A Lossless Image Compression Algorithm Using Variable Block Size Segmentation

N. Ranganathan, *Senior Member, IEEE,* Steve G. Romaniuk, and Kameswara Rao Namuduri

Abstract—**The redundancy in digital image representation can be classified into two categories: local and global. In this paper, we present an analysis of two image characteristics that give rise to local and global redundancy in image representation. Based on this study, we propose a lossless image compression scheme that exploits redundancy both at local and global levels in order to obtain maximum compression efficiency. The proposed algorithm segments the image into variable size blocks and encodes them depending on the characteristics exhibited by the pixels within the block. The proposed algorithm is implemented in software and its performance is better than other lossless compression schemes such as the Huffman, the arithmetic, the Lempel–Ziv and the JPEG.**

I. Introduction

DATA compression is the process of eliminating or reducing the redundancy in data representation in order to achieve savings in storage and communication costs. Data compression techniques can be classified into two categories: lossless and lossy schemes. In lossless methods, the exact original data can be recovered while in lossy schemes only a close approximation of the original data can be obtained. The lossless methods are also called entropy coding schemes, since there is no loss of information content during the process of compression. In lossy compression methods, transform coding techniques such as Fourier [27], Karhunen–Loeve [25], orthogonal transforms [26], and other techniques such as DPCM [22], vector quantization [11], [15], [17] are used and such methods produce compression ratios of up to 100-to-1 depending on the quality of images obtained after reconstruction.

The classical lossless compression schemes are based on tree-based codes that represent a large class of variable-length encoding schemes such as Huffman codes [4], Shannon–Fano codes [1], [2], universal codes of Elias [6], the Fibonacci codes [3], etc. The tree-based compression methods have been used for large scientific and text files as well as image date and usually yield compression ratios in the range from 2–3. In the absence of a suitable model of data to be encoded, the arithmetic [13], [16] and Lempel-Ziv codes [7], [8] and the run-length code [5] provide better adaptive codes. The lossless methods in combination with lossy methods have been used in some image applications. For example, the baseline system proposed in ISO-JPEG [18], an international still image standard, recommends the use of Huffman coding or arithmetic coding to encode the compressed image obtained after transform and quantization steps to further exploit redundancy. Lossless methods are also used in specialized applications such as medical imaging and satellite photography (such as level 0 or level 1A space image data of NASA [20]) where reliability of reproduction of images is a critical factor. For a survey of various methods for image compression, the reader is referred to [21].

The redundancy in digital image representation can be classified into two categories: local and global. These two types of redundancy could be attributed to two specific characteristics exhibited by the image data. Local redundancy corresponds to the coherence, smoothness or correlation in the image data that is due to the fact that the gray level values within a neighborhood vary gradually rather than abruptly. Illumination and shadowing effects also contribute to gradual variation of data in images. Global redundancy could be attributed to the repetition of patterns within an image. Image compression algorithms eliminate or reduce local redundancy by representing the neighborhood in a compact form. For example, in run length coding [5] scheme, runs of pixels with the same gray level are replaced by the gray level and its run length. In compression schemes based on differential pulse code modulation [22], [23] and predictive coding [24] methods the next input is predicted based on the digitally coded past. Global redundancy can be eliminated or reduced by encoding the repeated patterns (linear or 2-D sub-blocks) by using suitable coding methods. Lempel–Ziv [7], [8] scheme searches for repeated linear patterns within a fixed size window and replaces the repeated patterns by suitable pointers to the previous occurrences. Arithmetic coding [13] and Huffman [4] schemes exploit the skewness in the distribution of patterns for optimal encoding of the given data. Lossy compression schemes based on vector quantization [11] parse the image into a sequence of groups of pixels. Each group of pixels is referred to as a vector. The encoder views an input vector and generates a channel codeword. The decoder uses a look-up table indexed by the channel codeword to decode the compressed data [21]. Most of image compression methods follow either local or global redundancy elimination methods. However, for achieving the maximum possible compression rates it is mandatory to exploit redundancy both at local and global levels.

The contribution of the paper is two-fold. In the first part of the paper, a mathematical analysis of local and global characteristics exhibited by image data is presented with an

Manuscript received May 11, 1993; revised October 24, 1994. This work was supported in part by a grant from Enterprise Florida Innovation Partnership FTRI Fund (formerly Florida High Technology and Industry Council). The associate editor coordinating the review of this paper and approving it for publication was Prof. Rama Chellapa.

N. Ranganathan is with the Center for Microelectronics Research, Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA.

S. G. Romaniuk is with the Department of Information Systems and Computer Science, National University of Singapore, Singapore 0511.

K. R. Namuduri is with GTE Information Services, Tampa, FL 33602 USA.

IEEE Log Number 9413840.

objective of deriving a lossless image compression method that works well on most of the images. In general, the performance of image compression methods are image dependent. A compression scheme performs well on a given image, if the scheme is based on exploiting the specific type of characteristic that is present within the image. Hence, in order to derive a compression scheme that performs well on most images, we need to find out the characteristics that are widely exhibited by images. Estimating the distribution of image characteristics and the resulting compression efficiency is a very difficult task due to the huge amount of computations involved. However, by observing a simplified image domain (images of size 32×32 with 256 gray levels) closely, we were able to plot and compare the distributions of two common types of image characteristics generally utilized for lossless image compression. This mathematical analysis also provides an understanding of the effect of some design parameters such as the block size and search space on compression efficiency.

The second part of the paper presents an algorithm for lossless compression based on some of the direct observations from the mathematical model. The proposed image compression scheme exploits both types of redundancy in order to get maximum compression for most types of images. The performance of the proposed algorithm is studied by software implementation. The proposed algorithm works better than other lossless compression schemes such as Huffman [4], arithmetic [16], Lempel–Ziv [7], [8], and the JPEG [29]. The performance factor, Compression Efficiency (CE), used in this paper for performance comparison is defined as follows:

$$\frac{\text{original image size} - \text{compressed image size}}{\text{original image size}} \times 100.$$

The outline of this paper is as follows: A detailed explanation of two typical image characteristics that lead to redundancy is presented in Section II. A mathematical model that represents the distribution of the image characteristics and quantitative analysis of the model is presented for a sample image domain. Based on this study, a new algorithm for lossless image compression is proposed in Section III and its complexity analysis is presented in Section IV. Section V concludes the paper.

II. Local and Global Redundancy in Images: An Analysis

In general, images exhibit characteristics that give rise to local and global redundancies in image representation. The compression efficiency that can be obtained on an image depends mainly on the image characteristics. Hence, image compression schemes are generally designed to exploit the presence of such characteristics. Further, implementation aspects such as the image partitioning strategy, the block size, the area of search space, etc., influence the compression efficiency as well as the computational complexity. An algorithm will achieve the best possible compression efficiency if: 1) it exploits all the characteristics that contribute to the compression and 2) employs efficient implementation strategies. However, it is not possible to design an algorithm that will exploit every characteristic that leads toward compression. Hence, it is desirable to study the distribution of such characteristics in images in general and pick those characteristics that will result in higher compression efficiency. Then it is possible to design a versatile algorithm (i.e., an algorithm that works for most types of images). Further more, it is important to select the design parameters carefully so that the algorithm works efficiently (in terms of computational time as well as compression efficiency). In this section, two image characteristics: *smoothness* and *similarity* that appear widely in images are studied in detail in order to understand their effect in image compression. The two characteristics are discussed below:

1) Smoothness: A given neighborhood (or block) in an image often exhibits coherence or correlation a property that is commonly referred to as smoothness of the data. The difference (d) between the minimum and maximum gray levels within a neighborhood is a suitable parameter to measure the smoothness. For example if $d = 0$, it means that every pixel in the block is of same gray level. In this case, the block can be represented by one pixel and the size of the box. This representation is same as the run-length coding and it requires the least amount of space. Even if the difference is not equal to zero, it will generally be very small within a neighborhood. Hence, any given neighborhood can be represented by the minimum gray level (*min*) in the block and the *offset*s corresponding to the other pixels in the block with respect to the *min*. Such a scheme is referred as *base-offset* scheme (for representing a neighborhood) in this paper. By knowing the value of d within a block, it is clearly possible to compute the compression efficiency that can be obtained within the block. Since the parameter d directly affects the compression efficiency, it can be considered as the parameter for estimating the efficiency of compression algorithms based on the *smoothness* characteristic.

2) Similarity: In general, a given pattern might repeat itself anywhere in the rest of the image. If a pattern repeats itself n times in an image, then $(n - 1)$ patterns can be replaced by pointers to one of the occurrences of the pattern. For example, Lempel–Ziv compression schemes search for linear repeated patterns (or strings) in the data and replaces them with pointers to their previous occurrences [7], [8]. To determine how good this characteristic is from image compression point of view, we need to partition an image into subregions, consider each subregion as a pattern, and estimate how many times each pattern repeats itself in the entire image. Once the amount of repetition is known, the compression efficiency that can be achieved on the image can be estimated. We reformulate this *similarity* estimation problem as a grid coloring problem in Section II-B and estimate the distribution of this characteristic within images.

As a first step in the mathematical analysis, the set of all images of a given representation scheme is divided into a distinct number of classes. This classification is based on the degree of *smoothness* and *similarity* present in each class of images. The purpose of this classification is:

- to estimate the number of images that exhibit a specific characteristic (*smoothness* or *similarity*) at varying levels.
- to estimate the maximum compression that is possible to obtain in each class of images.

- to compare the distributions of the characteristics in order to find out which characteristic gives rise to better compression.
- and to estimate the influence of certain design parameters such as the block size and the are of search space on compression efficiency.

For this analysis, the set $\mathcal{I}_N$ of images of size $N \times N$, with C number of colors is considered. The cardinality of this set is C^{N^2}. This set of images is classified into n number of different classes (S_1 to S_n) based on the distribution of the two characteristics i) *smoothness* and ii) *similarity* in them.

A. Classification Based on Smoothness Characteristic

To study the *smoothness* characteristic, a given image is divided into a set of blocks. Given that each pixel can take any gray color value from the set $[0, C-1]$, the maximum difference in any block lies within $[0, C-1]$. Below, we present a theorem and prove that this classification is indeed possible.

Theorem 1: Let $N \times N$ be the image size, and let C be the maximum number of gray color values in any given image $I \in \mathcal{I}_N$. Then the set of images $\mathcal{I}_N$ can be classified into a finite number of disjoint subsets based on the degree of *smoothness* exhibited by the images, such that there is an upper bound of compression efficiency associated with each of the subsets.

Proof: This theorem is proved in two steps. First, the total number of distinct blocks of size $K \times K$ are classified into a finite number of disjoint subsets based on the maximum difference (d) between any two pixels in the $K \times K$ block. In the second step, each image is considered as an assembly of $K \times K$ blocks and the entire set of images thus formed is classified into a finite number of disjoint subsets based on the upper bound of compression that can be achieved on each subset of images.

If within a $K \times K$ block, each pixel takes on a value from within the range $[0, C-1]$, and if the maximum difference between any two pixels in the $K \times K$ block is d, then the number of distinct $K \times K$ blocks $N[K, d]$ that can be formed is given by the following formulas.

$$\begin{aligned} N[K, 0] &= C \\ N[K, 1] &= (C-1) \sum_{i=1}^{i=K^2-1} \binom{K^2}{i} \\ N[K, d] &= (C-d) \sum_{i=1}^{K^2-1} \sum_{j=i}^{K^2-1} \binom{K^2}{i} \cdot \binom{K^2-i}{K^2-j} \\ &\quad \cdot (d-1)^{j-i} \quad \text{for} \quad 1 < d \leq C-1. \end{aligned} \tag{1}$$

The above formulas can be interpreted as follows: When d is equal to zero, the number of distinct patterns that can be formed in a block of size $K \times K$ will be equal to the maximum gray colors allowed (C) and so on. Since the classification is based on d, and d can vary from 0 to $C-1$, the number of distinct sets will be equal to C.

We prove the correctness of the above formulas by showing that the sum of distinct blocks within the above described disjoint sets adds up to the total number of blocks of size $K \times K$. That is

$$\sum_{d=0}^{C-1} N[K, d] = C^{K^2}. \tag{2}$$

Consider the summation (from (2)) $\sum_{i=1}^{K^2-1} \sum_{j=i}^{K^2-1} \binom{K^2}{i} \binom{K^2-i}{K^2-j}(d-1)^{j-i}$. Let $a = K^2$. Expanding the two summations we obtain

$$\begin{aligned} &\sum_{i=1}^{a-1} \sum_{j=i}^{a-1} \binom{a}{i} \binom{a-i}{a-j} (d-1)^{j-i} \\ &= \binom{a}{1} \left[\binom{a-1}{1} (d-1)^{a-2} + \binom{a-1}{2} (d-1)^{a-3} \right. \\ &\quad \left. + \cdots + \binom{a-1}{a-1} (d-1)^0 \right] \\ &\quad + \binom{a}{2} \left[\binom{a-2}{1} (d-1)^{a-3} + \binom{a-2}{2} (d-1)^{a-4} \right. \\ &\quad \left. + \cdots + \binom{a-2}{a-2} (d-1)^0 \right] \\ &\quad + \cdots + \binom{a}{a-1} \left[\binom{1}{1} (d-1)^0 \right]. \end{aligned} \tag{3}$$

The above summation can be simplified by observing that $\sum_{i=1}^{n} \binom{n}{i} x^{n-i} = (x+1)^n - x^n$. This simplification results in

$$\begin{aligned} &\sum_{i=1}^{a-1} \sum_{j=i}^{a-1} \binom{a}{i} \binom{a-i}{a-j} (d-1)^{j-i} \\ &= \binom{a}{1} [d^{a-1} - (d-1)^{a-1}] \\ &\quad + \binom{a}{2} [d^{a-2} - (d-1)^{a-2}] \\ &\quad + \cdots + \binom{a}{a-1} [d^1 - (d-1)^1] \\ &= \sum_{i=1}^{a-1} \binom{a}{i} d^{a-i} - \sum_{i=1}^{a-1} \binom{a}{i} (d-1)^{a-i} \\ &= [(d+1)^a - d^a - 1] - [(d)^a - (d-1)^a - 1] \\ &= (d+1)^a - 2d^a + (d-1)^a. \end{aligned} \tag{4}$$

Using the above result, we can express the summation of subsets $\sum_{d=2}^{C-1} N[K, d]$ given in (2) as follows:

$$\begin{aligned} &\sum_{d=2}^{C-1} N[K, d] \\ &= \sum_{d=2}^{C-1} (C-d)[(d+1)^a - 2d^a + (d-1)^a] \\ &= C \sum_{d=2}^{C-1} [(d+1)^a - 2d^a + (d-1)^a] \\ &\quad - \sum_{d=2}^{C-1} [d(d+1)^a - 2d^{a+1} + d(d-1)^a] \\ &= C[1 - 2^a - (C-1)^a + C^a] \\ &\quad - [2 - 2^a - C(C-1)^a + (C-1)C^a] \\ &= C^a + 2^a + C - C * 2^a - 2. \end{aligned} \tag{5}$$

The above result delivers the summation from $N[K, 2]$ to $N[K, C-1]$. Adding $N[K, 0]$, which is equal to C, and $N[K, 1]$, which is equal to $(C-1)2^a - 2(C-1)$ results in the sum of all subsets, yielding $\sum_{d=0}^{C-1} N[K, d] = C^a = C^{K^2}$. This proves the validity of (1).

Given a block of size $K \times K$ in which pixels can take any value from within the set $[0, C-1]$, if the maximum difference between any two pixels is d, then the number of bits required ($BR[K, d]$) to represent the block using base-offset method described in Section I, is given by

$$BR[K, d] = \log_2(C) + (K^2 - 1)\log_2(d+1). \quad (6)$$

The above equation is derived as follows: A $K \times K$ block can be represented by its first pixel that requires $\log_2(C)$ bits followed by $K^2 - 1$ offsets each with $\log_2(d+1)$ bits. Hence, the compression efficiency $CE[K, d]$ for $K \times K$ block is given by

$$CE(K, d) = \frac{K^2 \log_2(C) - BR[K, d]}{K^2 \log_2(C)} \times 100. \quad (7)$$

Given N the size of an image, C the number of possible colors assigned to each pixel, K the block size, and $CE[K, d]$ the **compression efficiency** for different values of d, then the entire set of images $\mathcal{I}_N$ can be classified into a finite number of disjoint subsets based on an upper bound of compression that can be obtained for such images as shown below.

An image of size $N \times N$ can be viewed as an assembly of $K \times K$ blocks where the total number of $K \times K$ blocks is given by $\lfloor (N/K)^2 \rfloor$. Each of these $K \times K$ blocks gives rise to a different compression efficiency depending on the maximum difference d within each $K \times K$ block.

The gray level values 0 through $C-1$ can be partitioned into $\log_2(C)$ sets depending on the number of bits required to represent the gray level. These subsets are as follows:

partition:	range of gray levels
partition 0:	$(0, 1)$
partition 1:	$(2, 3)$
partition 2:	$(4, 5, 6, 7)$
partition 3:	$(8, \cdots, 15)$
$\cdots$	
partition l:	$(2^l, \cdots, 2^{l+1} - 1)$. (8)

Let A: $\langle a_0, a_1, \cdots, a_l \rangle$ represent a tuple where the components a_i represent the number of $K \times K$ blocks belonging to the partition i, subject to the following conditions:

$$i \in [0, \log_2(C)]$$
$$0 \le a_i \le \left\lfloor \left(\frac{N}{K}\right)^2 \right\rfloor$$
$$\sum_{i=0}^{\log_2(C)} [a_i] = \left\lfloor \left(\frac{N}{K}\right)^2 \right\rfloor. \quad (9)$$

Each tuple A gives a possible distribution of different $K \times K$ blocks in an image of size $N \times N$. Suppose A (i.e., the distribution of different $K \times K$ blocks in an image) is known, then the compression efficiency that can be obtained on a given image can be determined from the following equation.

$$CE = 100 \times \left\{ \frac{\sum_{i=0}^{\log_2(C)} \{a_i CE[K, 2^i - 1]\}}{\left\lfloor \left(\frac{N}{K}\right) \right\rfloor^2} + \frac{2\log_2(C) \sum_{i=0}^{N-k\lfloor (N/K) \rfloor - 1} (N-i)}{\left\lfloor \left(\frac{N}{K}\right) \right\rfloor^2} \right\}. \quad (10)$$

To obtain a compression efficiency of $X\%$, within a deviation ϵ, every image requires all tuples obey the following condition:

$$X - \epsilon \le CE \ge X + \epsilon. \quad (11)$$

Let us define the set $\mathcal{S}$ of tuples as

$$\mathcal{S} = \{\langle a_0, a_1, \cdots, a_{\log_2(C)} \rangle \text{ subject to Condition (10) and Condition (11)} \}. \quad (12)$$

$\mathcal{S}$ is the solution space of all tuples that satisfy conditions (11) and (12). Furthermore, let us define BC_i as representing the total number of $K \times K$ blocks in which the degree of variance d varies from 2^i to $2^{i+1} - 1$ according to (9).

$$\begin{aligned} BC_i &= N[K, 0], \quad \text{when} \quad i = 0 \\ &= \sum_{j=2^i}^{2^{i+1}-1} N[K, j], \quad i \in [1, \log_2(C)]. \end{aligned} \quad (13)$$

For any tuple $A \in \mathcal{S}$, we can calculate the total number of combinations ($Comb\,[A]$) in which all $K \times K$ blocks belonging to tuple A can be arranged to form an image, such that every one of the constructed images can be compressed by $X\%$ within a deviation ϵ. We can write

$$Comb\,[A] = \prod_{i=0}^{\log_2(C)} BC_i(a_i^{(N/K)^2 - \Sigma_{k=0}^{j-1} a_k}). \quad (14)$$

$Comb\,[A]$ represents the number of images that can be formed from the $K \times K$ blocks selected according to the tuple A. Finally, we can estimate the total number of images that can be formed from all tuples belonging to the set $\mathcal{S}$ as

$$Comb\,[S] = \sum_{A \in \mathcal{S}} Comb\,[A]. \quad (15)$$

Since the tuples in $\mathcal{S}$ are chosen according to (11), $Comb\,[S]$ represents the number of images that give rise to $X \pm \epsilon\%$ compression. Thus, we have shown that the set of all images $\mathcal{I}_N$ can indeed be partitioned, such that on every image $I \in \mathcal{I}_N$ $X\%$ compression within a deviation $\pm\epsilon$ can be obtained. □

B. Classification Based on Similarity Characteristic

In this section, a classification of images based on the *similarity* characteristic is described. Suppose, an image is divided into blocks of size $K \times K$, then it consists of b blocks, where each block can take on p possible patterns, where b and p are given by

$$\begin{aligned} b &= \frac{N^2}{K^2} \\ p &= C^{K^2}. \end{aligned} \tag{16}$$

Furthermore, let $\mathcal{P}$ be the set of all possible color patterns associated with any given block, then we say the cardinality $N[\mathcal{P}] = p$. For the sake of argument, let the compression efficiency we are aiming at be 60%. This desired goal of compression is possible only if 40% of the blocks are colored such that every block is distinct from every other pattern and the remaining 60% of the blocks are colored with duplicated patterns. The cardinality of the set of images exhibiting such characteristic can be determined by estimating the number of ways b blocks can be colored, such that exactly 40% of them are colored with distinct patterns selected from p color patterns. Suppose one is given a grid consisting of 36 blocks to paint and a color palette with 12 different colors. How many ways can one paint this grid using exactly 1, 2, 3, $\cdots$, 12 colors? This task can be performed as follows: let $N[C_k]$ represent number of ways the grid can be colored using exactly k colors. To estimate $N[C_1]$, we find out the number of ways one color can be selected from 12 colors, which is equal to $\binom{12}{1}$ and multiply this number with the number of ways the grid can be colored using one color. This gives us the result $N[C_1] = \binom{12}{1}$. To determine $N[C_2]$, first we find out the number of ways two colors can be selected from 12 colors, which is equal to $\binom{12}{2}$ and multiply this number with the number of ways the grid can be colored using exactly two colors. The number of ways the grid can be colored with exactly two colors can be determined by subtracting the number of ways the grid can be colored using any one of the two colors, which is equal to $\binom{2}{1}1$ from the number ways the grid can be colored using up to two colors (2^{36}). This gives us the result $N[C_2] = \binom{12}{2})[2^{36} - \binom{2}{1}]$. Following this procedure, we can determine $N[C_3]$, $N[C_4]$, $\cdots$, $N[12]$. All the images in the class $N[C_k]$ can be compressed to $(k/36)\%$ giving rise to $(36 - k/36)\%$ compression efficiency where k varies from 1–12. Notice that the maximum value of k is equal to the smaller of the two values b and p. Hence, the possible compression efficiencies run from $0/b$ to $(b - 1/b)\%$ in steps of $1/b$ increments or from $0/b$ to $(p - 1/b)\%$ in steps of $1/b$ increments depending on whether b is smaller or p is smaller. The following theorem provides the classification of images based on the characteristic *similarity* with respect to the compression efficiency parameter.

Theorem 2: Let $N \times N$ be the image size, and let C be the maximum number of gray values in any given image $I \in \mathcal{I}_N$. Then the set of images $\mathcal{I}_N$ can be classified into a finite number of disjoint subsets based on the *similarity* exhibited by the images, such that there is a maximum upper bound of compression efficiency associated with each of the subsets.

Proof: To utilize the characteristic **similarity**, a given image is required to be decomposed into a set of nonoverlapping blocks. If $K \times K$ is the size of each block, the image can be subdivided into b uniformly sized blocks, where b is equal to (N^2/K^2). Each of the blocks can be assigned p color patterns, where $p = C^{K^2}$. If there are k distinct patterns in the image, then an upper bound for the compression efficiency is given by $(b - k/b) \times 100$ since $(b - k)$ out of b blocks can be replaced by pointers. Since, the image consists of b blocks and there are p patterns to choose from, the number of distinct patterns in the image can vary from 1 to b (if $b < p$) or from 1 to p (if $p < b$). This implies, that the entire set of images can be subdivided into b or p disjoint subsets (C_1, C_2, $\cdots$, C_b, $\cdots$, C_p) based on the number of distinctly colored blocks in the image. Each of these subsets gives rise to a different compression efficiency.

Using the block coloring example discussed earlier, we can determine the number of images in each of these subsets (C_1, C_2, $\cdots$, C_b, $\cdots$, C_p). Let $N[C_k]$ be the cardinality of the subset C_k.

$$\begin{aligned}
N[C_1] &= \binom{p}{1}[1] \\
N[C_2] &= \binom{p}{2}\left\{2^b - \binom{2}{1} \cdot \frac{N[C_1]}{\binom{p}{1}}\right\} \\
N[C_3] &= \binom{p}{3}\left\{3^b - \binom{3}{2} \cdot \frac{N[C_2]}{\binom{p}{2}} - \binom{3}{1} \cdot \frac{N[C_1]}{\binom{p}{1}}\right\} \\
&\cdots \\
N[C_r] &= \binom{p}{r}\left\{r^b - \binom{r}{r-1} \cdot \frac{N[C_{b-1}]}{\binom{p}{r-1}}\right. \\
&\qquad \left. - \binom{r}{r-2} \cdot \frac{N[C_{b-2}]}{\binom{p}{r-2}} - \cdots - \binom{r}{1} \cdot \frac{N[C_1]}{\binom{p}{1}}\right\} \\
&\cdots \\
N[C_b] &= \binom{p}{b}\left\{b^b - \binom{b}{b-1} \cdot \frac{N[C_{b-1}]}{\binom{p}{b-1}}\right. \\
&\qquad \left. - \binom{b}{b-2} \cdot \frac{N[C_{b-2}]}{\binom{p}{b-2}} - \cdots - \binom{b}{1} \cdot \frac{N[C_1]}{\binom{p}{1}}\right\} \\
&\cdots \\
N[C_p] &= \binom{p}{p}\left\{p^b - \binom{p}{p-1} \cdot \frac{N[C_{p-1}]}{\binom{p}{p-1}}\right. \\
&\qquad \left. - \binom{p}{p-2} \cdot \frac{N[C_{p-2}]}{\binom{p}{p-2}} - \cdots - \binom{p}{1} \cdot \frac{N[C_1]}{\binom{p}{1}}\right\}.
\end{aligned} \tag{17}$$

A straightforward way to verify the above classification is by checking whether the sum of all subsets adds up to the total number of images C^{N^2}. Consider the expression for $N[C_p]$ given in (17)

$$N[C_1] + N[C_2] + \cdots + N[C_p] = p^b. \qquad (18)$$

This implies that the sum of all subsets adds up to p^b, which is equal to the total number of images C^{N^2}. Generally, p is larger than b. If there are b blocks in the image, the image can have 1 to b distinct patterns in which case $N[C_{b+1}]$, $N[C_{b+2}]$, $\cdots$, $N[C_p]$ evaluate to zero. This implies that there will be exactly b number of distinct classes. □

C. Analysis

In this section, we provide an analysis based on the mathematical model presented in the previous subsections. The model requires three parameters (C, K, N) to represent a set of images. First, the distributions of each of the two characteristics *smoothness* and *similarity* are independently estimated for the set of images represented by ($C = 256$, $K = 2$, and $N = 8$).

These results are shown in Fig. 1(a). The plots shown in Fig. 1(b) and (c) are obtained by keeping the maximum number of gray colors ($C = 256$) and the block size ($K = 2$) constant and increasing the size of the image (N). The computations are performed using Mathematica software package. Due to the huge size of numbers involved, it was possible to compute the distributions of the characteristics for images of size up to 32×32 only. The plots clearly indicate that as the size of the image is increased, the global characteristic *similarity* becomes a versatile characteristic and it contributes significantly to the compression efficiency. From this first set of results we conclude that any compression scheme should exploit a global characteristic in order to achieve good compression efficiency. In order to increase the compression rates, it is desirable to look for more than one characteristic. In particular, a combination of disjoint characteristics (with no common aspects) such as *similarity* and *smoothness* will result in more efficient compression algorithms. One important issue in implementing a compression scheme based on a global characteristics such as *similarity* is the degree of global search that should be conducted. In Fig. 2(a), we show the effect of block size (K) on the number of images of size N (=16 in this case) exhibiting *similarity* characteristic. Obviously, as the size of the block is increased the number of images that have a high degree of *similarity* decreases drastically. The observed behavior can be explained by the simple fact that as the block size (K) increases the probability of finding identical matches in a fixed size image decreases (fewer combinations can be detected). From this, we can conclude that the choice of K should be limited to a few small values rather than attempting matches for large K. This also implies that the probability of finding pattern matches outside some small predefined neighborhood decreases. This has the desired benefit of decreasing search time and making a global

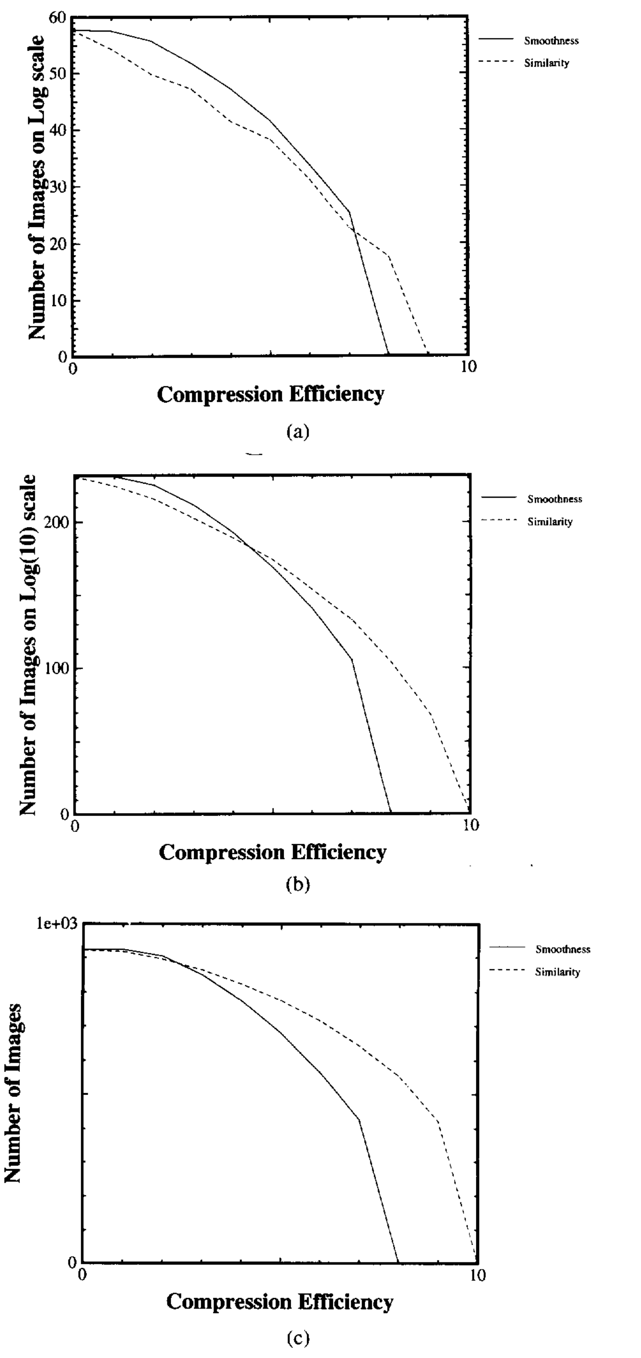


Fig. 1. (a) Distribution of characteristics in images of size 8×8 with 256 gray colors. (b) Distribution of characteristics in image size 16×16 with 256 gray colors. (c) Distribution of characteristics in images of size 32×32 with 256 gray colors.

scheme such as *similarity* practically feasible for large N. The loss in possible compression on the other hand due to this restriction will be limited. Finally, the last graph Fig. 2(b) depicts the results for different sizes of K for exploiting the *smoothness* characteristic. Similar to the previous graph this graph shows that for most of the images significant portion of compression efficiency can be obtained by selecting a block size of 2. Selecting larger sized blocks can improve results. When $N = 16$ for example, a block size of 4 gives highest compression efficiency for most of the images. From this

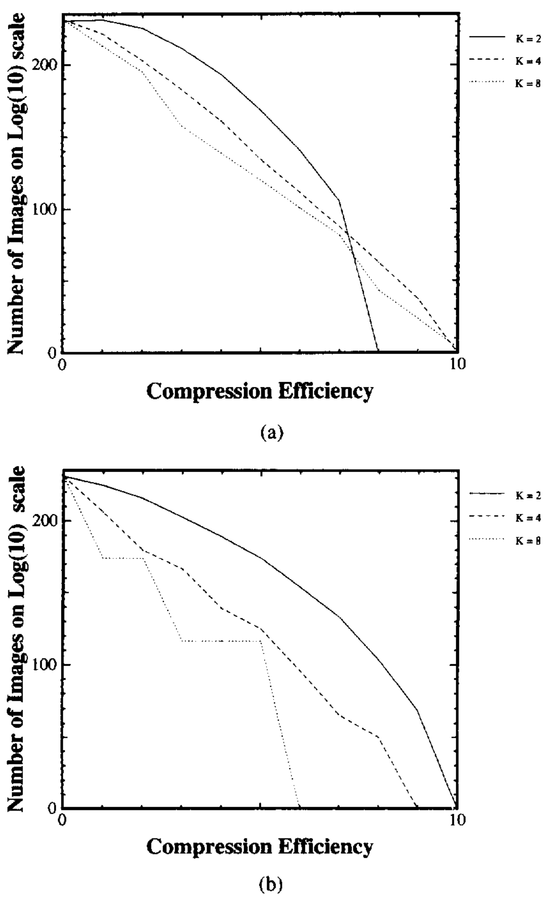


Fig. 2. Distribution of characteristics in images of size 16 × 16 and block sizes of $K = 2, 4,$ and 8. (a) Distribution of similarity characteristic. (b) Distribution of smoothness characteristic.

observation, we can conclude that for exploiting *smoothness*, variable block segmentation is necessary. For the purpose of this paper, any further experiments are not required, but may be considered for future work. Instead, we next draw our attention to applying the results found so far in developing a compression algorithm.

III. A Lossless Image Compression Algorithm Using Variable Block Size Segmentation

In our implementation, a given image is segmented into square sized blocks of sizes eight, four, and two. Each block is classified into one of the twenty eight categories depending on the size of the block as well as characteristic exhibited by the pixels within the block. Three coding schemes are utilized to encode these blocks. The coding schemes and the codewords for each of the twenty eight categories are listed in Table I.

1) *Run-length coding:* If the maximum difference (d) within a block is zero, (i.e., all the pixels in the block under consideration are of the same color), run-length coding is utilized to optimally encode the block. In this case, the block can be represented by one pixel (the base) and the size of the block. In our scheme, each pixel is encoded as an offset with respect to one of its three neighboring pixels (north, east, and north east) or with respect to the average of two of its neighboring pixels (north and east). Hence, it is not necessary to represent the base pixel. Instead, the direction of the selected reference pixel is specified by using two bits. The direction of the reference pixel is updated for every block. The selection of the reference pixel from the neighboring pixels is based on the nearest neighborhood principle presented in [28]. The header entry for this block consists of the category to which this block belongs (in which the size of the block is implicitly encoded) and it is stored in a separate header file. Since three different sizes are considered, three categories (0, 1, and 2 as shown in Table I) belong to this classification.
2) *Block matching:* In our scheme, the area of search space is chosen as four times the size of the block being considered. The codeword for this coding scheme consists of the differences along x and y directions from the pattern under consideration and an identical pattern if it was found within the search space. Hence, the length of the codeword depends on the area of search space. For a 2×2 block, the length of the codeword is eight. Category 5 (shown in Table I) falls under this classification. Again, the header for the block consists of the category to which the block belongs and is stored separately in a header file. In our algorithm, block matching is implemented only for 2×2 blocks to reduce the computational complexity involved with the search.
3) *Base-offset coding scheme:* Base-offset coding scheme is followed if the maximum difference d within the block is sufficiently small but not equal to zero. For 8×8 and 4×4 blocks, the codeword consists of offsets for all

TABLE I
Local and Global Coding Schemes Used in the Compression Algorithm

Category	Block	Offset	Coding	length
	size	range	scheme	(bits)
0	8	0	run-length	2
1	4	0	run-length	2
2	2	0	run-length	2
3	8	1 - 2	base-offset	66
4	4	1 - 2	base-offset	18
5	2	-	block matching	8
6	2	-1 - 0	base-offset	10
7	2	0 - 1	base-offset	10
8	2	-2 - 1	base-offset	10
9	2	0 - 3	base-offset	10
10	2	-3 - 0	base-offset	10
11	2	-3 - 4	base-offset	14
12	2	-5 - 2	base-offset	14
13	2	-2 - 5	base-offset	14
14	2	-7 - 8	base-offset	18
15	2	-4 - 11	base-offset	18
16	2	-11 - 4	base-offset	18
17	2	-2 - 13	base-offset	18
18	2	-13 - 2	base-offset	22
19	2	-15 - 16	base-offset	22
20	2	-4 - 27	base-offset	22
21	2	-27 - 4	base-offset	22
22	2	-20 - 11	base-offset	22
23	2	-11 - 20	base-offset	22
24	2	-31 - 32	base-offset	26
25	2	-15 - 48	base-offset	26
26	2	-48 - 15	base-offset	26
27	2		raw data	32

pixels in the block and the direction of the base pixel. The categories 3, 4, and from 6–26 (in Table I) belong to this classification. The length of the codeword (which excludes the header information) is equal to $n \times \log_2$ (Maximum offset) where n is the number of pixels in the block and two bits to represent the direction of the base pixel.

If none of the above schemes are applicable, then the pixels in the block are represented as they are without any encoding (category 27 in Table I). This scheme is followed only for 2×2 blocks. Four bytes are required to represent the block. The category information is stored separately in the header file.

A. Proposed Algorithm

The proposed algorithm for lossless image compression is given in the form of a flow chart in Fig. 3. The algorithm works as follows: Initially, the given image is partitioned into uniformly sized blocks of size $K \times K$. In our scheme, the initial value of K is selected as eight. If all the pixels in the block are of the same gray level, then the block is encoded using run length encoding scheme. If the degree of variation in the block is reasonably small, then the block is coded using base-offset method. If both the above tests fail, then the block is subdivided into four smaller blocks of size $K/2 \times K/2$. Each of the four blocks are again analyzed for the local and global characteristics described above. This procedure is continued successively by reducing the size of the blocks to half of their size at every step till the block size (K) becomes 2×2. When the size of the block is 2×2, a different approach is followed. If all pixels in the block are of same gray level, the run length coding scheme is applied. Otherwise, the neighborhood area of the block is searched to find if an identical block exists. If such an identical block is located, then the block is encoded by a pointer representing its distance along x and y directions from its previous occurrence. If both above tests fail, then the block is coded using the base-offset scheme provided the degree of variation is less than a specified threshold.

If the degree of variation exceeds the threshold, then the four pixels in the 2×2 block are stored as they are without any compression. As the image is encoded, a compressed image file consisting of the codewords for the variable size blocks as well as a header file are created. The header file keeps track of the category number for each codeword and updates the frequency of each codeword while the coding is in progress. At the end of coding, the header file is compressed separately using Huffman coding scheme.

For comparison purposes, we considered the LZW scheme (compress utility on UNIX), the LZ77 scheme (gzip utility on UNIX), the adaptive Huffman scheme (compact on UNIX), the arithmetic scheme [16] and the JPEG. The software for lossless JPEG is obtained from the public domain facility at Stanford University. The results for a selected set of fourteen images are shown in Table II. From this table, it can be observed that the proposed algorithm yields better compression compared to all other schemes. The characteristic distribution in all the test images is also listed in Table III.

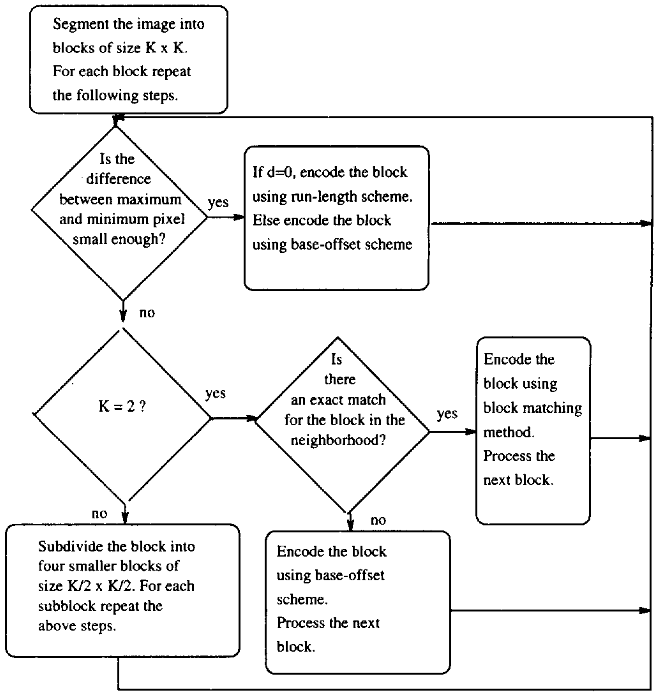

Segment the image into blocks of size K x K. For each block repeat the following steps.
Is the difference between maximum and minimum pixel small enough?
yes
If d=0, encode the block using run-length scheme. Else encode the block using base-offset scheme
no
K = 2 ?
yes
Is there an exact match for the block in the neighborhood?
yes
Encode the block using block matching method. Process the next block.
no
Subdivide the block into four smaller blocks of size K/2 x K/2. For each subblock repeat the above steps.
no
Encode the block using base-offset scheme. Process the next block.

Fig. 3. Flowchart for the proposed compression algorithm.

IV. Time and Space Complexity Analysis

In this section, the time and space complexities (we refer to run-time storage requirement) of the proposed algorithm exploiting the *smoothness* and *similarity* characteristics are analyzed. The results of the analysis are stated in the following theorem.

Theorem 3: Let $N \times N$ be the size of an image and $K \times K$ be the size of the block. An algorithm that implements the smoothness characteristic for image compression has a time order $\mathcal{O}_{Time}(N, K) = N^2$ and requires $\mathcal{O}_{Space}(N, K) = K^2$ space. The time and space complexities are the same for average, best and worst cases. An algorithm that is based on similarity has the following complexity behavior:

$$\begin{aligned}\mathcal{O}_{\text{Avg Time}}(N, K) &= \frac{K^2+1}{4}\left\{\left(\frac{N}{K}\right)^4 - \left(\frac{N}{K}\right)^2\right\}\\ \mathcal{O}_{\text{Best Time}}(N, K) &= \frac{\left(\frac{N}{K}\right)^4 - \left(\frac{N}{K}\right)^2}{2}\\ \mathcal{O}_{\text{Worst Time}}(N, K) &= \frac{N^4}{K^2} - N^2. \end{aligned} \tag{19}$$

The average, best, and worst case storage requirements are given by

$$\begin{aligned}\mathcal{O}_{\text{Avg Storage}}(N, K) &= \frac{N^2+K^2}{2}\\ \mathcal{O}_{\text{Best Storage}}(N, K) &= K^2\\ \mathcal{O}_{\text{Worst Storage}}(N, K) &= N^2. \end{aligned} \tag{20}$$

TABLE II
COMPARATIVE PERFORMANCE OF THE PROPOSED ALGORITHM

	Compression Efficiency											
Image #	H	A	LZ	LZ77	J1	J2	J3	J4	J5	J6	J7	P
1) Man	9.12	14.05	23.49	28.12	47.22	48.52	40.66	56.16	53.87	54.09	51.02	54.38
2) Planets	65.70	69.26	72.21	76.7	62.63	69.97	66.97	70.29	70.28	69.99	69.37	80.16
3) Challenger	15.48	20.98	24.38	29.8	33.46	39.87	30.88	37.97	36.53	39.18	37.40	40.07
4) House	9.78	16.34	19.32	30.0	39.52	30.43	26.39	39.44	40.84	35.82	34. 98	44.58
5) Shuttle	16.19	21.34	27.04	30.5	44.86	40.47	38.37	41.64	45.22	42.38	44. 49	45.42
6) F16	18.68	25.55	33.82	38.4	50.48	47.86	44.87	49.20	51.59	50.11	51.38	52.51
7) Coral	14.32	19.19	17.10	24.0	29.48	28. 23	24. 55	30. 55	32.48	31.81	32. 84	32.58
8) Bridge	3.07	7.05	36.33	50.10	40.52	37.10	23.31	52. 93	38.12	35.37	29.45	40.54
9) Lenna	6.62	9.4	13.62	16.4	36.84	44.45	34.46	43.74	42.87	46.18	43.56	44.71
10) Pentagon	16.55	17.7	10.99	18.3	28.52	28.57	25.35	27.37	29.98	29.92	32.00	30.99
11) Renault	12.49	21.97	34.25	36.9	52.22	51.73	48.77	52.52	54.50	54.16	55.58	54.36
12) Sandwich	22.30	32.08	27.92	32.2	41.11	38.92	38.01	37.38	40.65	39.20	42.40	41.50
13) Truckrun	31.18	39.49	61.34	60.6	67.13	61.84	59.50	69.03	69.60	67.01	67.87	69.39
14) Rocks	30.87	35.79	34.31	38.1	40.11	36.07	33.06	38.56	41.48	39.70	40.91	44.06

H= The Huffman (Unix compact) scheme, A= The arithmetic scheme, LZ = The Lempel-Ziv (Unix Compress) scheme, The LZ77(Unix GZIP) scheme, Jx = The JPEG scheme with prediction method x, P = The proposed scheme.

Proof: We first prove the results for the characteristic **smoothness**.

$$\begin{aligned}\mathcal{O}_{\text{Time}}(N, K) &= \sum_{i=1}^{(N/K)^2} (K^2 - 1)\\ &= (K^2 - 1)\left(\frac{N}{K}\right)^2\\ &= \frac{K^2 - 1}{K^2} N^2\\ &= N^2 \quad (\text{assuming} \quad K \ll N). \end{aligned} \tag{21}$$

The amount of space required by the local method is clearly given by $\mathcal{O}_{\text{Space}}(N, K) = K^2$, since at all times a single block is tested for compression.

For the *similarity* characteristic, we can derive the following relationship:

$$\mathcal{O}_{\text{Time}}(N, K) = \sum_{i=0}^{(N/K)^2-1} BT_o(N, K) \tag{22}$$

in which BT_o represents the number of block matching steps required during compression. For each of the three complexity cases, BT_o is given by

$$\begin{aligned} BT_{\text{Avg}}(N, K) &= \sum_{j=1}^{i} \frac{\sum_{l=1}^{K^2} l}{K^2}\\ &= \frac{K^2 + 1}{2} i\\ BT_{\text{Worst}}(N, K) &= \sum_{j=1}^{i} K^2\\ &= K^2 i\\ BT_{\text{Best}}(N, K) &= \sum_{j=1}^{i} 1\\ &= i. \end{aligned} \tag{23}$$

Hence, the time complexity for average, worst, and best case are as follows:

1) *Average Time Case:*

$$\begin{aligned}\mathcal{O}_{\text{Avg Time}}(N, K) &= \sum_{i=0}^{(N/K)^2-1} \frac{(K^2 + 1)i}{2}\\ &= \frac{K^2 + 1}{2}\left\{\frac{\left[\left(\frac{N}{K}\right)^2 - 1\right]\left(\frac{N}{K}\right)^2}{2}\right\}\\ &= \frac{K^2 + 1}{4}\left\{\left(\frac{N}{K}\right)^4 - \left(\frac{N}{K}\right)^2\right\}. \end{aligned} \tag{24}$$

2) *Worst Time Case:*

$$\begin{aligned}\mathcal{O}_{\text{Worst Time}}(N, K) &= \sum_{i=0}^{(N/K)^2-1} K^2 i\\ &= K^2\left\{\left(\frac{N}{K}\right)^4 - \left(\frac{N}{K}\right)^2\right\}\\ &= \frac{N^4}{K^2} - N^2. \end{aligned} \tag{25}$$

3) *Best Time Case:*

$$\begin{aligned}\mathcal{O}_{\text{Best Time}}(N, K) &= \sum_{i=0}^{(N/K)^2-1} i\\ &= \frac{\left[\left(\frac{N}{K}\right)^2 - 1\right]\left(\frac{N}{K}\right)^2}{2}\\ &= \frac{\left(\frac{N}{K}\right)^4 - \left(\frac{N}{K}\right)^2}{2}. \end{aligned} \tag{26}$$

Finally, the results for the space complexity for the **similarity** characteristic are straight forward. In the worst case, the whole image needs to be stored in form of $(N/K)^2$ K-blocks,

TABLE III
DISTRIBUTION OF CHARACTERISTICS

Categ-gory → Image ↓	0	1	2	3	4	5	6 to 26	27
1. Man	0	1	548	0	168	1432	13710	18
2. Planets	707	176	558	1	5	85	3373	316
3. Challenger	0	27	777	0	2	1412	13981	98
4. House	75	63	763	0	124	931	12403	339
5. Shuttle	1	24	271	0	1	617	15277	103
6. F16	13	23	490	5	24	2387	12926	105
7. Coral	0	18	176	0	6	510	15344	258
8. Bridge	3	433	13275	2	208	8848	34996	5773
9. Lenna	0	0	441	0	2	993	63929	165
10. Pentagon	0	0	16	0	0	55	63787	1678
11. Renault	0	0	1036	0	6	7795	56663	18
12. Sandwich	64	0	249	0	2	649	63599	7
13. Truckrun	253	157	5325	20	981	15651	35628	12
14. Rocks	446	134	536	0	2	93	57216	11

whereas in the best case only 2 K-blocks need to be held in memory. On the average

$$\mathcal{O}_{\text{Avg Storage}}(N, K) = K^2 \frac{\sum_{i=1}^{(N/K)^2} i}{\left(\frac{N}{K}\right)^2} = \frac{N^2 + K^2}{2}.$$

□

V. CONCLUSIONS

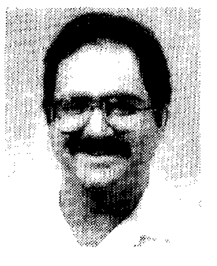
In this paper, two of the most common image characteristics have been analyzed with a goal to develop an algorithm that works well on most images. Based on this analysis, a lossless compression algorithm is presented that utilizes both local and global properties of images. The performance of the proposed scheme is compared with the Huffman, the arithmetic, the Lempel–Ziv and the JPEG lossless schemes.

ACKNOWLEDGMENT

The authors thank the PVRG group at Stanford University for making the JPEG software available in the public domain.

REFERENCES

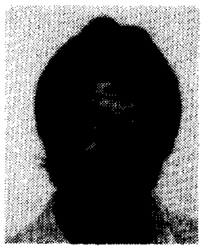
[1] R. M. Fano, *Transmission of Information.* Cambridge, MA: MIT Press, 1949.
[2] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication.* Urbana, IL: Univ. Illinois Press, 1949.
[3] A. S. Fraenkwl and S. T. Klein, "Robust universal complete codes as alternatives to Huffman codes," Dept. of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel, Tech. Rep. CS85-16, 1985.
[4] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE,* vol. 40, pp. 1098–1101, 1952.
[5] S. Golomb, "Run-length encodings," *IEEE Trans. Inform. Theory,* vol. IT-12, pp. 399–401, 1966.
[6] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inform. Theory,* vol. IT-21, pp. 194–203, 1975.
[7] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory,* vol. IT-23, pp. 337–343, 1977.
[8] ——, "Compression of individual sequences via variable rate coding" *IEEE Trans. Inform. Theory,* vol. IT-24, pp. 530–536, 1978.
[9] W. Chen and C. Smith, "Adaptive coding of monochrome and color images," *IEEE Trans. Commun.,* vol. COM-25, pp. 1285–1291, 1977.
[10] R. Gallager, "Variation on a theme by Huffman," *IEEE Trans. Inform. Theory,* vol. IT-24, pp. 668–674, 1978.
[11] A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 157–166, 1982.
[12] J. Vaisey and A. Gersho, "Image vaiiable block size segmentation," *IEEE Trans. Signal Processing,* vol. 40, pp. 2040–2060, 1992.
[13] G. Langdon, "An introduction to arithmetic coding," *IBM J. Res. Dev.,* vol. 28, no. 2, pp. 135–139, 1984.
[14] R. Clarke, *Transform Coding of Images.* New York: Academic, 1985.
[15] B. Ramamurthy and A. Gersho, "Classified vector quantization of images," *IEEE Trans. Commun.,* vol. COM-34, pp. 1105–1115, 1986.
[16] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *CACM,* vol. 30, no. 6, pp. 520–540, 1987.
[17] N. Nasrabadi and R. King, "Image coding using vector quantization a review," *IEEE Trans. Commun.,* vol. 36, pp. 957–971, 1988.
[18] G. K. Wallace, "The JPEG still picture compression standard," *CACM,* vol. 34, no. 4, pp. 30–44, 1991.
[19] J. L. Green, "Space data management at the NSSDC applications for data compression," in *Proc. Sci. Data Compression Workshop,* Snowbird, UT, 1988.
[20] R. B. Miller *et al.,* "Science data management subpanel report," in *Proc. Sci. Data Compression Workshop,* Snowbird, UT, 1988.
[21] J. A. Storer, *Image and Text Compression.* Boston, MA: Kluwer, 1992.
[22] A. Habibi, "Comparison of the nth-order DPCM encoder with linear transformations and block quantization techniques," *IEEE Trans. Commun.,* vol. COM-19, pp. 948–956, 1971.
[23] S. K. Goyal and O'Neal, Jr., "Entropy coded differential pulse code modulation for television," *IEEE Trans. Commun.,* vol. COM-23, pp. 660–666, 1975.
[24] H. Kobayashi and L. R. Bahl, "Image data compression by predictive coding I prediction algorithms, and II encoding algorithms," *IBM J. Res. Dev.,* vol. 18, no. 2, pp. 164–179, 1974.
[25] A. K. Anil, "A fast Karhunen–Loeve transform for a class of stochastic processes," *IEEE Trans. Commun.,* vol. COM-24, pp. 1023–1029, 1976.
[26] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing.* New York: Springer-Verlag, 1975.
[27] H. C. Andrews and W. K. Pratt, "Transform image coding," in *Proc. Computer Processing Communications.* New York: Polytechnic Press, 1969, pp. 63–84.
[28] P. G. Howard and J. S. Vitter, "Fast and efficient lossless image compression," in *Proc. Data Compression,* 1993, pp. 351–360.
[29] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard.* New York: Van Nostrand Reinhold, 1993.



N. Ranganathan (S'81–M'83–SM'92) was born in Tiruvaiyaru, India, in 1961. He received the B.E. (Honors) degree in electrical and electronics engineering from the Regional Engineering College, Tiruchirapalli, University of Madras, India, in 1983 and the Ph.D. degree in computer science from the University of Central Florida, Orlando, in 1988.

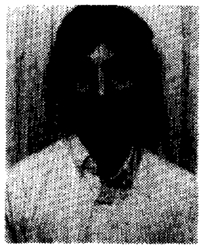
He is currently an Associate Professor in the Department of Computer Science and Engineering and the Center for Microelectronics Research at the University of South Florida, Tampa. His teaching and research interests include VLSI design and hardware algorithms, computer architecture, and parallel processing. He is currently involved in the design and implementation of VLSI architectures for computer vision, image processing, pattern recognition, databases, data compression, and signal processing applications.

Dr. Ranganathan is a member of IEEE Computer Society, the IEEE Computer Society Technical Committee on VLSI, the ACM, and the VLSI Society of India. He served as the Program Co-Chair for VLSI Design '94 and the General Co-Chair for VLSI Design '95. He is also on the Program Committees of ICCD, ICPP, IPPS, SPDP, and ICHPC during 1995. Currently, he serves as an Associate Editor for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *Pattern Recognition* and *VLSI Design.* He was guest editor of a special issue of *International Journal of Pattern Recognition and Artificial Intelligence* (IJPRAI), published in June 1995. He is the editor of a two-volume series on VLSI Algorithms and Architectures published by IEEE CS Press in June 1993.



Steve G. Romaniuk received B.S., M.S., and Ph.D. degrees from the University of South Florida in 1988, 1989, and 1991, respectively.

He was a Teaching Fellow at the National University of Singapore from 1992–1994.



Kameswara Rao Namuduri received the B.Tech. degree from Osmania University, India, the M.Tech. degree from University of Hyderabad, India, and the Ph.D. degree from the University of South Florida in 1984, 1986, and 1992, respectively.

He worked as a Research and Development Engineer in Center for Development of Telematics, Bangalore, India from 1986–1988. Currently, he is working as a System Analyst developing cellular fraud detection products at GTE, Tampa. His research interests include cellular telecommunications and mathematical models from computer vision.