------|------------------------|---------------------------|------------------|------------------|-------------|-------------|-----------------------------------|
| 10                      | 38.947     | 3.272                  | 3.927                     | 3.403            | 2.731            | 1.963       | 2.219       | 6.908                             |
| 50                      | 29.172     | 3.995                  | 4.794                     | 6.932            | 4.972            | 4.593       | 5.190       | 13.810                            |
| 100                     | 25.907     | 4.000                  | 4.800                     | 7.788            | 5.705            | 8.053       | 9.100       | 15.441                            |
| 150                     | 23.833     | 4.000                  | 4.800                     | 7.975            | 5.984            | 11.004      | 12.435      | 15.837                            |

TABLE 6  Compression ratio and PSNR values obtained from all the entropy coders for the fourth test image Brikwall.tiff

| Quantization factor (QF) | PSNR in dB | Regular Huffman coding | MSRT-based Huffman coding | Arithmetic coding | Exp-Golomb coding | CAVLC coding | CABAC coding | Proposed ISRT-based Huffman coding |
|-------------------------|------------|------------------------|---------------------------|------------------|------------------|-------------|-------------|-----------------------------------|
| 10                      | 38.947     | 3.272                  | 3.927                     | 3.403            | 2.731            | 1.963       | 2.219       | 6.908                             |
| 50                      | 29.172     | 3.995                  | 4.794                     | 6.932            | 4.972            | 4.593       | 5.190       | 13.810                            |
| 100                     | 25.907     | 4.000                  | 4.800                     | 7.788            | 5.705            | 8.053       | 9.100       | 15.441                            |
| 150                     | 23.833     | 4.000                  | 4.800                     | 7.975            | 5.984            | 11.004      | 12.435      | 15.837                            |
requirements to transmit the additional indexing lists with the encoded bit stream, which reduces the compression ratio from the expected values. Meanwhile, from the resultant tables, we can also observe that the proposed ISRT-based Huffman coder provides highest compression ratios on all the QF values for all the four test images compared to the existing entropy coders. Furthermore, analyse the plots of compression ratios shown from Figures 3–6, which provides a good visualization about the comparative compression performances of all the tested entropy coders on different reconstruction qualities. The plots clearly validate that the proposed ISRT-based Huffman coding technique provides higher compression as compared to the state-of-the-art entropy coders over the same reconstruction qualities for all the four test images.

Furthermore, in order to provide the quantitative performance gain of the proposed entropy coder on different
THAKUR ET AL.

FIGURE 6  Plot of compression ratio (CR) against the PSNR index for the four test image Brikwall.tiff

TABLE 7  Average gains in the compression ratio of the proposed ISRT-based Huffman coder against the existing entropy coders

| S. no. | Input image size | Regular Huffman coding | MSRT-based Huffman coding | Arithmetic coding | Exp-Golomb coding | CAVLC coding | CABAC coding |
|-------|-----------------|------------------------|---------------------------|-------------------|-------------------|--------------|--------------|
| 1     | 256 × 256       | 254.73                 | 188.40                    | 101.21            | 173.49            | 132.74       | 100.63       |
| 2     | 512 × 512       | 246.19                 | 188.49                    | 99.27             | 171.73            | 137.06       | 115.50       |
| 3     | 1024 × 1024     | 234.76                 | 180.61                    | 99.11             | 172.15            | 158.58       | 130.82       |

complexity levels of the input signal; its average percentage gains in the compression ratio against the existing entropy coders have been calculated according to the size of input images. The obtained values of average percentage gains in the compression ratio are tabulated in Table 7.

From Table 7, it can be easily observed that the proposed entropy coder offers high compression gain against the existing entropy coders irrespective of the complexity level of the input signal. These significant gains in the compression ratio confirm the higher reliability of the proposed ISRT-based Huffman coder for the practical entropy coding applications. Apart from this, in order to properly justify the high gains in the compression ratio obtained from the proposed ISRT-based Huffman coding technique as compared to the existing entropy coders; we now present an example based comparative description for the working of the proposed ISRT-based Huffman coding against the regular Huffman coding technique.

Let us consider a 4 × 4 block of pixel value as a source matrix \( X \) for the entropy coding problem given as:

\[
X = \begin{bmatrix}
125 & 30 & 25 & 255 \\
100 & 10 & 75 & 178 \\
255 & 128 & 5 & 70 \\
55 & 200 & 245 & 2
\end{bmatrix}
\]  

The source matrix \( X \) is actually one of the 4 × 4 size blocks extracted from the first test image Birds.png. Next according to Figure 1 the source matrix \( X \) is first DCT transformed and the transformed matrix is given in Equation (15).

\[
Y = \begin{bmatrix}
439.5000 & 12.2346 & 80.5000 & -1.8206 \\
-34.7383 & -129.5196 & 233.0301 & -39.7127 \\
29.0000 & -73.3612 & -112 & 7.1158 \\
21.9658 & 116.7873 & 96.1416 & -55.9804
\end{bmatrix}
\]  

Next, the DCT transformed matrix \( Y \) is then quantized with \( QF = 10 \) and rounded along with the sign of symbols. The quantized and rounded matrix is given in Equation (16).

\[
YQR = \begin{bmatrix}
44 & 1 & 8 & 0 \\
-3 & -13 & 23 & -4 \\
3 & -7 & -11 & 1 \\
2 & 12 & 10 & -6
\end{bmatrix}
\]  

Matrix \( YQR \) is the final symbol source matrix for entropy coding that has 16 source symbols. Let us first encode the \( YQR \) matrix using the regular Huffman coding technique. The regular Huffman coder generates the codeword for each of the symbols as shown in Table 8.

Further, in order to encode the source symbols using the proposed ISRT-based Huffman coder, the matrix \( YQR \) given in Equation (16) has been first reshaped in a column matrix as given by:

\[
YQRR = \begin{bmatrix}
44 & -3 & 3 & 2 & 1 & -13 & -7 & 12 & 8 & 23 & -11 & 10 & 0 & -4 & 1 & -6
\end{bmatrix}
\]
TABLE 8  Codewords for each source symbol of matrix YQR obtained from the regular Huffman coding technique

| S. no. | Source symbol (column wise) | Codeword generated by regular Huffman coder |
|--------|----------------------------|---------------------------------------------|
| 1      | 44                         | 0000                                        |
| 2      | −3                         | 0111                                        |
| 3      | 3                          | 0110                                        |
| 4      | 2                          | 0001                                        |
| 5      | 1                          | 010                                         |
| 6      | −13                        | 0011                                        |
| 7      | −7                         | 0010                                        |
| 8      | 12                         | 1101                                        |
| 9      | 8                          | 1100                                        |
| 10     | 23                         | 1111                                        |
| 11     | −11                        | 1110                                        |
| 12     | 10                         | 1001                                        |
| 13     | 0                          | 1000                                        |
| 14     | −4                         | 1011                                        |
| 15     | 1                          | 010                                         |
| 16     | −6                         | 1010                                        |
| Total bits needed to transmit |                           | 62                                           |
| Average codeword length $L_{avg}$ using Equation (3) |                           | 4.25                                         |
| Compression ratio (CR) |                           | $(16 \times 8) / 62 = 2.064$                |

Finally, as per the proposed encoding algorithm of ISRT-based Huffman coding given in Algorithm 2, the maximum negative value of the YQRR matrix (i.e. −13) has been added to all the elements of this matrix to obtain all the positive source symbols. The modified source symbol matrix YQRRM is given in Equation (18).

$$YQRR = \begin{bmatrix} 57 & 10 & 16 & 15 & 14 & 0 & 6 & 25 & 21 & 36 & 2 & 23 & 13 & 9 & 14 & 7 \end{bmatrix}$$

Lastly, the above-modified source symbol matrix that contains sixteen different positive symbols is reduced in eight symbols using Algorithm 2. The obtained combined source symbol matrix using the proposed ISRT-based Huffman encoder is given as,

$$Z = \begin{bmatrix} 14602 & 4111 & 3584 & 1561 & 5412 & 535 & 3337 & 3591 \end{bmatrix}$$

Finally, the codewords obtained from the proposed ISRT-based Huffman coder for each of the symbols of matrix $Z$ is shown in Table 9.

From Tables 8 and 9, we can easily analyse that the proposed ISRT-based Huffman coding technique offers a significant gain in compression ratio over the regular Huffman coder. The quantitative gain in the compression ratio can be simply calculated by the following equation.

$$\text{Gain in compression ratio (GCR)(in\%)} = \frac{\text{CR}_p - \text{CR}_r}{\text{CR}_r} \times 100$$

where, $\text{CR}_p$ is the compression ratio obtained from the proposed ISRT-based Huffman coder, and $\text{CR}_r$ is the compression ratio obtained from the regular Huffman coder.

From Equation (20) we can easily obtain the value of GCR for the proposed ISRT-based Huffman coder over the regular Huffman coding as equal to 158.38 %. This example can be easily extended to justify the respective high compression gains of the proposed entropy coder against the other existing entropy coders tabulated in Table 7.

6.1 Real-time practical performance evaluation of proposed entropy coder using the JPEG image coder used in Apple-iPhone 4/4S model

This subsection presents a detailed comparative real-time and practical entropy coding performance of the proposed entropy coder using JPEG image coder used in the Apple-iPhone 4/4S model [25]. The JPEG image coder is the most popular and widely used tool for practical image coding applications [26]. The basic structure of a JPEG image coder is shown in Figure 7, which contains four main functions for example:

1. The input image is first divided into non-overlapping blocks (NOBs) of size $8 \times 8$.
2. Decorrelation of image pixels from each of the NOBs by the forward DCT transformation,
3. Quantization of individual DCT coefficients from each NOBs to achieve compression,
4. Entropy coding of the quantized DCT coefficients using the Huffman coding technique to generate compressed bit-stream.

In the standard JPEG coder, a default quantization table (DQT) is used to generate the compressed stream at different bitrates using the scaling technique of the DQT. The DQT of the JPEG standard (DQT_{JPEG}) is shown in Equation (21).

\[
\text{DQT}_{\text{JPEG}} = \begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \\
\end{bmatrix}
\] (21)

However, in the past decade, several modifications for the quantization table of JPEG standard have been proposed and find utilization in the practical applications. For instance, JPEG coder used in the Apple-iPhone 4/4S model is almost the same as the standard JPEG coder except for the quantization table. The quantization table used in JPEG image coder of Apple-iPhone 4/4S model is shown in Equation (22).

\[
\text{QT}_{\text{Apple-iPhone}} = \begin{bmatrix}
1 & 1 & 1 & 1 & 2 & 5 & 6 & 5 \\
1 & 1 & 1 & 2 & 4 & 5 & 6 & 5 \\
1 & 1 & 2 & 2 & 5 & 8 & 8 & 6 \\
1 & 2 & 3 & 5 & 6 & 10 & 10 & 7 \\
2 & 3 & 5 & 6 & 8 & 10 & 11 & 9 \\
4 & 6 & 7 & 8 & 10 & 12 & 12 & 10 \\
7 & 9 & 9 & 9 & 11 & 10 & 10 & 9 \\
4 & 6 & 6 & 7 & 8 & 10 & 9 & 9 \\
\end{bmatrix}
\] (22)

Apart from the change in the quantization table, both the image coders: standard JPEG coder, and the JPEG coder of Apple-iPhone use the same regular Huffman coder for the entropy coding process. In order to evaluate the real-time performance of the proposed entropy coder for the practical applications, the regular Huffman coder of the Apple-iPhone JPEG coder has been replaced by the proposed entropy coder. Thereafter, the image compression performance in the form of PSNR characteristics for the standard JPEG coder, Apple-iPhone JPEG coder, and the Apple-iPhone JPEG coder with proposed entropy coder has been obtained for the two standard greyscale test images Birds.png and Lena.png on different bitrates. The resultant PSNR characteristics obtained after testing for the two test images Birds.png and Lena.png are shown in Figures 8 and 9, respectively.

In the PSNR plots of Figures 8 and 9, the resultant PSNR values obtained after compression and decompression process are plotted for different bitrates. The bitrate is an indicator of the compression amount usually defined in bits-per-pixel. The
The resultant PSNR characteristics obtained for Lena.png test image

FIGURE 9 Resultant PSNR characteristics obtained for Lena.png test image

higher value of bitrate signifies a lower compression amount (i.e. lower compression ratio), whereas the lower value bitrate signifies a higher compression amount.

From Figures 8 and 9, it is evident that the Apple-iPhone JPEG coder provides better compression than the standard JPEG coder for both the test images. Meanwhile, from both the figures it is clearly reflected that the compression quality of the Apple-iPhone JPEG coder with the proposed entropy coder is much better than the others because it offers higher PSNR values on all the bitrates. The PSNR performances shown in Figures 8 and 9 clearly validates the higher compression capability of the proposed ISRT-based Huffman coding technique over the regular Huffman coder for the real-time practical applications.

7  |  TIME COMPLEXITY ANALYSIS OF THE PROPOSED ISRT-BASED HUFFMAN CODING TECHNIQUE

The time complexity of an entropy coder is a very important parameter that actually reflects the feasibility of the coder for practical applications. In the case of entropy coding, the time complexity is a measure of total time consumption by the entropy coder for encoding and decoding of the source sequence. This section presents an extensive time complexity analysis for the proposed ISRT-based Huffman coding technique against the existing entropy coders to provide a clear visualization of its real-time application feasibility.

For the proper time complexity evaluation of entropy coders, the total time consumed by a particular coder for the encoding and decoding of the source sequence has been determined for all the four test images using the test system shown in Figure 1 over the different values of the quantization factor. Where-

upon, the respective average values of obtained time requirements for the different input image sizes have been taken for the comparison and analysis. The obtained values of total time requirements for encoding and decoding processes (TTRFEDP) for the proposed and the other existing entropy coders are tabulated in Table 10.

From Table 10, it is clearly observable that among all the tested entropy coders, the regular Huffman coder consumes the smallest average time of about 10.507 s for the encoding and decoding of all the four test images with the developed test system. This is the reason why in spite of the low coding efficiency the Huffman coder is still popular for practical entropy coding applications. On the other hand, the advance CABAC entropy coder takes the highest average time of about 53.077 s for the encoding and decoding of the four test images. The alternative of CABAC coder that is the CA VLC coder shows slightly lower time complexity than the CABAC but also consumes very high time than the other existing entropy coders. The entropy coders such as arithmetic coder and the Exp-Golomb coder show moderate time complexity for the practical implementations. However, they both require higher time to encode and decode the input sequence as compared to the regular Huffman coding.

Now moving to the recent MSRT-based Huffman coding, it consumes higher time of average 66.25% than the regular Huffman coding, but the time consumption is still less than the other existing entropy coders. Furthermore, it is worth to mention that the average time consumption for the proposed ISRT-based Huffman coder is about 14.60 s, which is only 35% higher than the regular Huffman coder. Otherwise, the time consumption of the proposed entropy coder is smallest than the recent MSRT-based Huffman coding and the other state-of-the-art entropy coders. Therefore, the proposed ISRT-based Huffman coder offers low complexity for coding and decoding, and hence highly suitable for real-time entropy coding applications.

8  |  CONCLUSIONS

In this paper, a new ISRT-based Huffman coder has been developed to achieve higher compression ratios with less complexity as compared to the recent and popular entropy coders. The developed entropy coder has been particularly designed to efficiently alleviate the negative symbol handling and additional indexing problems of the recent MSRT-based Huffman coding technique. Further, in order to provide the realistic validation for the coding performances of the entropy coders, a proper test system has been developed that satisfies all the structural requirements for testing of simple as well as advance entropy coders. The comparative compression gain and the time complexity of the proposed ISRT-based Huffman coder has been extensively analysed using the developed test system against the popular entropy coders. The obtained results validate that the developed ISRT-based Huffman coding technique provides
significant gains in the compression ratio with low complexity on the same reconstruction qualities as compared to the existing entropy coders. Hence, the developed entropy coder is the best alternative to replace the existing entropy coders to achieve better coding performance with the low complexity.

REFERENCES

1. Thakur, V.S., et al.: Hybrid WPT-BDCT transform for high-quality image compression. IET Image Proc. 11(10), 899–909 (2017)
2. Sayood, K.: Introduction to Data Compression. 4th ed., Morgan Kaufman, San Francisco, CA (2012)
3. Menasri, W., et al.: Field programmable gate array implementation of variable-bins high efficiency video coding CABAC decoder with path delay optimisation. IET Image Proc. 13(6), 954–963 (2019)
4. Huffman, D.A.: A method for the construction of minimum redundancy codes. Proc. IRE, 40, 1098–1101 (1952)
5. Yuan, S., Hu, J.: Research on image compression technology based on Huffman coding. J. Visual Commun. Image Represent. 59, 33–38 (2019)
6. Golomb, S.W.: Run-length encodings. IEEE Trans. Inf. Theory IT-12, 399–401 (1966)
7. Taubman, D., Marcelin, M.: JPEG2000 Image Compression Fundamentals, Standards and Practice. Springer, New York (2002)
8. Teuhola, J.: A compression method clustered bit vectors. Inf. Process. Lett. 7, 308–311 (1978)
9. Richardson, I.: The H.264 Advanced Video Coding Standard. 2nd ed., Wiley, Hoboken, NJ (2010)
10. Rissanen, J.: Generalized Kraft inequality and arithmetic Coding. IBM J. Res. Dev. 20, 198–203 (1976)
11. Paseo, R.: Source Coding: Algorithms for Fast Data Compression. Ph.D. dissertation, Stanford University (1976)
12. Masmoudi, A., Masmoudi, A.: A new arithmetic coding model for a block-based lossless image compression based on exploiting inter-block correlation. Signal Image Video Process. 9, 1021–1027 (2015)
13. ITU-T and ISO/IEC JTC 1, Advanced Video Coding for Generic Audio-Visual Services, (ICTV) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 ISO/IEC 23008-2 and ITU-T Recommendation H.265 (2013)
14. Saravanan, C., Ponalagusamy, R.: Lossless grey-scale image compression using source symbols reduction and huffman coding. Int. J. Image Process. 3(5), 246–251 (2009)
15. Sullivan, G., et al.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. 22(12), 1648–1667 (2012)
16. High Efficiency Video Coding (HEVC), Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JCT 1/SC 29/WG 11 ISO/IEC 23008-2 and ITU-T Recommendation H.265 (2013)
17. Shan, J.R., et al.: Comparison of the coding efficiency of video coding standards-including high efficiency video coding (HEVC). IEEE Trans. Circuits Syst. Video Technol. 22(12), 1669–1684 (2012)
18. Kumar, B., et al.: A new hybrid JPEG image compression scheme using symbol reduction technique. Int. J. Multimedia Appl. 4(3), 81–92 (2012)
19. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Technol. J. 27, 379–423 (1948)
20. Gonzalez, R.C., Woods, R.E.: Image compression. In: Digital Image Processing. 3rd ed., pp. 526–538. Prentice Hall, New Jersey (2012)
21. Jayaraman, S., et al.: Image Compression. In: Digital Image Processing. 1st ed., pp. 444–541. Tata McGraw Hill Education, New York (2009)
22. Clark, R.J.: Image Data Compression: An introduction. In: Transform Coding of Images. Academic Press, San Diego, CA (1985)
23. Rao, K.R., Yip, P.: Applications of DCT. In: Discrete Cosine Transform: Algorithms, Advantages, Applications. 1st ed., pp. 136–344. Academic Press, Cambridge, MA (1990)

24. Ahmed, N., et al.: Discrete cosine transform. IEEE Trans. Comput. 23(1), 90–93 (1974)

25. Sun, C., Yang, E.H.: An efficient DCT-based image compression system based on Laplacian transparent composite model. IEEE Trans. Image Process. 24(3), 886–900 (2015)

26. Wallace, G.K.: The JPEG still picture compression standard. Commun. ACM 34(4), 30–44 (1991)

How to cite this article: Thakur VS, Thakur K, Gupta S. An improved symbol reduction technique based Huffman coder for efficient entropy coding in the transform coders. IET Image Process. 2021;15:1008–1022. https://doi.org/10.1049/ipr2.12081