EFFECTIVE IMAGE COMPRESSION AND TRANSMISSION USING DCT AND HUFFMAN CODING

ADARSH KHARE, RAVI MOHAN
Department of EC, Shri Ram Institute of Technology (SRIT), Jabalpur (M.P)
Adarshk sare@gmail.com ravimohan7677@yahoo.com

ABSTRACT

Image Compression addresses the problem of reducing the amount of data required to represent the digital image. Compression is achieved by the removal of one or more of three basic data redundancies:

1. Coding redundancy, which is present when less than optimal (i.e. the smallest length) code words are used.
2. Inter pixel redundancy, which results from correlations between the pixels of an image.
3. Psycho visual redundancy which is due to data that is ignored by the human visual system (i.e. visually nonessential information).

Huffman codes contain the smallest possible number of code symbols (e.g., bits) per source symbol (e.g., grey level value) subject to the constraint that the source symbols are coded one at a time. So, Huffman coding remove coding redundancies, when combined with the technique of image compression using Discrete Cosine Transform (DCT) helps in compressing the image data to a very good extent.

For the efficient transmission of an image across a channel, source coding in the form of image compression at the transmitter side & the image recovery at the receiver side are the integral process involved in any digital communication system. Other processes like channel encoding, signal modulation at the transmitter side & their corresponding inverse processes at the receiver side along with the channel equalization help greatly in minimizing the bit error rate due to the effect of noise & bandwidth limitations (or the channel capacity) of the channel.

1. INTRODUCTION

Uncompressed graphics, audio and video data require considerable storage capacity and transmission bandwidth. Despite rapid progress in mass storage density, processor speeds and digital communication system performance demand for data storage capacity and data transmission bandwidth continues to out strip the capabilities of the available technologies. The recent growths of data intensive digital audio, image, and video based (multimedia) web applications, has sustained the need for more efficient ways. With the growth of technology and the entrance into the Digital Age, the world has found itself amid a vast amount of information. Dealing with such enormous amount of information can often present difficulties. Digital information must be stored and retrieved in an efficient manner in order to put it to practical use. Without some sort of compression, sorting, storing and searching for data would be nearly impossible. Typically television image generates data rates exceeding 10million bytes/sec. There are other image sources that generate even higher data rates. Storage and transmission of such data require large capacity and bandwidth, which could be expensive. Image data compression technique, concerned with the reduction of the number of bits required to store or transmit image without any appreciable loss of information.

2. THE DISCRETE COSINE TRANSFORM

The discrete cosine transform is a fast transform that takes a input and transforms it in to linear combination of weighted basis function, these basis functions are commonly the frequency, like sine waves. It is widely used and robust method for image compression, it has excellent energy compaction for highly correlated data, which is superior to DFT and WHT. Though KLT minimizes the MSE for any input image, KLT is seldom used in various applications as it is data independent obtaining the basis images for each sub image is a non trivial computational task, in contrast DCT has fixed basis images. Hence most practical transforms coding systems are based on DCT which provides a good compromise between the information packing ability and computational complexity. Compared to other independent transforms it has following advantages, can be implemented in single integrated circuit has ability to pack most information in fewer number of coefficients and it minimizes the block like appearance, called blocking artifact that results when the boundary between sub images become visible.

One dimensional DCT is defined as:

\[ c(i) = a(n) \sum_{x=0}^{N-1} f(n) \cos \left( \frac{(2x+1)iN}{2N} \right) \]  (2.1)

where \( i = 0, 1, 2, \ldots, N-1 \)

Inverse DCT is defined as:

\[ f(n) = a(i) \sum_{x=0}^{N-1} c(i) \cos \left( \frac{(2x+1)iN}{2N} \right) \]  (2.2)
The correlation between different coefficients of DCT is quite small for most of the image sources and since DCT processing is asymptotically Gaussian. Those transformed coefficients are treated as they are mutually independent. In general, DCT correlates the data being transformed so that most of its energy is packed in a few of its transformed coefficient’s. The goal of the transformation process is to de-correlate the pixels of each sub images or to pack as much information as possible into the smaller number of transform coefficients. The Quantization stage then selectively eliminates or more coarsely quantizes the coefficients that carry the least information. These coefficients have the smallest impact on the constructed sub image quality. The encoding process terminates by coding the quantized coefficient

\[ a(n) = \frac{1}{\sqrt{N}} \text{ for } n = 0 \]
\[ a(n) = \frac{2}{\sqrt{N}} \text{ for } n = 1, 2, \ldots, N-1 \]

**COMPRESSION PROCEDURE**

For a given image, you can compute the DCT of, say each row, and discard all values in the DCT that are less than a certain threshold. We then save only those DCT coefficients that are above the threshold for each row, and when we need to reconstruct the original image, we simply pad each row with as many zeroes as the number of discarded coefficients, and use the inverse DCT to reconstruct each row of the original image. We can also analyze image at the different frequency bands, and reconstruct the original image by using only the coefficients that are of a particular

3. LOSSLESS DATA COMPRESSION

3.1 SOURCE CODING

The most significant feature of the communication system is its unpredictability or uncertainty. The transmitter transmits at random any one of the pre-specified messages. The probability of transmitting each individual message is known. Thus our quest for an amount of information is virtually a search for a parameter associated with a probability scheme. The parameter should indicate a relative measure of uncertainty relevant to the occurrence of each message in the message ensemble. The principle of improbability (which is one of the basic principles of the media world)—“if a dog bites a man, it’s no news, but if a man bites a dog, it’s a news”—helps us in this regard. Hence there should be some sort of inverse relationship between the probability of an event and the amount of information associated with it. The more the probability of an event, the less is the amount of information associated with it, & vice versa. Thus:

\[ I(x_i) = \log \left( \frac{1}{p(x_i)} \right) \]  (3.1)

Where \( x_i \) is an event with a probability \( p(x_i) \) & the amount of information associated with it is \( I(x_i) \).

Now let there be another event \( y_k \) such that \( x_i \) & \( y_k \) are independent. Hence probability of the joint event is \( p(x_i, y_k) = p(x_i) p(y_k) \) with associated information content,

\[ I(x_i, y_k) = \log \left( \frac{1}{p(x_i, y_k)} \right) = \log \left( \frac{1}{p(x_i)p(y_k)} \right) \]  (3.2)

The total information \( I(x_i, y_k) \) must be equal to the sum of individual information \( I(x_i) \) & \( I(y_k) \), where \( I(y_k) = \log \left( \frac{1}{p(y_k)} \right) \) Thus it can be seen that function \( f( \_ ) \) must be a function which converts the operation of multiplication into addition, Logarithm is one such function. Thus, the basic equation defining the amount of information (or self-information) is,

\[ I(x_i) = \log_2 \left( \frac{1}{p(x_i)} \right) \]  (3.3)
When base is 2 (or not mentioned) the unit is bit, when base is e the unit is Nat, when base is 10 the unit is decbit or Hertley.

3.2 ENTROPY

Entropy is defined as the average information per individual message. Let there be L different messages or symbols m₁, m₂, …., mₙ, with their respective probabilities of occurrences be p₁, p₂, …., pₙ. Let us assume that in a long time interval, M messages have been generated. Let M be very large so that M>>L. The total amount of information in all M messages is given as:

\[ I = M \times (p₁) \times \log₂ \left( \frac{1}{p₁} \right) + M \times (p₂) \times \log₂ \left( \frac{1}{p₂} \right) + \ldots + M \times (pₙ) \times \log₂ \left( \frac{1}{pₙ} \right) \]

So, the average information per message, or entropy, will be then be

\[ H = \frac{1}{M} \left( \sum_{i=1}^{L} (pᵢ) \times \log₂(pᵢ) \right) \]

H is in bits/symbol, if symbol or message rate in symbols/second is given then the Entropy in bits/second will be given as:

\[ H' = -r \sum_{i=1}^{L} (pᵢ) \times \log₂(pᵢ) \] (3.5)

There are two types of code possible:

1. Fixed Length Code: All code words are of equal length
2. Variable Length Code: All code words are not of equal length. In such cases, it is important for the formation of uniquely decodable code that all the code words satisfy the Prefix condition, which states that “no code word forms the prefix of any other code word”.

The necessary & sufficient condition for the existence of a binary code with code words having lengths n₁ ≤ n₂ ≤ … ≤ nₙ that satisfy the prefix condition is:

\[ \sum_{k=1}^{n} 2^{-n_k} \leq 1 \] (3.6)

This is known as Kraft Inequality.

3.3 SOURCE CODING THEOREM

Let x be ensemble of letters from a discrete memory less source with finite Entropy H(x) & output symbols xᵢ with probabilities P(xᵢ), i=1,2,3, …., L. It is possible to construct a code that satisfies the prefix condition & has an average length R that satisfies the following inequality,

\[ H(x) \leq R < H(x) + 1 \]

And the efficiency of the prefix code is defined as

\[ \eta = \frac{H(x)}{R} \]

Where,

\[ H(x) = -\sum_{i=1}^{L} P(xᵢ) \log₂ [P(xᵢ)] \]

And R = -\sum_{i=1}^{L} nᵢ \log₂ [P(xᵢ)]

Here nᵢ denotes the length of i-th code word.

The source coding theorem tells us that for any prefix code used to represent the symbols from a source, the minimum number of bits required to represent the source symbols on an average must be at least equal to the entropy of the source. If we have found a prefix code that satisfies R=H(x) for a certain source X, we must abandon further search because we cannot do any better. The theorem also tells us that a source with higher entropy (uncertainty) requires on an average, more number of bits to represent the source symbols in terms of a prefix code.

3.4 HUFFMAN CODING

Huffman coding is an efficient source coding algorithm for source symbols that are not equally probable. A variable length encoding algorithm was suggested by Huffman in 1952, based on the source symbol probabilities P(xᵢ), i=1,2,……L. The algorithm is optimal in the sense that the average number of bits required to represent the source symbols is a minimum provided the prefix condition is met. The steps of Huffman coding algorithm are given below:

1. Arrange the source symbols in increasing order of their probabilities.
2. Take the bottom two symbols & tie them together as shown in Figure 3.1. Add the probabilities of the two symbols & write it on the combined node. Label the two branches with a ‘1’ & a ‘0’.
3. Treat this sum of probabilities as a new probability associated with a new symbol. Again pick the two smallest probabilities, tie them together to form a new probability. Each time we perform the combination of two symbols we reduce the total number of symbols by one. Whenever we tie together two probabilities (nodes) we label the two branches with a ‘0’ & a ‘1’.

4. Continue the procedure until only one procedure is left (& it should be one if your addition is correct). This completes the construction of the Huffman Tree.

5. To find out the prefix codeword for any symbol, follow the branches from the final node back to the symbol. While tracing back the route read out the labels on the branches. This is the code word for the symbol.

The algorithm can be easily understood using the following example.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Code word</th>
<th>Code Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>0.46</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X2</td>
<td>0.30</td>
<td>00</td>
<td>2</td>
</tr>
<tr>
<td>X3</td>
<td>0.12</td>
<td>010</td>
<td>3</td>
</tr>
<tr>
<td>X4</td>
<td>0.06</td>
<td>0110</td>
<td>4</td>
</tr>
<tr>
<td>X5</td>
<td>0.03</td>
<td>01110</td>
<td>5</td>
</tr>
<tr>
<td>X6</td>
<td>0.02</td>
<td>011110</td>
<td>6</td>
</tr>
<tr>
<td>X7</td>
<td>0.01</td>
<td>011111</td>
<td>7</td>
</tr>
</tbody>
</table>

The Entropy of the source is found to be

\[ H(x) = - \sum_{i=1}^{L} (P(x_i) \cdot \log_2(P_i)) \]

\[ H(x) = 1.978 \]

And \( R = - \sum_{i=1}^{L} n_i \log_2 \left[ p(x_i) \right] \)

\[ R = 1(0.46) + 2(0.30) + 3(0.12) + 4(0.06) + 5(0.03) + 6(0.02) + 6(0.01) \]

\[ R = 1.99 \]

Efficiency \( \eta = \frac{H(x)}{R} \)

\[ \eta = 0.994 \]

3.5 Huffman Decoding

The Huffman Code in Table 3.1 & FIGURE 3.2 is an instantaneous uniquely decodable block code. It is a block code because each source symbol is mapped into a fixed sequence of code symbols. It is instantaneous because each codeword in a string of code symbols can be decoded without referencing succeeding symbols. That is, in any given Huffman code, no codeword is a prefix of any other codeword. And it is uniquely decodable because a string of code symbols can be decoded only in one way. Thus any string of Huffman encoded symbols can be decoded by examining the individual symbols of the string in left to right manner. Because we are using an instantaneous uniquely decodable block code, there is no need to insert delimiters between the encoded pixels.

For Example consider a 19 bit string 101000111101101111 which can be decoded uniquely as \( x_1, x_3, x_2, x_4, x_1, x_1, x_7 \).

A left to right scan of the resulting string reveals that the first valid code word is 1 which is a code symbol for, next valid code is 010 which corresponds to \( x_1 \), continuing in this manner, we obtain a completely decoded sequence given by \( x_1, x_3, x_2, x_4, x_1, x_1, x_7 \).

4. Image Compression Using DCT
Different output parameters like SNR & Compression Ratio determine the efficiency of system. These parameters in turn depend mainly on different input parameters like number of quantization levels, number of diagonals considered for Zigzag traversing (or simply, the percentage of pixels neglected), size of blocks taken from image matrix for DCT transform & in some cases various other parameters like Signal to Noise Ratio (SNR) in the transmission channel.

4.1 GRAY IMAGE COMPRESSION USING DCT

The values of corresponding input & output parameters for different sized gray images by varying the Quantization Level & Block size taken for DCT are tabularized as shown. Number of Coefficients selected for transmission is kept fixed for the entire observation.

Table 4.1

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Block Size</th>
<th>Diagonals Taken</th>
<th>Quantization Level</th>
<th>SNR (Received Image)</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>128x128</td>
<td>4x4</td>
<td>4</td>
<td>3</td>
<td>8.00</td>
<td>3.99</td>
</tr>
<tr>
<td>128x128</td>
<td>4x4</td>
<td>5</td>
<td>3</td>
<td>9.27</td>
<td>3.55</td>
</tr>
<tr>
<td>128x128</td>
<td>4x4</td>
<td>6</td>
<td>3</td>
<td>10.20</td>
<td>3.36</td>
</tr>
<tr>
<td>128x128</td>
<td>8x8</td>
<td>9</td>
<td>3</td>
<td>6.48</td>
<td>3.90</td>
</tr>
<tr>
<td>128x128</td>
<td>8x8</td>
<td>10</td>
<td>3</td>
<td>6.59</td>
<td>3.59</td>
</tr>
<tr>
<td>128x128</td>
<td>8x8</td>
<td>11</td>
<td>3</td>
<td>6.68</td>
<td>3.33</td>
</tr>
<tr>
<td>128x128</td>
<td>16x16</td>
<td>16</td>
<td>3</td>
<td>4.91</td>
<td>4.51</td>
</tr>
<tr>
<td>128x128</td>
<td>16x16</td>
<td>18</td>
<td>3</td>
<td>4.91</td>
<td>4.22</td>
</tr>
<tr>
<td>128x128</td>
<td>16x16</td>
<td>24</td>
<td>3</td>
<td>4.60</td>
<td>3.71</td>
</tr>
<tr>
<td>128x128</td>
<td>16x16</td>
<td>27</td>
<td>3</td>
<td>4.49</td>
<td>3.58</td>
</tr>
</tbody>
</table>

4.2 COLOR IMAGE COMPRESSION USING DCT

Now for color image taking 8X8 as the block size for DCT, the SNR & the Compression Ratio obtained for different number of diagonals.
Table 4.2

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Block Size</th>
<th>Diagonals Taken</th>
<th>Quantization Level</th>
<th>SNR (Original Image)</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>256x256</td>
<td>4x4</td>
<td>d</td>
<td>1</td>
<td>3.00</td>
<td>3.00</td>
</tr>
<tr>
<td>256x256</td>
<td>8x8</td>
<td>d</td>
<td>4</td>
<td>0.57</td>
<td>1.11</td>
</tr>
<tr>
<td>256x256</td>
<td>4x4</td>
<td>d</td>
<td>5</td>
<td>10.20</td>
<td>3.20</td>
</tr>
</tbody>
</table>

| 256x256    | 16x16      | 8               | 4                  | 7.74                 | 4.16              |
| 256x256    | 16x16      | 8               | 7                  | 3.19                 | 3.81              |
| 256x256    | 16x16      | 8               | 8                  | 8.82                 | 3.49              |
| 256x256    | 16x16      | 8               | 5                  | 5.64                 | 3.78              |
| 256x256    | 16x16      | 8               | 6                  | 1.88                 | 3.81              |
| 256x256    | 16x16      | 8               | 1                  | 3.91                 | 3.04              |
| 256x256    | 16x16      | 8               | 7                  | 1.11                 | 3.01              |
| 256x256    | 16x16      | 8               | 9                  | 0.64                 | 0.76              |

4.3 COLOR IMAGE COMPRESSION USING DCT

Now for color image taking 8X8 as the block size for DCT, the SNR & the Compression Ratio obtained for different number of diagonals taken for some 256X256 Images are as follows

4.3.1 Original image of size 256x256

CONCLUSION

Thus we see that the SNR (of received image) & the compression ratio are directly affected by changes in quantization level & number of diagonals. As expected the SNR increases & compression ratio decreases by increasing the number of diagonals & number of quantization levels, though the effect of quantization level is
more pronounced. Apart from such obvious results, it can also be noticed that SNR decreases & compression ratio increases with the increase in the block size taken for DCT (keeping the percentage of pixels taken to be almost constant with respect to block sizes). This behavior can be explained on the fact that a longer string of continuous zeros can be obtained (after neglecting the similar percentage of pixels) by increasing the block size. One more behavior worth analyzing is that when the block size taken for DCT is increased to 16X16, then on increasing the participating number of diagonals compression ratio is decreased as expected but the SNR also reduces (though very slightly). This again can be explained on the basis of the fact that an increasing number of symbols are being quantized by the same number of quantization level resulting increase in quantization error. So, in this case SNR can be increased by increasing the number of quantization levels.

ACKNOWLEDGEMENTS
The author would like to thank Dr. A.K.shrivastava professor director global engg.collage and professors and H.O.D Electronics and communication Engineering Department,S.I.R.T. Jabalpur Thank you for constant encouragement and valuable guidance during our research work.

REFERENCES

AUTHOR BIOGRAPHY
ADARSH KHARE
1 am diploma and degree holder with electronics and communication and now doing master of Engg.with DIGITAL COMMUNICATION from SRIT – collage Jabalpur.