Efficient Algorithms for Lossless Compression of 2D/3D Images

F. Chen, S. Sahni, and B. C. Vemuri
CISE Department, University of Florida, Gainesville, FL 32611
{sahni, vemuri}@cise.ufl.edu

Abstract

We propose enhancements to the 2D lossless image compression method embodied in CALIC. The enhanced version of CALIC obtained better compression than obtained by CALIC on 37 of the 45 images in our benchmark suite. The two methods tied on 6 of the remaining 8 images. This benchmark suite includes medical, natural, and man-made images. We also propose a lossless compression method for 3D images. Our method employs motion estimation and obtained better compression than competing wavelet-based lossless compression methods on all 8 3D medical images in our benchmark suite.

Keywords

Lossless Compression, 2D and 3D Images, CALIC, context-based adaptive coding.

EDICS Categories: 1-MODL and 1-OTHA

Contact Author: Sartaj Sahni (phone: 352-392-1527, fax: 352-392-1220)

This research was support, in part, by the National Institute of Health under grant R01LM05944-03.

I. Introduction

Image compression plays a very important role in applications like tele-video-conferencing, remote sensing, document and medical imaging, and facsimile transmission, which depend on the efficient manipulation, storage, and transmission of binary, gray scale, and color images. Image compression techniques may be classified as either lossy or lossless. Lossy compression methods are able to obtain high compression ratios (size of original image / size of compressed image), but the original image can be reconstructed only approximately from the compressed image. Lossless compression methods obtain much lower compression ratios than obtained by lossy compression methods. However, from the compressed image, we can recover the exact original image. In this paper, we are concerned solely with lossless compression of 2D and 3D images.

Image compression is done by reducing the statistical and spatial redundancy that is present in an uncompressed image. Statistical redundancy results from the fact that different symbols (e.g., gray values) occur with different frequency. This kind of redundancy is handled using coding methods such as Huffman coding [13], arithmetic coding [5], variable bit-length coding [3], and LZW coding [14], [15]. In Huffman coding, for example, frequently occurring symbols are coded using codes that have a smaller number of bits than the codes used for less frequently occurring symbols.

Naturally occurring images have coherence, smoothness, and correlation, which cause spatial redundancy. There are mainly two approaches to reduce the spatial redundancy (*i.e.*, to decorrelate the image). The first is to apply a spatial method such as prediction, interpolation and segmentation on the image. The second approach is to use a transform method such as the wavelet transform [11].

A recent survey and evaluation of coding, spatial, and transform methods for 2D image compression appears in [4]. A comparison of lossless compression methods is also presented in [16], [17]. As concluded in [4], the best compression ratios are obtained by the compression system called CALIC [1]. This system, which falls into the catgory of context-based compression systems, employs both spatial and statistical redundancy methods.

In this paper, we propose enahancements to CALIC. Although these enhancements do not affect the runtime of CALIC, they result in a greater amount of compression. We also propose a motion-based lossless compression scheme for 3D images. This scheme utilizes the 2D image registration algorithm of [2]. Although we also generalize CALIC so that 3D images may be compressed, this generalization does not result in better compression than obtained by our motion-based compression method.

We begin, in Section II with a description of the images used in the experimental evaluation of our proposed compression methods. In Section III we describe the compression algorithm CALIC for later reference and comparison. Our enhancements to CALIC are developed in Section IV. The enhanced version of CALIC is called ECALIC. In section IV, we also present experimental results using the benchmark image suite of Section II. These results show that ECALIC outperforms the original CALIC on 37 of our 45 benchmark images. The two methods are tied on 6 of the remaining 8 images. Our motion-based and contex-based 3D image compression methods are described in Section V. Experimental results are also presented in Section V. These results show that our motion-based 3D compression method outperforms competing methods on 7 of the 8 images in our 3D bechmark suite and ties for first place with one of the competing methods on the 8th image. The paper concludes in Section VI.

II. Benchmark Image Suites

For the experimental evaluation of ECALIC, we use a benchmark suite of 45 2D images. This suite includes the 11 ISO test images shown in Fig. 1, the 16 medical images of Fig. 2, the 9 NASA images of Fig. 3, and the 9 images shown in Fig. 4. These images are described below.

- Among the ISO test images, the images *lenna, man, chall, coral* and *shuttle* are natural gray level images. The image *sphere* is a computer generated image. All images have a resolution of 8 bits/pixel. The images *lenna, man, chal, coral, shuttle*, and *sphere* are 256×256 pixels in size. The size of the remaining images varies. Note that *cmpnd1* is an image with a combination of text and grayscale patterns.
- *heart* is a 256×256 8 bits/pixel image. The images *brain1, brain2, brain3, slice10, slice45, slice90* and *head* are slices from different MRI volume image data sets of the brain. All are 256×256 pixels in size and have a resolution of 8 bits/pixel.
- The images *skull, wrist, carotid, Aperts* are slices from different MR volume image data sets and *liver1, liver2, sag, ped_chest* are from different CT volume image datasets. All are 256×256 pixels in size and have a resolution of 8 bits/pixel. These images were obtained from Washington University, St. Louis, Medical image library, Mar. 1995, `ftp://carlos.wustl.edu/dist/imagelib`.

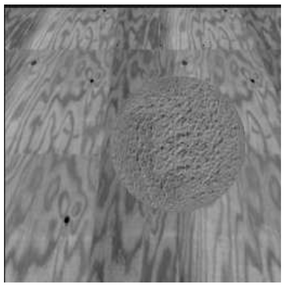
(a) lenna

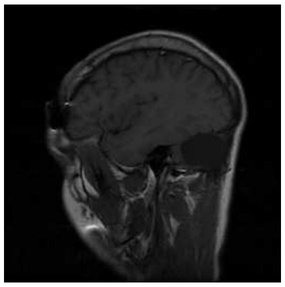
(b) man

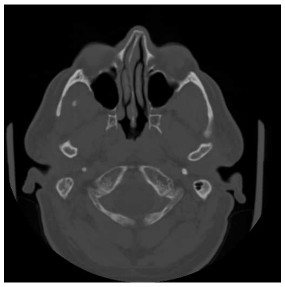
(c) chal

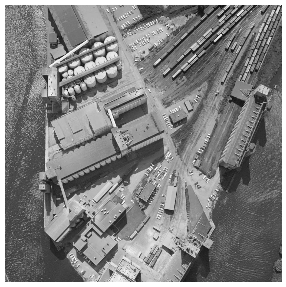
(d) coral

(e) shuttle

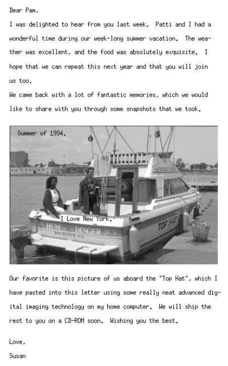

(f) sphere


(g) mri


(h) ct


(i) air1

(j) finger


Dear Pam,

I was delighted to hear from you last week. Patti and I had a wonderful time during our week-long summer vacation. The weather was excellent, and the food was absolutely exquisite. I hope that we can repeat this next year and that you will join us too.

We came back with a lot of fantastic memories, which we would like to share with you through some snapshots that we took.

Summer of 1994.

I Love New York.

Our favorite is this picture of us aboard the "Top Hat", which I have pasted into this letter using some really neat advanced digital imaging technology on my home computer. We will ship the rest to you on a CD-ROM soon. Wishing you the best.

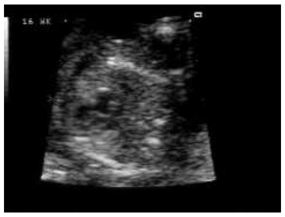
Love,

Susan

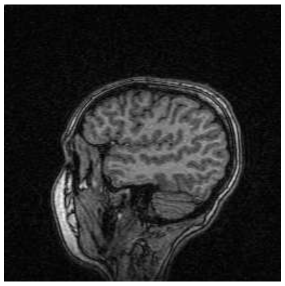
(k) cmpnd1

Fig. 1. ISO Images

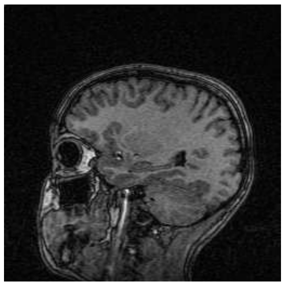
- The images of Fig. 3 are from the NASA image research database. All have a resolution of 8 bits/pixel. They are of varying size and are mostly rectangular in shape.
- The images *mandrill, fprint, pepper, country, city, animals, miranda, w6* and *w7* are a mixed bag of images containing various natural scenes, finger print images, and two ocean images *w6* amd *w7*. These images have a resolution of 8 bits/pixel and are of varying size.



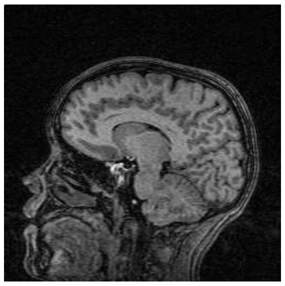
(a) heart



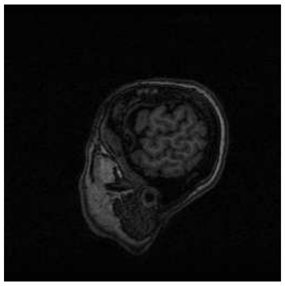
(b) brain1



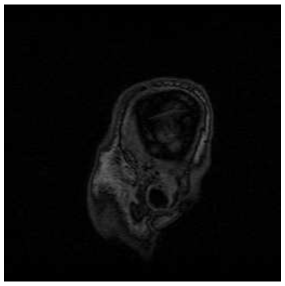
(c) brain2



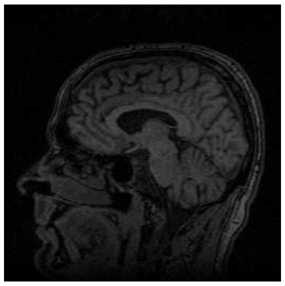
(d) brain3



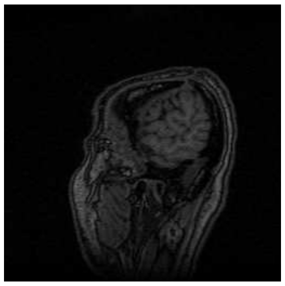
(e) head



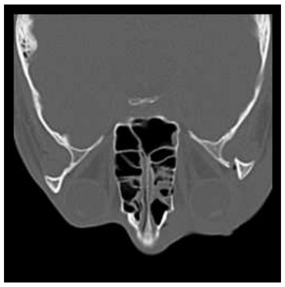
(f) slice0



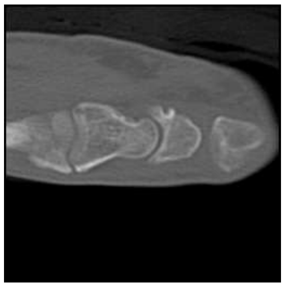
(g) slice45



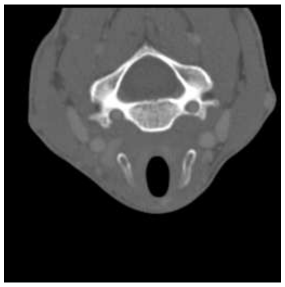
(h) slice90



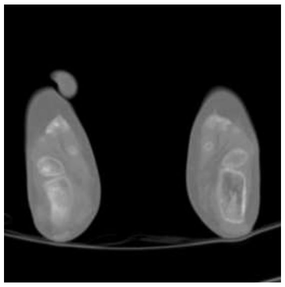
(i) skull



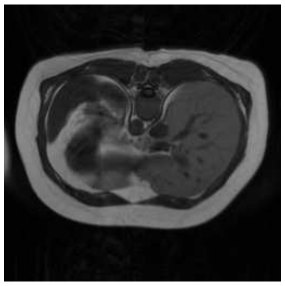
(j) wrist



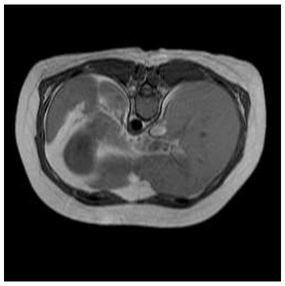
(k) carotid



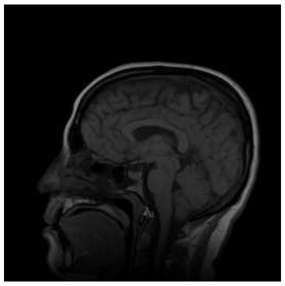
(l) aperts



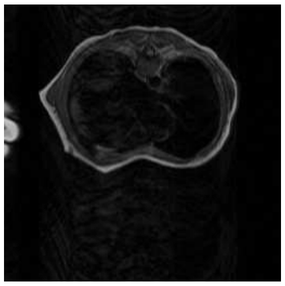
(m) liver1



(n) liver2



(o) sag



(p) ped_chest

Fig. 2. Medical Images

For the evaluation of the 3D compression methods, we use the suite of 8 medical images used in [24]. These images have a resolution of 8 bits/pixel and are composed of a varying number of slices (between 16 and 192). Each slice is a 256×256 image.

III. CALIC: A Context Based Adaptive Lossless Image Coder

CALIC is a context based adaptive lossless image coder that was developed by Wu and Memon [1]. Basically, it is a lossy+residual approach. In the first stage, the algorithm uses a gradient-based non-linear prediction to get a lossy image and a residual image. In the second stage, it uses arithmetic coding to encode the residuals based on the conditional probability of the symbols in different contexts. CALIC also has a mechanism to automatically trigger a binary mode which is used to code uniform and/or binary subregions of the image. CALIC is currently the best lossless image compression scheme reported in the literature. In the following we describe the main stages of the CALIC algorithm.

- **Gradient-based Prediction**

  CALIC scans a 2D image in row-major order and uses a nonlinear predictor to predict the gray value of the next pixel in the same order. The predictor adapts itself to the image gradient near the predicted pixel, thereby achieving good prediction even at sharp edges.

  Let $I[i,j]$ be the gray value of image pixel $[i,j]$. The horizontal and vertical gradients at the current pixel x (see Fig. 5) are estimated using the values of pixels in the neighborhood of x. Notice that all labeled pixels of Fig. 5 precede pixel x in row-major order. The horizontal, d_h, and vertical, d_v, gradients at pixel x are estimated as follows:

$$d_h = |w - ww| + |n - nw| + |ne - n|$$

$$d_v = |w - nw| + |n - nn| + |ne - nne|$$

  Where: $n = I[i-1,j]$, $w = I[i,j-1]$, $ne = I[i-1,j+1]$, $nne = I[i-2,j+1]$, $nw = I[i-1,j-1]$, $nn = I[i-2,j]$, $ww = I[i,j-2]$.

  Based on d_h and d_v, a gradient-adjusted prediction is defined as:

$$\dot{I}[i,j] = \begin{cases} w & (d_v - d_h > 80)\{sharp\ horizontal\ edge\} \\ n & (d_h - d_v > 80)\{sharp\ vertical\ edge\} \\ 3/4w + 1/4n + (ne - nw)/8 & (d_v - d_h > 32)\{horizontal\ edge\} \\ 5/8w + 3/8n + 3(ne - nw)/16 & (d_v - d_h > 8)\{weak\ horizontal\ edge\} \\ 3/4n + 1/4w + (ne - nw)/8 & (d_h - d_v > 32)\{vertical\ edge\} \\ 5/8n + 3/8w + 3(ne - nw)/16 & (d_h - d_v > 8)\{weak\ vertical\ edge\} \\ (w + n)/2 + (ne - nw)/4 & otherwise \end{cases}$$

However, this prediction is not precise enough to completely remove the spatial redundancy in the image. CALIC uses an error energy to estimate the prediction error $e[i,j] = I[i,j] - \dot{I}[i,j]$ for subsequent refinement of the prediction. The error energy is defined as: $\Delta = d_h + d_v + 2 * |ew|$, which is the least square estimator of $e[i,j]$ (ew is the prediction error at the preceding pixel $I[i,j-1]$). By conditioning the error distribution on Δ, the prediction errors are separated into classes of different variances. Thus, entropy coding of errors using estimated conditional probability $p(e|\Delta)$ improves coding efficiency over using $p(e)$. To achieve coding efficiency, Δ is further quantized into L levels (denoted as $Q(\Delta)$).

- **Context Modeling**

Since image gradients are inadequate for characterizing some of the more complex relationships between the predicted pixel $I[i,j]$ and its neighbors, context modeling of the prediction error is used in CALIC. Context modeling of the prediction error $e = I - \dot{I}$ exploits higher-order structures such as texture patterns in the image for further compression gains. The texture pattern is defined as

$$P = \{x_0, ..., x_6, x_7\} = \{n, w, nw, ne, nn, ww, 2n - nn, 2w - ww\}.$$

For space and time efficiency, P is quantized to an 8-bit binary number $B = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$ where:

$$b_k = \begin{cases} 0 & if\ x_k \geq \dot{I}[i,j] \\ 1 & if\ x_k < \dot{I}[i,j] \end{cases} \quad 0 \leq k < 8$$

We can now condition the prediction errors by the context $C(Q(\Delta), B)$. Note that $C(Q(\Delta), B)$ denotes a tuple of two items $(Q(\Delta), B)$. In this paper, we use the notation $C(Q(\Delta), B)$ as was used in [1]. Basically, the definition of the context (tuple $C(Q(\Delta), B)$) is the structure function defined in [20], [19]. In CALIC, the context $C(Q(\Delta), B)$ is not only used to condition or partition the prediction errors in the encoding phase but also used to estimate the prediction errors. The prediction errors in CALIC are estimated as the conditional expectations $E\{e|C(Q(\Delta), B)\}$ using the corresponding sample means

$\overline{e}(Q(\Delta), B)$ for different contexts. Computing $\overline{e}(Q(\Delta), B)$ involves accumulating the errors of processed pixels with the same context and maintaining a count of the occurrence of this context. Thus the context model $\overline{e}(Q(\Delta), B)$ is updated on the fly, a process involving the training of the context model to obtain accurate estimates of the prediction errors under the context $C(Q(\Delta), B)$. After the error estimation, error feedback can be applied on the prediction error: $\ddot{I} = \dot{I} + E\{e|C(Q(\Delta), B))\}$ to get a smaller error $e' = I - \ddot{I}$,

- **Encoding**

The next stage is encoding the errors e'. CALIC uses an adaptive arithmetic coding technique based on the conditional probability of the adjusted errors: $p(e|Q(\Delta))$. The errors are classified into several classes according to $Q(\Delta)$ and each class forms a model in the arithmetic coding as described in Section III.

If we examine the histograms of the errors in each class, for example Fig. 6, we see that large errors occur with diminishing frequency, but still occupy positions in the symbol table. This not only wastes memory space for storing frequency tables in arithmetic coding, but also introduces many unnecessary symbols. In the case of arithmetic coding, particularly for sharp conditional error probabilities $p(e|\delta)$ for small $\delta = Q(\Delta)$, we have to assign 1 to the frequency counts of the zero frequency symbols to safeguard the occurrence of an event of small probability. The forced counting of 1 in the error histograms can significantly distort the underlying error statistics, and thus reduce coding efficiency. This problem is called the *zero frequency problem.*

CALIC uses a simple and effective way to solve this problem: it truncates the tails of the error histogram and uses an escape character to code the errors beyond the truncated code range. Specifically, it limits the size of each conditional error histogram to some value N_δ, $1 \leq \delta < L$, such that a large majority of errors to be coded under the coding context δ fall into the range of the N_δ largest entries of the δ'th error histogram. Essentially the symbol N_δ for each coding context δ serves as an escape symbol. Symbols greater than or equal to N_i are encoded using more than one codeword. The value of each N_δ here is crucial because too small a value causes a large number of escape symbols to be coded, whereas too large a value does not overcome the zero frequency problem. CALIC's authors empirically set the value for N_δ, based on experiments with an image set.

- **Binary Mode**

  CALIC can dynamically trigger a binary mode when the neighbors of a pixel contain very few gray levels. In binary mode, CALIC encodes the pixel value instead of prediction errors. During the scanning process, the algorithm checks the neighbors of the pixel x: $\{n, w, nw, ww, nn, ne\}$ (see Fig. 5) and if they contain only two gray values, CALIC goes into binary mode and encodes a 0 or a 1 when x belongs to one of these two gray values. It encodes a 2 as an escape character when x is neither of the two gray values and then switches back to its normal mode to encode the prediction error of x. Binary mode is effective for binary images and very smooth images.

IV. Enhanced CALIC

In this section we propose several enhancements to CALIC. These enhancements result in a consistent improvement in the compression ratio obtained.

A. Adaptive Binary Mode

As noted in Section III, CALIC switches into a binary mode when in a pixel neighborhood where pixels have at most two values. Although there is no compression overhead incurred to switch into the binary mode, an escape symbol is inserted into the compressed file when switching out of binary mode. The number of pixels coded between the time CALIC switches into binary code and the time it switches out of this mode is called the *length of binary mode.* When the length of binary mode is large, the overhead of the escape symbol is compensated for by the savings obtained from being in binary mode. However, when the length of binary mode is small, better compression is obtained by not switching into binary mode. Thus binary mode improves the compression performance only for uniform or nearly uniform images and natural images which are expected to have large binary mode lengths.

CALIC works well for images with large smooth areas because the average length of binary mode is large and CALIC can trade off the overhead of encoding escape symbols. CALIC also works well for many textured images like *lenna*—which have very few nearly uniform areas—for which binary mode is not triggered and hence no overhead is incurred. When an image contains a fast changing textured background (e.g., the image *man* of Fig. 1(b)), or when an image has a lot of noise (e.g., the brain image of Fig. 2(b)), binary mode is

TABLE I
COMPARISON OF CALIC WITH AND WITHOUT BINARY MODE

Image	Bin_mode	avg_length	with binary	without binary
lenna	0.46%	1.65	1.82	1.82
finger	0.002%	1.00	1.47	1.47
cmpnd1	75.1%	132.04	6.23†	4.25
w6	79.5%	16.85	7.60†	7.05
carotid	47.5%	71.10	4.87†	4.73
man	10.1%	5.66	2.78	2.80†
brain1	4.64%	2.43	2.24	2.25†
mri	16.0%	4.80	2.83	2.87†
slice0	12.5%	3.60	2.70	2.73†

triggered thousands of times resulting in very short binary mode lengths. For these images, the overhead of encoding an escape symbol each time we exit binary mode degrades the compression performance of CALIC.

Columns 2 and 3 of Table I give the percentage of pixels that are coded in binary mode and the average length of binary mode for a subset of our benchmark images. Columns 4 and 5 give the compression ratio resulting when CALIC is used with and without binary mode. Best compression ratios are identified with a †. The images *lenna* and *finger* have very few uniform areas, so the overhead of escape symbols does not have a profound effect on the compression ratio. Two thirds of (*cmpnd1*) is binary text which has only two gray values. Therefore, binary mode makes a significant difference on the compression obtained. For the very smooth and clean images *w6*— an ocean image and *carotid*— a medical image, the use of binary mode also results in a significant improvement in the compression ratio. The remaining four images have a lot of noise and fast changing texture. For these images, the use of binary mode results in a reduced compression ratio.

When CALIC is in binary mode, the context model of prediciton errors is not updated. This means that no prediction occurs on binary mode pixels and thus no prediction error is accumulated in the model for the future computation of $\overline{e}(Q(\Delta), B)$. Therefore, for normal images where the average length of binary mode is small, we lose numerous opportunities to train the context model via omission of the prediction step in binary mode. Since the context model can only be updated on the fly, the earlier the model reaches its steady state, the more accurate the subsequent error estimations will be.

The above analysis leads us to introduce a preprocessing step in which we scan the image to decide whether or not to enable binary mode globally. This decision is made by computing the average length of binary mode

TABLE II
COMPARISON OF ALGORITHM WITH DIFFERENT CONTEXTS

Image	$(Q(\Delta, B))$	$(Q(\Delta), B, e_{neighbor})$
man	2.799	2.814†
ct	6.265	6.321†
finger	1.469	1.477†
skull	2.923	2.930†
wrist	4.616	4.629†
aperts	7.188	7.212†
liver2	3.573	3.580†
carotid	4.734	4.744†

and the number of escape symbols that will need to be introduced. If the average length of binary mode is less than 3 or if there are too many escapes ($\geq 2\%$ of the total number of pixels in the image), then binary mode is disabled during compression. If binary mode is enabled, the context models are updated when the binary mode length is less than 5 so as to increase the chances of early training of the context model.

B. Enhanced Error Prediction

In CALIC, the context for predicting the error $e[i, j] = I[i, j] - \dot{I}[I, j]$ is defined as a tuple $C(Q(\Delta), B)$, where Δ is the least square estimator of the prediction errors and B is the texture pattern of the neighbors. Since the errors associated with the neighbors are also good indicators of the current prediction errors, we add one more parameter to the context model, namely, the average of the prediction errors at the neighbors. This average is defined as $e_{neighbor} = |ew + en|/2$, where ew and en are the prediction errors of the neighbors to the west and north of the current pixel. Now the context becomes a triplet, $C(Q(\Delta), B, e_{neighbor})$. $e_{neighbor}$ is quantized into five levels. The cutoffs for the quantizer are set to (3, 8, 15, 45). So, the number of contexts becomes five times those used in CALIC. Table II gives the compression ratios for a subset of our test suite using both the original context scheme of CALIC and our scheme that includes $e_{neighbor}$. We can see that $e_{neighbor}$ captures important neighborhood information and yields better compression performance consistently.

C. Adaptive Histogram Truncation

CALIC codes the final errors $e' = I - \dot{I} - e$ using arithmetic coding. Since large errors occur with small frequency, we have a severe zero frequency problem in the arithmetic coding. This is particularly true for sharp conditional error probabilities $p(e|\delta)$ with small $\delta = Q(\Delta)$. CALIC uses a histogram tail truncation method to reduce the number of symbols in the arithmetic coding. It limits the size of each conditional error histogram to some value N_d, $1 \leq d < L$, such that a majority of the errors to be coded under the coding context $\delta = Q(\Delta)$ fall into the range of the N_d. For any error e' in this coding context that is larger than N_d, a truncation occurs and the symbol N_d is used to encode the escape character which represents the truncation. Following this, the truncated information $e' - (N_d - 1)$ is assimilated into the following class of prediction errors with the coding context $(\delta + 1)$. In CALIC, N_d, $1 \leq d < L$, is fixed and selected empirically.

Instead of using a fixed histogram tail truncation, we propose an adaptive histogram tail truncation that truncates the low frequency tail of the histogram according to the current error histogram. This eliminates the unnecessary zero frequency count of symbols during the arithmetic coding without incurring a large overhead. Theoretically, there is an optimal point where truncation yields the best compression ratio. But, since it is difficult to analyze quantitatively the relationship between the frequency and codeword length of a certain symbol in the adaptive arithmetic coding, we use a simple criteria to obtain a truncation point. Specifically, we first use a two dimensional counter X to count the frequency of an error in a certain context. For example, $X(\delta, val)$ represents the number of errors that have a value val in the context δ. Then, prior to the encoding, we search each histogram from the low frequency end (the tail) to the high frequency end. Whenever we encounter a frequency greater than a threshold T, we stop and set the truncation point. Since the error histograms are usually monotonically decreasing, such a truncation can retain the significant entries of each histogram and cut off the histogram tail with very low frequency entries. Although using such a simple criteria might seem inadequate for optimal truncation, experiments indicate that this criteria is more effective and efficient than other criteria such as the entropy.

Table III gives the compression ratios for three truncation strategies–no trunction, fixed histogram truncation, and adaptive histogram truncation. Adaptive truncation results in compression ratios that are at least as high as those obtained by fixed truncation and no trunction.

D. Experimental Evaluation of Enhanced CALIC

Enhanced CALIC (ECALIC) includes adaptive binary mode, enhanced error prediction, and adaptive histogram truncation. We have compared the compression effectiveness of ECALIC relative to that of competing

TABLE III
COMPARISON OF ALGORITHM WITH FIXED AND ADAPTIVE TRUNCATION

Image	no truncation	fixed truncation	adaptive truncation
lenna	1.80	1.82†	1.82†
man	2.78	2.76	2.80†
chal	2.33	2.33	2.35†
coral	1.91	1.92	1.93†
shuttle	2.21	2.22	2.24†
sphere	1.88	1.91	1.92†
brain	2.24	2.23	2.25†
head	2.29	2.29	2.31†
mri	2.84	2.84	2.87†
ct	6.76	6.90	6.95†
air1	1.47	1.48†	1.48†
finger	1.47	1.46	1.47†
cmpnd1	6.25	6.31	6.33†
w6	7.67	7.54	7.68†
slice45	2.18	2.18	2.19†
skull	2.92	2.94	2.97†
wrist	4.64	4.63	4.72†
carotid	4.80	4.78	4.88†
aperts	7.36	7.29	7.50†
liver1	2.70	2.70	2.73†
liver2	2.56	2.56	3.61†
sag	3.25	3.26	3.30†
ped_chest	2.62	2.61	2.64†

lossless compression methods including JPEG [7], LOCO-I/JPEG-LS [12], S+P [8], [10], [18] and CALIC [1]. JPEG and LOCO-I/JPEG-LS are the ISO 2D image lossless compression standards. S+P is an effective transform domain algorithm based on the principle of the wavelet transform. CALIC is the context-based spatial domain algorithm upon which ECALIC is based. As reported in [4] CALIC gives the best compression of the known lossless compression methods.

For our experiments, each of the cited methods was used on the raw images in our 45 image benchmark suite as well as on a preprocessed version of these images. The preprocessing [3] involved mapping the grayscale values that occur in an image into a contiguous range of integral values. The mapping table is stored along with the compressed image to enable reconstruction of the original image.

Our findings are presented as four tables, one for each of the four image categories in our suite. The best compression result for each image is marked with a † in each table.

TABLE IV

PERFORMANCE OF 2D COMPRESSION SCHEMES ON ISO TEST IMAGES

Image	JPEG		JPEG-LS		S+P		CALIC		ECALIC	
	w	w/o	w	w/o	w	w/o	w	w/o	w	w/o
lenna	1.56	1.56	1.75	1.76	1.76	1.77	1.82	1.83†	1.82	1.82
man	2.02	2.04	2.69	2.71	2.63	2.65	2.74	2.77	2.77	2.80†
chal	1.93	1.59	2.28	1.85	2.21	1.79	2.35†	1.90	2.35†	1.91
coral	1.67	1.49	1.88	1.64	1.87	1.64	1.93†	1.68	1.93†	1.68
shuttle	1.89	1.80	2.19	2.05	2.13	2.01	2.24†	2.12	2.24†	2.11
sphere	1.58	1.59	1.84	1.85	1.84	1.85	1.89	1.89	1.91	1.92†
mri	2.16	2.16	2.67	2.69	2.76	2.76	2.76	2.78	2.87	2.89†
ct	3.46	3.51	6.03	6.05	5.27	5.27	6.69	6.71	6.92	6.95†
air1	1.34	1.34	1.45	1.45	1.44	1.45	1.49†	1.49†	1.47	1.48
finger	1.34	1.34	1.41	1.41	1.45	1.45	1.46	1.47†	1.47†	1.47†
cmpnd1	2.91	2.87	5.76	6.12	3.32	3.33	6.19	6.19	6.27	6.32†

w and *w/o* indicate the algorithm with and without mapping

The performance of the mapping preprocessor is inconsistent; on some images (e.g., the ocean images *w6* and *w7*), the preprocessing doubles the compression ratio obtained by CALIC and increases the ratio for ECALIC 75%. On other images (e.g., *man*), there is a small reduction in the compression ratio. ECALIC and CALIC generally provide higher compression ratio than any of the other schemes. In fact, CALIC and ECALIC were outperformed on only one of our test images (*fprint*). CALIC and ECALIC performed better than S+P and JPEG on all the test images. They performed better than LOCI-I/JPEG-LS on 42 of 45 images. ECALIC did better than CALIC on 37 of our 45 images; CALIC did slightly better than ECALIC on only 2 of our 45 images; the two methods tied on the remaining 6 images.

V. Lossless 3D Image Compression

Several of today's diagnostic imaging techniques, such as computed tomography (CT), magnetic resonance (MR), positron emission tomography (PET), and single photon emission computed tomography (SPECT), produce a three-dimensional volume of the object being imaged, represented by multiple two-dimensional slices. These images may be compressed independently on a slice by slice basis. However, such a two-dimensional approach does not benefit from exploiting the dependencies that exist among all three dimensions. Since the image slices are cross sections that are adjacent to one another, they are partially correlated.

An alternative approach to compress the sequence of slices is to view the slices as a sequence of moving

TABLE V
PERFORMANCE OF 2D COMPRESSION SCHEMES ON MEDICAL IMAGES

Image	JPEG		JPEG-LS		S+P		CALIC		ECALIC	
	w	w/o	w	w/o	w	w/o	w	w/o	w	w/o
heart	2.01	2.03	2.83	2.88	2.66	2.70	3.01	2.96	3.07†	3.02
brain1	1.86	1.49	2.09	1.63	2.16	1.68	2.21	1.71	2.25†	1.71
brain2	1.84	1.50	2.07	1.65	2.14	1.71	2.19	1.73	2.22†	1.73
brain3	1.83	1.44	2.03	1.56	2.11	1.62	2.15	1.64	2.18†	1.64
head	1.92	1.92	2.15	2.16	2.20	2.20	2.26	2.23	2.31†	2.26
slice0	2.16	2.17	2.52	2.52	2.54	2.54	2.65	2.60	2.73†	2.66
slice45	1.79	1.80	2.04	2.05	2.12	2.12	2.16	2.15	2.19†	2.17
slice90	1.92	1.93	2.25	2.25	2.29	2.29	2.37	2.34	2.42†	2.38
skull	2.00	2.01	2.79	2.82	2.57	2.59	2.91	2.94	2.84	2.97†
wrist	2.61	2.63	4.49	4.55	4.04	4.09	4.42	4.49	4.65	4.72†
carotid	2.73	2.76	4.43	4.51	3.95	4.01	4.67	4.75	4.78	4.88†
aperts	3.43	3.47	6.81	6.95	5.79	5.89	6.97	7.13	7.33	7.50†
liver1	2.06	2.07	2.60	2.61	2.64	2.65	2.68	2.70	2.71	2.73†
liver2	2.20	2.22	3.34	3.37	3.27	3.30	3.54	3.58	3.57	3.61†
sag	2.34	2.35	3.14	3.16	3.00	3.02	3.19	3.22	3.27	3.30†
ped_chest	2.09	2.11	2.45	2.47	2.56	2.58	2.58	2.60	2.62	2.64†

w and *w/o* indicate the algorithm with and without mapping

TABLE VI
PERFORMANCE OF 2D COMPRESSION SCHEMES ON NASA IMAGES

Image	JPEG		JPEG-LS		S+P		CALIC		ECALIC	
	w	w/o	w	w/o	w	w/o	w	w/o	w	w/o
EL36	1.52	1.53	1.80	1.81	1.83	1.83	1.86	1.86	1.87†	1.87†
EL92	1.80	1.80	2.33	2.34	2.28	2.29	2.45	2.46	2.46	2.47†
EL93	1.89	1.90	2.55	2.56	2.46	2.47	2.58	2.59	2.59	2.60†
EL94	1.84	1.85	2.46	2.47	2.39	2.40	2.52	2.53	2.54	2.55†
EL107	1.90	1.90	2.52	2.53	2.46	2.47	2.62	2.63†	2.63†	2.63†
EL199	2.10	2.11	3.05	3.06	3.05	3.06	3.12	3.13	3.15	3.16†
EL213	3.23	3.24	4.91	4.94	4.44	4.46	5.49	5.53	5.55	5.59†
EL216	2.76	2.77	3.92	3.95	3.55	3.57	4.16	4.19	4.21	4.24†
EL475	1.95	1.95	2.66	2.67	2.47	2.48	2.77	2.78	2.79	2.80†

w and *w/o* indicate the algorithm with and without mapping

frames. The third dimension can be treated as the time axis and motion analysis techniques from the computer vision literature can be applied to determine the motion between consecutive frames. The motion estimator acts as a predictor for the next slice. The error between the estimation of the next slice and the actual values for the next slice may be viewed as a 2D image, which can be compressed using 2D compression methods.

Yet another approach is to consider the set of slices as a 3D volume and use prediction or 3D frequency transform compression methods. For example, we could use the lossless 3D wavelet compression algorithm of Bilgin

TABLE VII
PERFORMANCE OF 2D COMPRESSION SCHEMES ON OTHER IMAGES

Image	JPEG		JPEG-LS		S+P		CALIC		ECALIC	
	w	w/o	w	w/o	w	w/o	w	w/o	w	w/o
w6	4.63	2.41	6.66	2.69	5.15	1.76	7.45	4.24	7.68†	4.69
w7	4.66	2.41	6.77	2.69	5.21	1.77	7.22	3.83	7.52†	4.22
mandrill	1.15	1.16	1.21	1.21	1.25	1.25	1.25	1.26	1.25	1.26†
fprint	1.72	1.36	2.42†	1.77	1.88	1.39	1.99	1.47	2.02	1.49
peppers	1.53	1.53	1.73	1.73	1.67	1.67	1.76	1.76	1.77†	1.77†
city	1.65	1.46	2.26†	1.90	1.68	1.45	1.93	1.65	1.97	1.70
country	1.79	1.59	2.27†	1.99	1.79	1.58	2.06	1.82	2.09	1.86
animals	1.30	1.18	1.37	1.22	1.34	1.21	1.42	1.27	1.42†	1.27
miranda	1.98	1.98	2.63	2.63	2.65	2.66	2.78	2.79	2.81	2.82†

w and w/o indicate the algorithm with and without mapping

[24]. This algorithm first decomposes the image data into subbands using a 3D integer wavelet transform, it then uses a generalization of the zerotree coding scheme [9] together with context-based adaptive arithmetic coding to encode the subband coefficients. [21], [22], [23] also use wavelets for 3D image compression.

A. Motion Based Compression Algorithm

Prediction using a motion estimator is called motion compensation. This prediction permits extracting the temporal redundancy that exists in a sequence of images through motion estimation. If the motion vector at each pixel location is (u, v), then the motion error is given by $e(x, y) = I(x, y; n) - I(x - u, y - v; n - 1)$, where n is the slice or frame number. The task of the encoder is to encode the motion errors.

Our proposed algorithm comprises the following steps:

1. Motion estimation

The motion from one slice to the next can be represented by a geometric transformation on one slice. A 2D affine transformation is an example of a geometric transformation and it includes rotation, translation and scaling. The affine motion model is defined in the following manner:

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} t_0 & t_1 \\ t_3 & t_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_2 \\ t_5 \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix}$$

where, $T = (t_0, ..., t_5)^T$ performs a global transformation. So, given two 2D images, a motion estimation algorithm will compute a 2D affine transform vector T. To reduce the computational burden, the transform T is computed only at a subset of the image grid called the control point grid. The transformation can then be interpolated at other locations using a B-Spline representation for the control grid. In our motion-based 3D compression algorithm we used the robust and efficient estimator proposed in [2].

2. Compute the motion error

   We use the motion transform vector T to compute

$$E = T(I_{n-1}) - I_n \tag{1}$$

   which is the motion error between the predicted value $T(I_{n-1})$ of slice n and the actual value of slice n. Specifically, the following steps are used:

   (a) Apply the motion estimation algorithm on the control points to get the motion vector (u, v) at the control points,

   (b) Use bilinear interpolation to get the motion vectors at other locations of the image grid.

   (c) Compute the difference (i.e., the motion error) E between the transformed image $\hat{I}_n = T(I_{n-1})$ and the target image I_n.

3. Encode the motion error

   The error is the difference E between two images as defined in (1). The absolute values of the motion errors are expected to be smaller than the intensities of the pixels in a slice. Thus, applying a 2D compression algorithm on the motion errors is expected to yield a smaller compressed file than when a 2D compression method is applied on the original slice. Since the value of the motion errors can be both positive and negative, we make all errors nonnegative by offsetting the errors an amount equal to the most negative error. This offsetting operation does not alter the correlations in the error image.

The first slice of a 3D image is compressed using a 2D compression scheme such as ECALIC (Section IV). The motion estimation scheme is applied to all other slices. The motion vector and motion errors are stored for each of these remaining slices. Since the motion parameters are six floating point numbers, the storage space they require is very small.

Besides providing good overall compression, our motion based scheme requires only enough computer memory to store just three slices. Thus we can handle very large 3D images. The other schemes require the entire 3D image be in memory at the same time. Further, unlike the wavelet-based 3D compression schemes which

TABLE VIII

COMPARISON OF WAVELET AND MOTION BASED METHODS

Image	slices #	wavelet	motion
CT_skull	192	3.981	4.092†
CT_wrist	176	7.022	7.022†
CT_carotid	64	5.743	6.070†
CT_Aperts	96	8.966	9.875†
MR_liver_t1	48	3.624	3.663†
MR_liver_t2el	48	4.823	4.921†
MR_sag_head	16	3.502	4.063†
MR_ped_chest	64	4.267	4.304†

† indicates the best result

require that the number of slices be a power of 2, our motion-based scheme can be applied to images with any number of slices.

Table VIII gives the compression ratios obtained by our motion-based compressor and the algorithm (wavelet)proposed by Bilgin [24]. Our motion-based scheme outperforms the wavelet-based scheme on 7 of the 8 test images used in [24]. On the 8th image, the two methods are tied.

B. 3D Context-Based Compression Algorithm

This algorithm is an extension of our proposed 2D lossless compression algorithm ECALIC. We introduce a third dimension in the prediction and context modeling steps. For each slice k, the algorithm comprises the following steps:

1. Binary mode detection

   In addition to the six neighbors of the current pixel x used in the 2D algorithm, we use three neighbors from the previous slice $k-1$ (see Fig. 7(b)). If all 9 of these neighbors (neighbors are shown in bold fonts in Fig. 7) have only two distinct values, then binary mode is triggered. The operation of binary mode here is the same as in the 2D case.

2. Prediction

   Once again we use a gradient based non-linear prediction. We add d_z as the gradient in the z/temporal direction and the prediction equation becomes:

$$\dot{I}[i,j] = \begin{cases} w_k & (d_v - d_h > 80 \text{ } or \text{ } d_z - d_h > 80)\{sharp\ horizontal\ edge\} \\ n_k & (d_h - d_v > 80 \text{ } or \text{ } d_z - d_v > 80)\{sharp\ vertical\ edge\} \\ x_{k-1} & (d_v - d_z > 80 \text{ } or \text{ } d_h - d_z > 80)\{sharp\ z\ edge\} \\ (Avg + w_k)/2 & (d_v - d_h > 32 \text{ } or \text{ } d_z - d_h > 32)\{horizontal\ edge\} \\ (3 * Avg + w_k)/4 & (d_v - d_h > 8 \text{ } or \text{ } d_z - d_h > 8)\{weak\ horizontal\ edge\} \\ (Avg + n_k)/2 & (d_h - d_v > 32 \text{ } or \text{ } d_z - d_v > 32)\{vertical\ edge\} \\ (3 * Avg + n_k)/4 & (d_h - d_v > 8 \text{ } or \text{ } d_z - d_v > 8)\{weak\ vertical\ edge\} \\ (Avg + x_{k-1})/2 & (d_v - d_z > 32 \text{ } or \text{ } d_h - d_z > 32)\{z\ edge\} \\ (3 * Avg + x_{k-1})/4 & (d_v - d_z > 8 \text{ } or \text{ } d_h - d_z > 8)\{weak\ z\ edge\} \\ Avg & otherwise \end{cases}$$

where

$Avg = (w_k + n_k + x_{k-1})/3 + ((ne_k - nw_k)/4 + (e_{k-1} - w_{k-1})/4 + (n_{k-1} - s_{k-1})/4)/3$

$d_z = |w_k - w_{k-1}| + |n_k - n_{k-1}| + |x_{k-1} - x_{k-2}|$. The pixel labels used in the above equations are shown in Fig. 8.

3. Context modeling

For 3D context modeling, we first define the error energy as:

$$\Delta = (d_h + d_v + d_z + |ew_k| + |en_k| + |ex_{k-1}|)$$

where ew_k and en_k are the errors in positions w and n of slice k and ex_{k-1} is the error in position x of slice $k-1$. To reduce the number of parameters, we still quantize Δ into $L = 8$ levels.

Since there are several slices in a volume data set, we have access to more context (relative to the 2D case) due to the increased neighborhood size in 3D. Thus, in addition to the 8 bits required to code the texture pattern in the 2D case, we add 10 bits to represent the texture pattern in 3D. These 10 bits are obtained from the pixels contained in slice $k-1$ shown in Fig. 8 and a pixel in slice $k-2$ corresponding to the location of x.

4. Entropy coding

We maintain the context models of prediction errors for the entire volume, and they are updated during the prediction process of each slice. However, for reasons of space efficiency, we encode the errors in each slice separately from other slices. Thus the truncation of the histogram and the arithmetic coding of the errors are done at the end of the prediction stage of every slice. Following this, the histograms of each context are reset for the next slice. In this way, the algorithm needs to access at most three slices at one time, thereby reducing the space complexity of the algorithm.

TABLE IX
COMPARISON OF WAVELET AND CONTEXT-BASED METHODS

Image	slices #	wavelet	3D context-based
CT_skull	192	3.981†	3.910
CT_wrist	176	7.022†	6.447
CT_carotid	64	5.743	6.041†
CT_Aperts	96	8.966	9.741†
MR_liver_t1	48	3.624†	3.280
MR_liver_t2el	48	4.823†	4.405
MR_sag_head	16	3.502	4.093†
MR_ped_chest	64	4.267†	4.100

† indicates the best result

Table IX gives the compression ratios obtained by our 3D context-based compression algorithm and the wavelet-based algorithm proposed by Bilgin [24]. The wavelet-based algorithm does better than our context-based algorithm on 5 of the 8 test images.

C. *Evaluation of all 3D Methods*

To complete the evaluation of the 3D methods, we consider the two versions of the wavelet method proposed in [24]. The methods wavelet-based(a) and wavelet-based(b) differ in the block size used. The results for the schemes wavelet-based(a) and wavelet-based(b) are from Bilgin [24]. The method wavelet-based(a) uses a 2-level dyadic decomposition on blocks of 16 slices, while wavelet-based(b) uses a 3-level dyadic decomposition on the entire volume. As pointed out in Bilgin [24], no single transform performs best over the entire data set, therefore, for comparison pusposes, we include the best results from all the transforms considered in [24]. Table X shows that our motion-based scheme gives best compression on 7 of the 8 test images; our context-based scheme yields the best result on the remaining image. The wavelet-based schemes tie the performance of our motion-based scheme on 1 image.

From the space complexity point of view, our context-based algorithm has the advantage that during each step of compression at most three slices of data are brought into memory, whereas, in the wavelet-based algorithm (b), the whole volume needs to be loaded into memory in order to apply the wavelet transform. When the method wavelet-based(a) is used, blocks of 16 slices are read in and compressed. This scheme, however, does not give as much compression as does wavelet-based(b).

TABLE X

Comparison of 3D compression methods

Image	slices #	wavelet based(a)	wavelet based(b)	motion based	context based
CT_skull	192	3.671	3.981	4.092†	3.910
CT_wrist	176	6.522	7.022†	7.022†	6.447
CT_carotid	64	5.497	5.743	6.070†	6.041
CT_Aperts	96	8.489	8.966	9.875†	9.741
MR_liver_t1	48	3.442	3.624	3.663†	3.280
MR_liver_t2el	48	4.568	4.823	4.921†	4.405
MR_sag_head	16	3.644	3.502	4.063	4.093†
MR_ped_chest	64	4.021	4.267	4.304†	4.100

† indicates the best result

VI. CONCLUSION

In this paper, we have proposed enhancements to the 2D lossless compression method CALIC. The enahnced version, ECALIC, outperforms CALIC on 37 of our 45 test images, and ties with CALIC on 6 of the remaining 8 images.

We have also proposed two lossless 3D image compression algorithms, one is motion-based, the other is context-based. The motion-based algorithm outperformed the wavelet-based algorithms of [24] on 7 of the 8 data sets used; it tied on the 8th data set. Besides providing better compression, our motion-based scheme requires less computer memory than required by the competing methods of [24].

References

[1] X. Wu, N. Menon, "CALIC-A context based adaptive lossless image codec," *Proc. of 1996 International Conference on Acoustics, Speech, and Signal Processing*, pp.1890-1893, 1996.

[2] R. Szeliski and J. Coughlan. "Hierarchical spline-based image registration," *IEEE Conf. Conput. Vision Patt. Recog.*, pp. 194-201, Seattle, WA, June 1994.

[3] S. Sahni, B. C. Vemuri, F. Chen, and C. Kapoor. "Variable-Bit-Length Coding: An Effective Coding Method," University of Florida, 1998.

[4] B. C. Vemuri, S. Sahni, et al.. "State of the art lossless image compression algorithms," University of Florida, 1998.

[5] I. H. Witten, R.M. Neal and J. G. Cleary, "Arithmetic coding for data compression," *Commum. ACM*, vol. 30, pp. 520-540, June 1987.

[6] Weidong Kou, *Digital Image Compression - Algorithms and Standards*, Kluwer Academic Publishers, 1995.

[7] Wallace, G. K. "The JPEG still picture compression standard," *Communications of the ACM*, vol 34, pp. 30-44, April 1991.

[8] A. Said, W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. On Circuits And Systems For Video Technology*, vol. 6, No. 3, June, 1996.

[9] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. On Signal Processing*, vol. 41, No. 12, December 1993.

[10] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. Image Process.*, vol. 5, no. 9, Sept. 1996.

[11] A. R. Calderbank, I. Daubechies, W. M. Sweldens, and Boon-Lock Yeo, "Wavelet transforms that map integers to integers," *Department of Mathematics, Princeton University*, August, 1996.

[12] M. J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A low complexity, context-based lossless image compression algorithm," *Proc. of 1996 Data Compression Conference*, pp. 140-149, 1996.

[13] D.Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, Vol 40, pp. 1098-1101, 1952.

[14] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. on Information theory,* Vol. 24, No. 5, pp. 530-536, 1978.

[15] T. Welch, "A technique for high-performance data compression," *IEEE Computer*, June 1994, 8-19.

[16] V. K.Heer and H.E. Reinfelder, "A comparison of reversible methods for data compression," *Proc. SPIE-Med. Imaging IV*, 1990. vol. 1233, pp. 354-365.

[17] G. R. Kuduvalli and R. M. Rangayyan, "Performance analysis of reversible image compression techniques for high-resolution digital teleradiology," *IEEE Trans. Med. Imaging*, vol. 11, pp. 430-445, Sept. 1992

[18] A.Said and W. A. Pearlman, "Reversible image compression via multiresolution representation and predictive coding," *IEEE Trans. Image Processing*, Vol. 2094, pp. 664-674, Nov. 1993.

[19] M. J. Weinberger, J. J. Rissanen, R. B. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Processing*, vol. 5, No. 4, Appril, 1996.

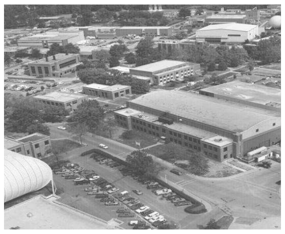
[20] J. Rissanen, G. Gg. Langdon, Jr., "Universal modeling and coding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 12-23, Jan. 1981.

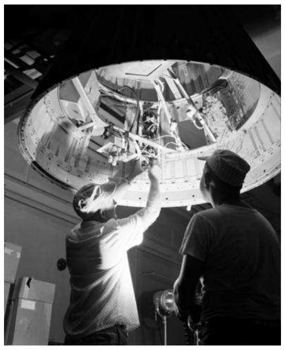
[21] J. Luo, X. Wang, C. W. Chen, and K. J. Parker, "Volumetric medical image compression with three-dimensional wavelet transform and octave zerotree coding," *Proc. of SPIE, Visual Communications and Image Processing '96*, vol. 2727, pp. 579-590.

[22] M. A. Pratt, C. H. Chu, and S. Wong, "Volume compression of MRI data using zerotrees of wavelet coefficients," *Proc. of SPIE, Wavelet Applications in Signal and Image Processing IV*, 1996, vol. 2825, pp. 752-763.

[23] A. Baskurt, H. Benoit-Cattin, and C. Odet, "On a 3-D medical image coding method using a separable 3-D wavelet transform," *Proc. of SPIE, Medical Imaging 95*, 1995, vol. 2431, pp. 184-194.

[24] A. Bilgin, G. Zweig, M. W. Marcellin, "Efficient lossless coding of medical image volumes using reversible integer wavelet transforms," *Proc. 1998 Data Compression Conference*, March 1998, Snowbird, Utah.

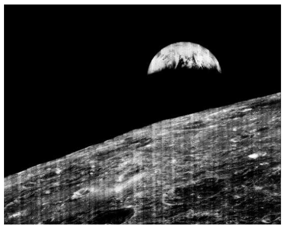

(a) EL36

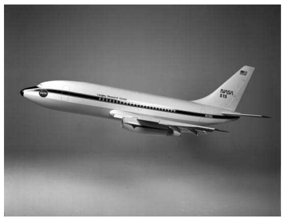

(b) EL92

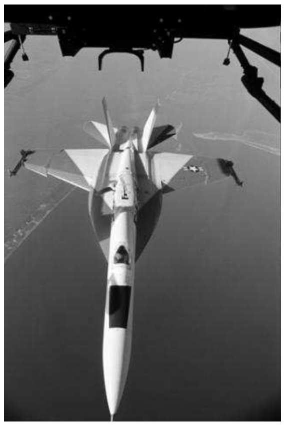
(c) EL93

(d) EL94

(e) EL107


(f) EL199


(g) EL213


(h) EL216

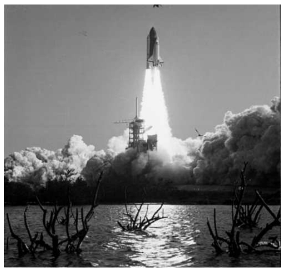
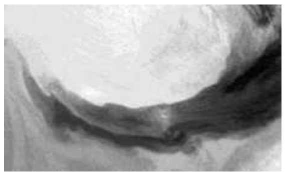
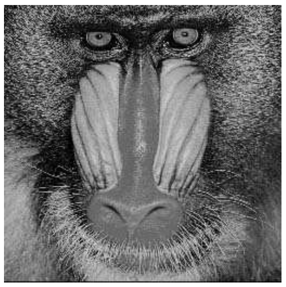

(i) EL475

Fig. 3. NASA Images

(a) w6


(b) w7

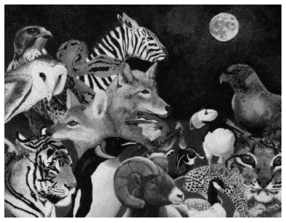

(c) mandrill

(d) fprint

(e) pepper

(f) city

(g) country


(h) animals

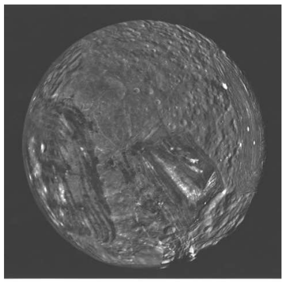

(i) miranda

Fig. 4. Other Images

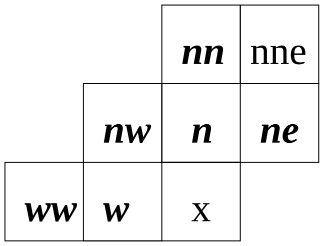
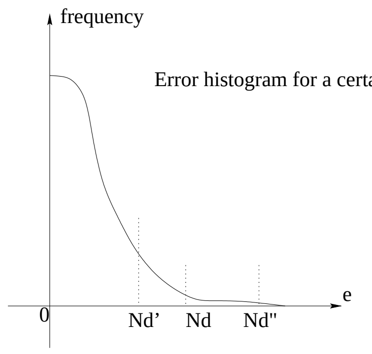
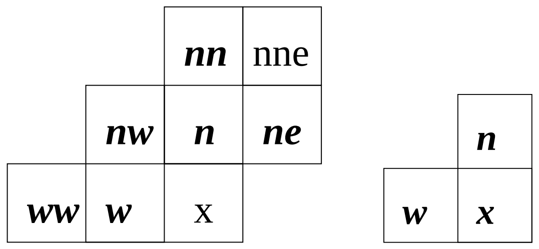


Fig. 5. Neighborhood template for CALIC



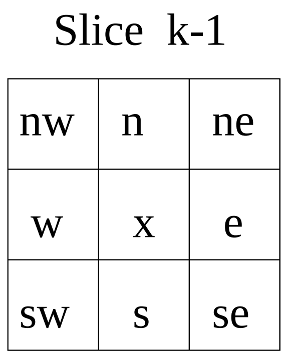
(a)

Fig. 6. Error histogram, d represents δ



(a) (b)

Fig. 7. (a)neighbors in slice k, (b)neighbors in slice $k-1$



(a)

Fig. 8. neighbors in texture pattern in slice k-1