CHAPTER 15

JPEG-LS Lossless and Near Lossless Image Compression

MICHAEL W. HOFFMAN

15.1 LOSSLESS IMAGE COMPRESSION AND JPEG-LS

The original JPEG lossless compression component contained eight options for differentially encoding images. Seven predictors used some subset of neighboring pixels to form the prediction and the eighth used a zero value (no prediction). If an application permitted, the best of these predictors could be used for encoding an image.

The Context Adaptive Lossless Image Compression (CALIC) algorithm was developed in response to a call for proposals for a replacement lossless image compression standard in 1994 [1, 2]. In the initial evaluation of competing proposals, CALIC came in first in six of seven categories. The JPEG-LS standard is derived from a Hewlett-Packard scheme called LOCO-I [3] that is much simpler than CALIC but which incorporates some of the key features of CALIC: context-based adaptive prediction with context-dependent adaptive bias removal.

In the following section the JPEG-LS algorithm is described. This includes an overview and descriptions of the context-based modeling, extension to multicomponent images, error correction and quantization, and entropy coding. The final section presents some example results using the JPEG-LS coding algorithm in both lossless and near lossless modes.

15.2 JPEG-LS

15.2.1 Overview of JPEG-LS

Figure 15.1 presents the overall block diagram of the JPEG-LS encoding algorithm. The main functional blocks consist of context modeling, mode determination (run versus regular), context-based prediction with a context-dependent error correction term, error quantization, and entropy coding. Since JPEG-LS has both lossless and near lossless modes, it is convenient to introduce a

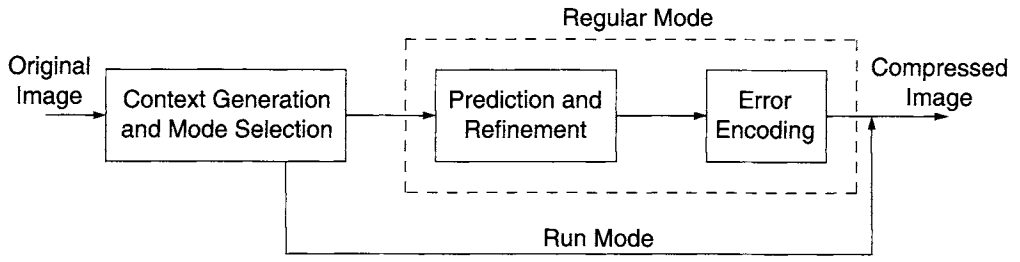

FIGURE 15.1
Overall block diagram of JPEG-LS encoding.

term called NEAR that specifies the maximum absolute pixel error tolerated in the near lossless mode. Note that if NEAR is zero, then lossless encoding is achieved. In the context modeling a set of gradients is determined from reconstructed pixels adjacent to the pixel being encoded. If all of these local differences are less than NEAR, then near lossless coding is possible using the "run mode." In this mode, runs of pixels that are close enough to their neighbors (i.e., within NEAR) are simply encoded as runs rather than as individual pixels. If this is not possible, then the pixels must be encoded individually in the "regular mode."

In the regular mode a context is obtained by classifying the local gradients and merging these classified groups into a reasonable number of contexts. An initial prediction is made based on an edge-detecting predictor that uses the values of the neighboring reconstructed pixels. This prediction is subsequently refined to remove the error bias associated with a given context. Then this error is mapped and quantized, followed by entropy coding.

In the case of multicomponent images, several options are available for encoding the image. First, each component can be encoded completely and separately, with the data for the second following the data for the first, etc. This is referred to as a single-component scan. Second, one of two multicomponent scans can be used, line interleaving or sample interleaving. There are slight differences in the determination of run and regular mode processing for the multicomponent scans (or modes). In the line-interleaved mode a predetermined number of lines from each component are encoded in a fashion very similar to the single-component mode. The data for the given number of lines in each component is encoded with the data for component 2 following that for component 1, and the data for component 3 following that for component 2, etc. In the sample-interleaved mode, pixels from each component are encoded sequentially, from one component to the next before the next pixel in the first component is encoded. This implies that for sample interleaving all of the components in a scan must have the same dimensions. The main difference in coding with sample interleaving is that runs must exist across all components before run mode processing is entered.

The JPEG-LS data stream includes markers, marker segments, and coded data segments. The special character $0xFF$ (i.e., FF in hexadecimal) is used to identify markers—with special restrictions on both coded image data and marker symbols that contain $0xFF$ to allow unambiguous decoding. Since the JPEG-LS encoder uses the reconstructed data available at the decoder, the decoding steps simply mirror those of the encoder to ensure lossless or near lossless reproductions of the original image data.

15.2.2 JPEG-LS Encoding

In this section the various blocks of the JPEG-LS encoding algorithm are discussed. Initially, the context determination and mode are described for single-component images. The extension of these procedures for multicomponent (e.g., color) images is described in the second part of this

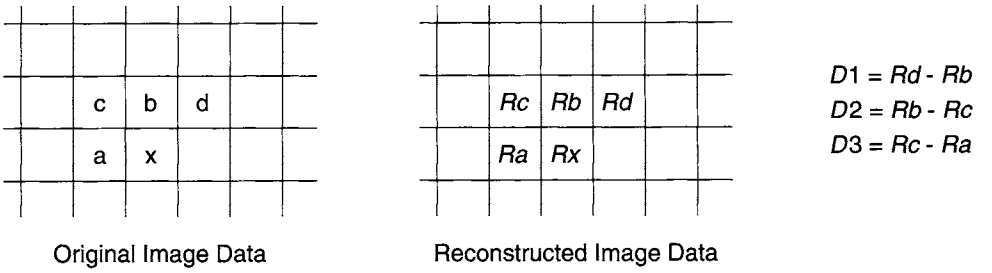

FIGURE 15.2
Prediction neighborhood and gradient estimation based on reconstructed image data.

```
if (Di <= -T3) Qi = -4;
else if (Di <= -T2) Qi = -3;
else if (Di <= -T1) Qi = -2;
else if (Di <= -NEAR) Qi = -1;
else if (Di <= NEAR) Qi = 0;
else if (Di <= T1) Qi = 1;
else if (Di <= T2) Qi = 2;
else if (Di <= T3) Qi = 3;
else Qi = 4;
```

where i=1,2,3 for Di and Qi
T1, T2, and T3 are nonnegative thresholds
NEAR is near lossless threshold (or 0 if lossless)

Context is (Q1,Q2,Q3) {or (−Q1,−Q2,−Q3) if first nonzero element is negative (with SIGN = −1)}

365 possible contexts

Gradient Quantization and Definitions of Context

FIGURE 15.3
Context definition from gradient data.

section. In the third part the prediction error correction, quantization, and reconstruction functions are described. In the final part of this section the entropy coding used in JPEG-LS is described.

15.2.2.1 Context and Mode—Single-Component Images

Figure 15.2 illustrates the prediction neighborhood and context basis of a pixel, x, to be encoded. Note that the encoder uses the reconstructed image data for context determination and prediction so that these processes can be duplicated at the decoder. Also shown in Fig. 15.2 are the formulas used to define the gradients (or pixel differences) $D1$, $D2$, and $D3$. For the first line in an image the reconstructed values Rb, Rc, and Rd are set to 0, as is the reconstructed value Ra for the first pixel. For the first (or last) pixel in subsequent lines, the reconstructed value Ra (or Rd) is set to the value of Rb, while Rc is set to the value assigned to Ra in the previous line.

The context is defined by quantizing the set of gradients. Given a set of gradients, Fig. 15.3 illustrates how the context ($Q1$, $Q2$, $Q3$) is derived. A set of known non-negative thresholds is defined as $T1$, $T2$, and $T3$. These thresholds are used to quantize each of the three gradients to one of nine levels. By using NEAR as either 0 or the near lossless allowable distortion, this quantization of gradients can be applied to either lossless or near lossless modes of operations. Note that if the first non-zero context is negative, then the variable SIGN is defined as -1 and the quantized gradients are stored as the negative of their actual values. Otherwise SIGN is set to 1. This merging of contexts results in 365 rather than $729(= 9^3)$ different contexts. Three hundred

sixty-four of these 365 possible contexts are stored in the variable arrays with indices ranging over [0, . . . , 363]. The all-zero context is used to detect transitions into run mode.

15.2.2.1.1 Run Mode If the three gradients are all equal to 0 (or less than NEAR for near lossless compression), then JPEG-LS enters a run mode in which runs of the same (or nearly the same) value as the previous reconstructed pixel are encoded. Otherwise, "regular mode processing" is used. In run mode processing, as long as the successive image pixels are equal to Ra (or within NEAR of Ra for the near lossless case), a run continues. The run is terminated by either the end of the current image line or a run interruption sample (i.e., one that violates the match condition to Ra defining the run). The coding for the run mode is broken into two sections: run coding and run interruption coding. Run-lengths in powers of 2 are encoded. In a "hit" situation, either the run-length is that power of 2 or an end of line has occurred. In a "miss" situation, the location of the termination of the run is transmitted. Run interruption samples can be one of two classes: those where the interruption sample has neighboring pixels Ra and Rb with values that are more than NEAR apart and those where this absolute difference is less than NEAR. Different predictors and a slightly different updating of context information are used in these cases—with the entropy coding being similar to that used in regular mode processing.

15.2.2.1.2 Regular Mode In regular mode processing, a median "edge-detecting" predictor is used. Figure 15.4 shows the pseudo-code used to determine the value of the prediction Px for pixel x using the reconstructed values Ra, Rb, and Rc. This prediction is then corrected based on the context, as will be discussed shortly.

15.2.2.2 Modifications for Multicomponent Images

When more than a single image component exists, as is the case in color images, two multicomponent scans can be used. The different components may have the same sampling resolution or they may have different resolutions. In the horizontal and vertical directions different sampling factors can be used. For an image with Nf components the sampling factors can be written as $H_0, H_1, \ldots, H_{Nf-1}$ in the horizontal direction and $V_0, V_1, \ldots, V_{Nf-1}$ in the vertical direction. Letting H_{max} and V_{max} be the maximum sampling factors in each direction, then the subsampling performed for the horizontal and vertical directions of component i relative to the highest resolution component can be denoted as H_i/H_{max} and V_i/V_{max}, respectively. Figure 15.5 illustrates an

```
if (Rc >= max(Ra,Rb)) {
  Px = min (Ra,Rb);
} else {
  if (Rc <= min(Ra,Rb)) {
    Px = max(Ra,Rb);
  } else {
    Px = Ra + Rb - Rc;
  }
}
```

Predicted Value for Pixel x

FIGURE 15.4
Edge-detecting predictor used to generate the prediction Px.

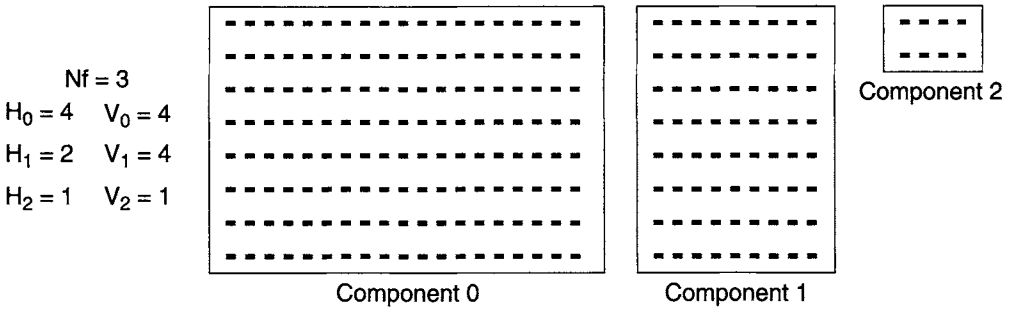

FIGURE 15.5
Example sampling for a three-component image.

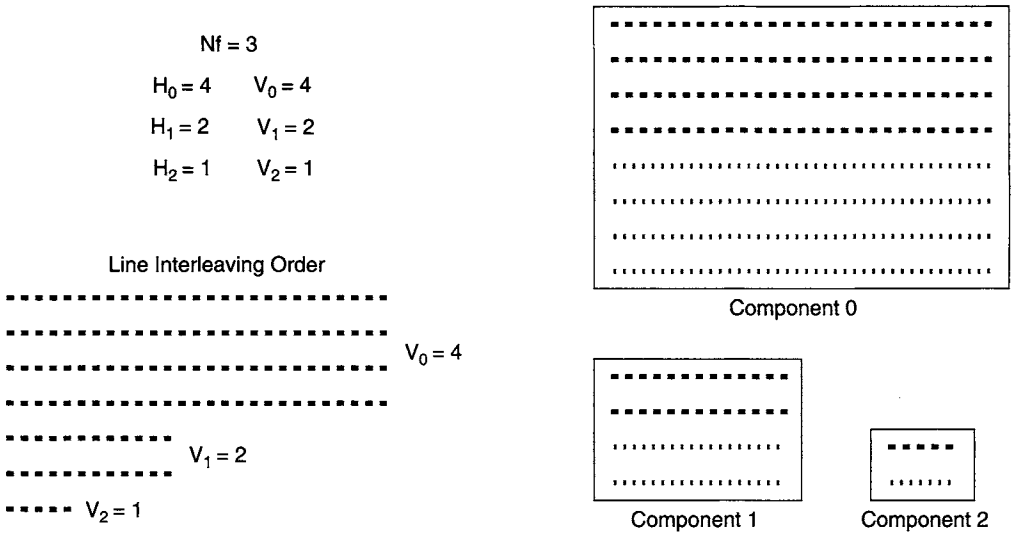

FIGURE 15.6
Example line interleaving for a three-component image.

example multicomponent image that contains three components. The larger the sampling factor, the more lines in that dimension for a particular component. For a red–green–blue color image, these sampling factors may be the same for each of the three components. In a Y-U-V color image, the luminance component, Y, may have higher sampling factors than the two chrominance components, U and V. (This representation exploits the relative spatial sensitivity to chrominance changes with respect to the luminance spatial sensitivity.) There are two ways a multicomponent image can be interleaved: line interleaving and sample interleaving.

15.2.2.2.1 Line Interleaving In line interleaving, the vertical sampling factors for each component determine the number of lines coded for that component before lines are interleaved from the next component. Figure 15.6 illustrates line interleaving for a given multicomponent image example. In the example $V_0 = 4$, four lines are coded from component 0. $V_1 = 2$, so two lines of component 1 are coded (note that these are shorter lines). Finally, $V_2 = 1$ and a single line is coded from component 2. In line interleaving the determination of run versus regular mode processing

is identical to the process used for single-component images. Recall that in single-component images the run versus regular mode processing decision was determined within lines of the image—so the extension to line interleaving is straightforward.

15.2.2.2.2 Sample Interleaving For a sample-interleaved multicomponent image, pixels are interleaved from one component to the next; i.e., one pixel is taken from each component sequentially. For this interleaving mode to make sense, the components must have identical sampling factors (as in red–green–blue color images). The run versus regular mode processing decision is slightly modified to accommodate sample interleaving. For a run to be declared it must exist *across all the image components*. When the run ends in any one of the components, i.e., the difference between the pixels being coded and Ra exceeds NEAR (or zero in lossless coding) in any component, then the run is broken and the encoder returns to regular mode processing. One side effect of this is that the all-zeros context must be included for the predictor since several image components may have a zero gradient but another component's value may preclude the use of run mode processing.

15.2.2.3 Prediction Error Correction, Quantization, and Reconstruction

The edge-detecting predictor used in regular mode processing produces an initial estimate of pixel value x, Px. This estimate is corrected based on the context of the given pixel. A JPEG-LS encoder keeps track of several arrays of variables that are indexed by the context, Q. The arrays include $N[Q]$, $A[Q]$, $B[Q]$, and $C[Q]$. $N[Q]$ is increased by 1 each time the given context Q is used (this is initialized to the value 1). $A[Q]$ is used to track the absolute normalized (by quantizer step size) error to determine the parameters used in entropy coding. $B[Q]$ accumulates the non-normalized pixel error bias (positive or negative) and it is used to increment or decrement $C[Q]$, the error correction term for context Q. These values are updated after the error for a given pixel has been computed. When $N[Q]$ exceeds a RESET value, $A[Q]$, $B[Q]$, and $N[Q]$ are right-shifted by 1 bit to halve their contents and avoid numerical problems. Note that all of these context-indexed arrays use the quantized variables (such as the prediction error) that are available to both the encoder and the decoder.

When the value of $B[Q]$ falls outside of the range $[-N[Q]+1, 0]$ its value is mapped back into this interval. If $B[Q]$ is less than or equal to $-N[Q]$, then $N[Q]$ is added to it and $C[Q]$ is decremented by 1; i.e., the error correction term is reduced. If $B[Q]$ is greater than 0, then $N[Q]$ is subtracted from it and $C[Q]$ is incremented by 1. After these operations $B[Q]$ is hard-limited to the nearest end-point of the $[-N[Q]+1, 0]$ interval. In this way, the bias term continues to push the error correction term up or down depending on how consistently and significantly the error tends to be positive or negative for a short run of data. The maximum change in the value of $C[Q]$ is ± 1 for any one sample error estimate. Figure 15.7 gives the correction to a pixel prediction, Px, for a given context, Q, based on the error bias correction contained in an array $C[Q]$.

The error is either directly coded for lossless compression or quantized to a step size of (2NEAR+1) for near lossless compression. The representation inside the encoder is normalized to an integer label in which the value 1 corresponds to the step size (this is 1 for lossless compression). The encoder requires the reconstructed value that will be produced at the decoder. This is obtained by multiplying the normalized error value by the error SIGN and the step size of the quantizer and adding this to the predicted value (including corrections) produced by both the encoder and the decoder. At this point the error value is mapped to a non-negative number to facilitate entropy coding.

```
if (SIGN == +1) {
  Px = Px + C[Q];
} else {
  Px = Px - C[Q];
}
if (Px > MAX) {
  Px = MAX;
} else if ( Px < 0 ) {
  Px = 0;
}
```

Correction for Pixel x Prediction

FIGURE 15.7
Prediction error correction based on context Q accumulated error bias correction, $C[Q]$. *MAX* is the maximum pixel value.

15.2.2.4 JPEG-LS Entropy Coding

The entropy coding in JPEG-LS can be conveniently separated into regular mode coding of mapped error values and run mode coding of run-lengths and run interruption samples. In this section regular mode coding is discussed first, followed by a discussion of run mode coding.

15.2.2.4.1 Regular Mode Coding Limited-length Golomb codes are used to represent the mapped (non-negative) error values in regular mode processing. The codewords can be thought of as two separate codes: one for the remainder of the error value divided by 2^k and one for the result of the integer division of the error value by 2^k. k is a code parameter that specifies the number of bits used to represent the remainder. The code for the k remainder bits is simply a fixed-length k-bit unsigned binary representation. The code for the integer division result is a unary code that terminates in a 1. For example, to represent 37 with a $k = 3$ code the unary part is given by $37/8 = 4$ zeros and a one (00001) while the remainder is the 3-bit representation of 5 (101). The entire codeword is obtained as 00001101.

Figure 15.8 illustrates the codewords for a $k = 2$ with a maximum length limit of 32 bits. The unary code represents the result of integer division of the mapped errors by 4 ($= 2^2$). To resolve the ambiguity associated with the many-to-one mapping provided by the integer division, the 2-bit unsigned binary code represents the remainder. The use of the unary code results in very large codewords for large error values. The JPEG-LS entropy coding imposes an upper limit on the number of bits that can be used for a codeword. This is done by using an escape code after which the L-bit value of the mapped error is directly represented by an unsigned binary code. L is determined by the number of bits required to represent the quantizer error values. In the example the maximum number of bits in a codeword is 32 and the value of L is 8. An example of a mapped error of 123 is shown. For this mapped error value an escape code of 23 zeros followed by a one indicates the exception and this is followed by the unsigned 8-bit binary representation of $123 - 1$. Since there is never a reason to code a mapped error value of 0 using an escape code, the mapped errors are coded as their value less 1.

15.2.2.4.2 Run Mode Coding When run mode processing is entered, a different form of coding is used. An internal array of values defines a variety of length of runs. Figure 15.9 illustrates an array J that contains 32 values of rk. Potential runs are compared to the current run-length rm.

Prediction Error	Mapped Error	Codewords (2 parts) Unary	Remainder
0	0	1	00
−1	1	1	01
1	2	1	10
−2	3	1	11
2	4	01	00
−3	5	01	01
3	6	01	10
−4	7	01	11
4	8	001	00
−5	9	001	01
5	10	001	10
−6	11	001	11
6	12	0001	00
−7	13	0001	01
7	14	0001	10
−8	15	0001	11
8	16	00001	00
−9	17	00001	01
⋮	⋮	⋮	⋮
Example −62	123	00000000000000000000001 (escape)	
		01111010 (8 bit representation of 123-1)	
		total of 32 bits	

FIGURE 15.8
Example error values, mapped error values, and codewords for JPEG-LS error coding.

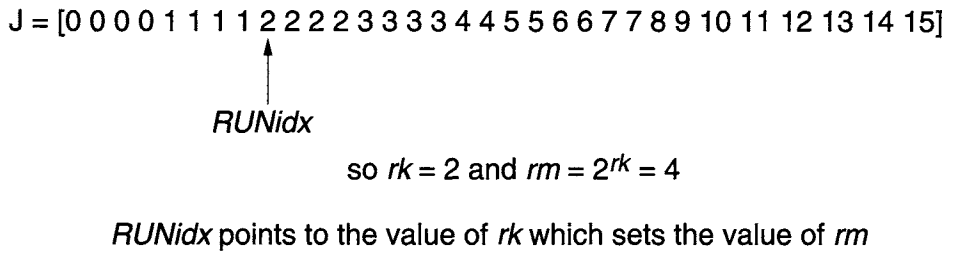

FIGURE 15.9
Run mode coding length determination.

rm is defined as 2 to the power *rk*, which is the array value currently pointed to by the pointer *RUNidx*. In Fig. 15.9, *RUNidx* is 8, which results in an *rk* value of 2 and an *rm* value of 4. If run mode processing is entered, the current value of *rm* is compared to the length of the current run. If the run is equal to or in excess of *rm*, then a hit is declared and a 1 is transmitted to

indicate this hit. At this point the value of *RUNidx* is incremented by 1 and the process continues. If an end of line is reached during a run that has not reached *rm* in length, a 1 is also transmitted; however, the variable *RUNidx* is not changed. If the run ends in a length less than *rm*, then a miss is indicated by the transmission of a 0 bit. After this bit *rk* bits are transmitted to indicate the length of the run (this must be $\leq 2^{rk} - 1$). The value of *RUNidx* after a miss is decreased by 1. The coding of the run interruption sample is similar to regular mode coding except that two separate contexts are reserved for determining the coding used on this sample. In addition, the prediction uses either the pixel *Ra* that started the run or its neighbor *Rb* as the prediction without further correction.

15.2.3 JPEG-LS Decoding

Since the JPEG-LS encoding algorithm uses information available to the decoder, the decoding process recapitulates many of the basic encoder functions. After decoding the entropy data, the decoder either completes the run replication (in run mode) or reconstructs the error values (in regular mode) and adds these to the predicted values (with the same corrections provided at the encoder) to generate the image. Note that the context definitions and updates in the decoder are also identical to those produced in the encoder.

15.3 SUMMARY

Figure 15.10 illustrates near lossless coding for the *sensin* test image (8-bit grayscale). The original image is shown on the top left (NEAR = 0). The reconstructed images for NEAR = 3, 7, and 15 are shown on the top right, bottom left, and bottom right, respectively. Note the degradation in image quality—particularly in flat regions such as the blackboard—as the value of NEAR becomes larger.

The compression attained on the *sensin* image for varying values of NEAR is illustrated in Table 15.1. By removing the restriction of lossless compression and replacing it with a near lossless constraint that the maximum difference allowed in defining a run is 1, the compressed file size drops nearly 40%, a savings of about 1.4 bits per pixel. By increasing NEAR to 3, an additional 0.75 bits per pixel of compression are obtained. As was seen in Fig. 15.10, for larger values of NEAR the additional compression probably is not worth the artifacts introduced by the lossy coding—i.e., it is not near lossless compression, rather it is lossy compression. At these lower rates, better image compression strategies exist [4].

Table 15.1 Compression in Bits per Pixel and Percentage of Original as NEAR Is Varied from 0 (Lossless) to 15 for the 8-Bit Grayscale Image *sensin*

NEAR	Bits/Pixel	% of Original
0	3.704	46.3%
1	2.281	28.5%
3	1.528	19.1%
7	1.093	13.7%
15	0.741	9.3%

FIGURE 15.10
The *sensin* test image (top left), NEAR = 3 (top right), NEAR = 7 (bottom left), NEAR = 15 (bottom right).

15.4 REFERENCES

1. Wu, X., N. D., Memon, and K. Sayood, 1995. A Context Based Adaptive Lossless/Nearly Lossless Coding Scheme for Continuous Tone Images, ISO Working Document, ISO/IEC SC29/WG1/N256.
2. Wu, X., and N. D. Memon, 1996. CALIC—A context based adaptive lossless coding scheme. *IEEE Transactions on Communications*, vol. 45, pp. 437–444, May 1996.
3. Weinberger, M., G. Seroussi, and G. Sapiro, 1996. LOCO-I: A low complexity, context-based, lossless image compression algorithm, In *Proceedings of Data Compression Conference*, Snowbird, UT, March–April 1996.
4. Sayood, K., 2000. *Introduction to Data Compression*, 2nd ed., Morgan-Kaufmann, San Mateo, CA.
5. ITU-T Recommendation T.87. 2000. Information Technology—Lossless and near lossless compression of continuous tone still images, *JPEG-LS Recommendation.*