Application of Grammar-Based Codes for Lossless Compression of Digital Mammograms

Xiaoli Li
Ryerson University

Sridhar Krishnan
Ryerson University

Ngok-Wah Ma
Ryerson University

Application of grammar-based codes for lossless compression of digital mammograms

Xiaoli Li
Sridhar Krishnan
Ngok-Wah Ma
Ryerson University
Department of Electrical and Computer Engineering
Toronto, Ontario M5B 2K3
Canada
E-mail: x4li@ryerson.ca

Abstract. *A newly developed grammar-based lossless source coding theory and its implementation was proposed in 1999 and 2000, respectively, by Yang and Kieffer. The code first transforms the original data sequence into an irreducible context-free grammar, which is then compressed using arithmetic coding. In the study of grammar-based coding for mammography applications, we encountered two issues: processing time and limited number of single-character grammar G variables. For the first issue, we discover a feature that can simplify the matching subsequence search in the irreducible grammar transform process. Using this discovery, an extended grammar code technique is proposed and the processing time of the grammar code can be significantly reduced. For the second issue, we propose to use double-character symbols to increase the number of grammar variables. Under the condition that all the G variables have the same probability of being used, our analysis shows that the double- and single-character approaches have the same compression rates. By using the methods proposed, we show that the grammar code can outperform three other schemes: Lempel-Ziv-Welch (LZW), arithmetic, and Huffman on compression ratio, and has similar error tolerance capabilities as LZW coding under similar circumstances.* © *2006 SPIE and IS&T.* [DOI: 10.1117/1.2178792]

1 Introduction

Breast cancer is the most common cancer among women in Canada. Mammograms, which are x-ray images of the breast, allow better cancer diagnosis. Mammograms are large size images and have scattered correlation details, which means the gray values of pixels within one area progressively increase or decrease. To offer higher details, the ideal resolution of digital mammograms should be up to 4096×4096, 12 bits per pixel. For better archival and communication of such large size images, lossless compression techniques are essential, since loss of any information is usually unacceptable in medical applications.

The amount of work related to the lossless compression of mammograms is very little, and most of them use lossy[1] or near-lossless techniques,[2,3] or lossless encoding of the object or objects of interest within the mammograms to achieve a high compression ratio.[4] Another group in the University of South Florida adapted wavelet-based compression methods for mammograms,[5] and they concluded that wavelet-based methods are promising for visually lossless digital mammogram compression for high compression ratio.

In the near future, it is expected that medical images will become more multimedia in nature. For example, the images may include speech/audio commentary of radiologists, physicians, and other clinical practitioners. In addition, patient information in text form could be encoded in the image. With the ubiquity of the Internet, telemedicine applications such as web-based clinical interpretation of medical images also holds a bright future. The well-known lossless compression schemes of Huffman,[6] Lempel-Ziv-Welch (LZW),[7] and arithmetic[8] coding have been widely investigated and used in a variety of applications including medical image analysis. The performance of these conventional lossless compression techniques is questionable for multimedia/multimodal sources.

In 1999 and 2000, a new universal grammar-based lossless source coding algorithm, called the grammar-based code,[9,10] was developed by Yang and Kieffer and has been adapted for web application. It has been shown that grammar codes can specifically provide good performance on files with multimedia characteristics.[11] It is interesting to see if this lossless compression scheme can be adapted effectively in medical imaging with multimedia nature. This work is a first attempt to investigate the applicability of grammar codes for compression of medical images. The investigation procedure involves the following steps: real-time implementation of the grammar-based code; application to digital mammograms obtained from the Mammographic Image Analysis Society (MIAS)[12]; and comparison of its compression performance and error-tolerance ability with three other commonly used lossless compression algorithms (Huffman, arithmetic, and LZW). We encountered two issues in the investigation: processing time and limited number of single-character grammar variables (G variables). For the first issue, an extended grammar code technique is proposed and the processing time of the grammar code can be significantly reduced. For the second issue, we propose to use double-character symbols to increase the number of grammar variables. Under the condition that all the G variables have the same probability of being used,

Paper 04176RR received Nov. 5, 2004; revised manuscript received Sep. 13, 2005; accepted for publication Sep. 27, 2005; published online Feb. 24, 2006.
1017-9909/2006/15(1)/013021/11/$22.00 © 2006 SPIE and IS&T.

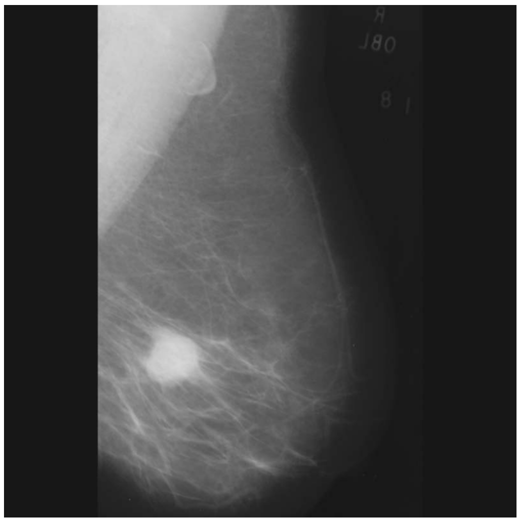


Fig. 1 A mini-MIAS digital mammogram with well-defined/circumscribed masses and malignant tumor.

```
P2
18 5 255
175 177 179 178 175 171 172 173 174 176 176 178 175 175 173 171 175 175
177 180 182 180 175 176 175 175 173 176 178 179 178 176 175 177 178 177
181 181 181 179 179 180 181 178 179 179 178 179 179 176 176 182 182 179
181 180 182 180 180 182 181 180 181 183 181 182 181 178 179 177 181 181
180 181 182 180 183 183 183 182 180 184 183 182 180 180 177 178 178 179
```

Fig. 2 A PGM format sample.

our analysis shows that the double- and single-character approaches have the same compression rates.

This work is organized as follows. The characteristic features of digital mammograms are discussed in Sec. 2. Section 3 presents the theory of the grammar-based code. The implementation, performance results, and analysis of the code and its extended form for compression of mammograms are discussed in Sec. 4. Section 5 mainly analyzes the error tolerance of the grammar-based code. Some conclusions are covered in Sec. 6.

2 Mammograms

MIAS generated a database of digital mammograms for researchers around the world in this field. The original MIAS database was digitized at 50-μm pixel edge and represents each pixel with an eight-bit word. In the free download version of the database, the resolution of images has been reduced to 200-μm pixel edge and clipped so that every image contains 1024×1024 pixels. This shortened database of 30 mammographic images is called the mini-MIAS. The mini-MIAS images contain the significant portion, or region of interest of the mammogram, and is normally used for computer-aided diagnosis of breast cancer. The database uses portable gray map (PGM) as the format of digital mammograms. An example of a digital mammogram with a malignant tumor is shown in Fig. 1. The example of a PGM file cut from an original 1024×1024 mammogram is illustrated in Fig. 2. In each PGM file, the first line specifies the type of encoding, P2, which identifies the PGM grayscale image, stored in ASCII, one value per pixel. The first two numbers in the second line are width and height of the image in pixels. The third number indicates the maximum number of gray-level values represented by integer numbers. The rest of the data in the file are pixels of different gray levels, and their total number is $width \times height$.

3 Grammar-Based Source Coding

Traditional lossless data compression source coding can be divided into two groups, probability based[6,8] and dictionary based.[7] The well-known probability-based lossless source codings are Huffman coding and arithmetic coding and their variants. LZW coding belongs to the dictionary-based group. The new grammar-based code also can be viewed as a dictionary-based code.

The purpose of this section is to briefly review the grammar-based code. For a detailed description, please refer to Ref. 2. The grammar-based code has the structure shown in Fig. 3. The code includes four parts: irreducible grammars, reduction rules, the greedy irreducible grammar transform, and hierarchical compression algorithm.

Let Λ be our source alphabet with cardinality greater than or equal to 2. Let Λ^+ be the set of all finite sequences of positive length from Λ. A sequence from Λ is sometimes called a Λ sequence. Fix a countable set $S=\{s_0,s_1,s_2,\ldots\}$ of symbols, disjointed from Λ. Symbols in S will be called *variables*; symbols in Λ will be called *terminal symbols*. For any $j \geqslant 1$, let $S(j)=\{s_0,s_1,s_2,\ldots,s_{j-1}\}$. A context-free grammar G is a mapping from $S(j)$ to $[S(j)\cap\Lambda]^+$ for some $j \geqslant 1$. The set $S(j)$ will be called the variable set of G or G variables. To describe the mapping explicitly, we write for each $s_i (i<j)$ the relationship $[s_i, G(s_i)]$ as $s_i \rightarrow G(s_i)$, and call it a *production rule*. Let x be a sequence from Λ, which is to be compressed. The *irreducible grammar transform* starts from the grammar G consisting of the only production rule $s_0 \rightarrow x$, where s_0 is the first G variable in the G-variable set, and repeatedly applies reduction rules in some order to reduce G into an irreducible grammar G'. Therefore, the function of reduction rules is to ensure that an existing grammar G is irreducible. The grammar-based code then uses a zero-order arithmetic code to compress the irreducible grammar G' to achieve high compression ratio. The original sequence x can be recovered from the compressed binary codeword through the use of arithmetic decoding, which produces G', and the use of parallel replacement on G' to recover x.

An irreducible grammar G satisfies the following properties.

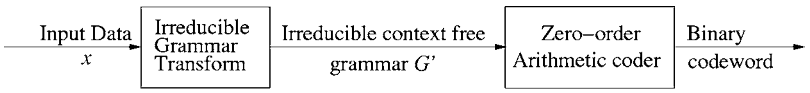


Fig. 3 Structure of the grammar-based code.

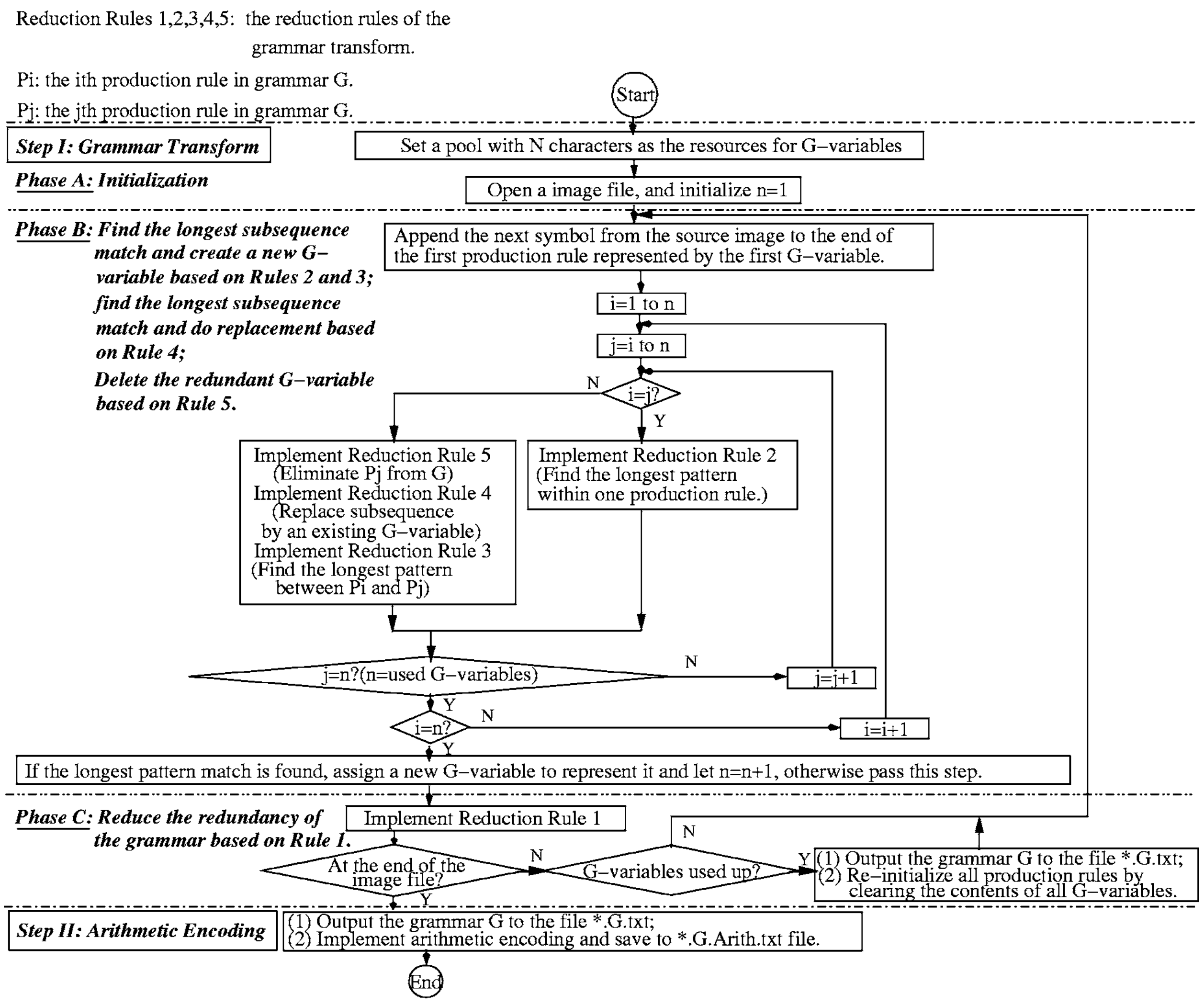

Reduction Rules 1,2,3,4,5: the reduction rules of the grammar transform.
Pi: the ith production rule in grammar G.
Pj: the jth production rule in grammar G.
Start
Step I: Grammar Transform
Phase A: Initialization
Set a pool with N characters as the resources for G−variables
Open a image file, and initialize n=1
Phase B: Find the longest subsequence match and create a new G−variable based on Rules 2 and 3; find the longest subsequence match and do replacement based on Rule 4; Delete the redundant G−variable based on Rule 5.
Append the next symbol from the source image to the end of the first production rule represented by the first G−variable.
i=1 to n
j=i to n
i=j?
N
Y
Implement Reduction Rule 5 (Eliminate Pj from G) Implement Reduction Rule 4 (Replace subsequence by an existing G−variable) Implement Reduction Rule 3 (Find the longest pattern between Pi and Pj)
Implement Reduction Rule 2 (Find the longest pattern within one production rule.)
j=n?(n=used G−variables)
N
j=j+1
Y
i=n?
N
i=i+1
Y
If the longest pattern match is found, assign a new G−variable to represent it and let n=n+1, otherwise pass this step.
Phase C: Reduce the redundancy of the grammar based on Rule 1.
Implement Reduction Rule 1
At the end of the image file?
N
G−variables used up?
N
Y
(1) Output the grammar G to the file *.G.txt; (2) Re−initialize all production rules by clearing the contents of all G−variables.
Y
Step II: Arithmetic Encoding
(1) Output the grammar G to the file *.G.txt; (2) Implement arithmetic encoding and save to *.G.Arith.txt file.
End

Fig. 4 The encoding flow chart of the grammar-based code.

- Each G-variable s other than s_0 appears at least twice in the right-hand side of the production rules of G.
- There is no nonoverlapping repeated pattern of length greater than or equal to 2 in the right-hand side of the production rules of G.
- Each distinct G variable represents a distinct Λ sequence.

The irreducible grammar transform is a greedy one. It parses x sequentially into nonoverlapping subsequences and builds sequentially an irreducible grammar for each subsequence. Figure 4 gives a flow chart of the grammar-based code implementation. In the flow chart, the main loop, which includes phases B and C, will be executed every time the next symbol from the source x is appended to the end of $G(s_0)$. Within the main loop, reduction rules 1 to 5 introduced in Refs. 1 and 2 will be applied. In the remainder of this work, we refer to the greedy irreducible grammar transform as the *G transform.* As demonstrated in Ref. 2, one of the main tasks of the G transform is simply to search for the longest repeated nonoverlapping subsequences. Two kinds of searches are involved: one is to search the subsequences among all the sequences on the right-hand side of the existing production rules; therefore we name this kind of search as *among-search*; the other one is named *within-search*, which searches the subsequences within each production rule's sequence. Reduction rules 1, 3, 4, and 5 are utilized in among-search, while reduction rule 2 is used in within-search. Reduction rules 2 and 3 have to be applied within the main loop of the grammar code implementation, as shown in Fig. 4, because they will be applied at least once in the main loop, and may be invoked more than once if reduction rule 4 is invoked. For example, in Fig. 4, we define i as a counter of the subloop from 1 to n, where 1 to n correspond to the production rule 1 to n. We define j as another counter of the subloop from i to n,where i to n correspond to the production rule i to n. Among-search can be described as screening whether a subsequence resides in the i'th production rule and searching its match in the j'th production rule. When i equals j, the common subsequences match search is the within-search. Overall, the purpose of within-search (or among-search) is to find if the subsequence appears more than once within that production rule (or among all production rules).

Below is an example of the G-transform procedure. Let $A=\{0,1\}$ and

$$x = 10011100010001110001 1. \qquad (1)$$

Let G_n', G_n'', and so on be the grammars G, which are reducible, and G_n be the corresponding irreducible grammar. The footnote n represents the grammar G' being associated with the subsequence of x, which includes the first n symbols of x. The sequence that consists of the first six symbols of x does not have any nonoverlapping repeated pattern, so no reduction rules can be applied. Consequently, the G_6 only contains one G variable and one production rule:

$$s_0 \rightarrow 100111.$$

Appending the symbol 0 to the end of $G_6(s_0)$, we get a grammar G_6' given by

$$s_0 \rightarrow 1001\underline{110}.$$

G_6' is not irreducible any more, since there is a nonoverlapping repeated pattern 10 in the right-hand side of the production rules of G_6'. At this point, only reduction rule 2 is applicable. Applying reduction rule 2 once, we get the irreducible grammar G_7 given by

$$s_0 \rightarrow s_1 011 s_1$$

$$s_1 \rightarrow 10.$$

Appending the next symbol $x_8=0$ to the end of $G_7(s_0)$, we get grammar G_7' given by

$$s_0 \rightarrow s_1 011 \underline{s_1 0}$$

$$s_1 \rightarrow 10.$$

G_7' is not irreducible. Applying reduction rule 2 once, which is the only applicable reduction rule at this point, we get a grammar G_7''

$$s_0 \rightarrow s_2 11 s_2$$

$$s_1 \rightarrow 10$$

$$s_2 \rightarrow s_1 0.$$

In the previous, the variable s_1 appears only once in the right-hand side of the production rules of the grammar. Applying reduction rule 1 once, we get our irreducible grammar G_8:

$$s_0 \rightarrow s_1 11 s_1$$

$$s_1 \rightarrow 100.$$

From G_7 to G_8, we have applied reduction rule 2 followed by reduction rule 1. Similarly, appending the symbols from $x_9=0$ to $x_{15}=1$ to the end of $G(s_0)$ step by step, we get G_{12}

$$s_0 \rightarrow s_1 11 s_2 s_2 \underline{11}$$

$$s_1 \rightarrow 100$$

$$s_2 \rightarrow s_1 0.$$

For the detailed description of the G-transform procedure from G_8 to G_{12}, please refer to Ref. 2. From the G_{12} we get, applying reduction rule 2 once, we obtain G_{13}

$$s_0 \rightarrow s_1 s_3 s_2 s_2 s_3$$

$$s_1 \rightarrow 100$$

$$s_2 \rightarrow s_1 0$$

$$s_3 \rightarrow 11.$$

Appending $x_{16}=1$, $x_{17}=0$, $x_{18}=0$, and $x_{19}=0$ to the end of $G_{13}(s_0)$, respectively, and then applying reduction rule 2 followed by reduction rule 1, we get G_{14}

$$s_0 \rightarrow s_1 s_3 s_2 s_3$$

$$s_1 \rightarrow 100$$

$$s_2 \rightarrow s_1 0$$

$$s_3 \rightarrow 11 s_2.$$

G_{15} is obtained by appending $x_{20}=1$ to the end of G_{14} (s_0). Appending $x_{21}=1$ to the end of G_{15} (s_0), we get

$$s_0 \rightarrow s_1 s_3 s_2 s_3 \underline{11}$$

$$s_1 \rightarrow 100$$

$$s_2 \rightarrow s_1 0$$

$$s_3 \rightarrow 11 s_2.$$

Applying reduction rule 3 once, we get G_{16}

$$s_0 \rightarrow s_1 s_3 s_2 s_3 s_4$$

$$s_1 \rightarrow 100$$

$$s_2 \rightarrow s_1 0$$

$$s_3 \rightarrow s_4 s_2$$

$$s_4 \rightarrow 11.$$

In summary, the G transform transforms x into the irreducible grammar G_{16}

$$s_0 \rightarrow s_1 s_3 s_2 s_3 s_4$$

$s_1 \rightarrow 100$

$s_2 \rightarrow s_1 0$

$s_3 \rightarrow s_4 s_2$

$s_4 \rightarrow 11.$

Once the final irreducible grammar G is obtained, it will be compressed by using a zero-order arithmetic code with a dynamic alphabet. The decoder recovers G from the compressed codewords and then performs the parallel replacement procedure to recover x, as shown next:

$$s_0 \underset{\rightarrow}{G} s_1 s_3 s_2 s_3 s_4$$

$$s_1 s_3 s_2 s_3 s_4 \underset{\rightarrow}{G} 100 s_4 s_2 s_1 0 s_4 s_2 11$$

$$100 s_4 s_2 s_1 0 s_4 s_2 11 \underset{\rightarrow}{G} 10011 s_1 0100011 s_1 011$$

$$10011 s_1 0100011 s_1 011 \underset{\rightarrow}{G} 100111000100011100011.$$

We start with s_0 and then repeatedly apply the parallel replacement procedure. We see that after four steps, we recover the original sequence, and the parallel replacement procedure terminates.

4 Implementation

As presented in Sec. 3, the grammar-based source coding is accomplished by taking the following three steps.

1. Define a size-on-demand variable set of G and ensure each G variable is distinct from source symbols.
2. Convert the source sequence x into an irreducible context-free grammar (in which the first G variable s_0 represents the source sequence x, and each of the other G variables represents a subsequence of x) by applying the G transform presented in Ref. 2.
3. Use one of the three universal lossless data compression algorithms (sequential algorithm, improved sequential algorithm, or hierarchical algorithm) to compress the irreducible grammar. All these algorithms combine the power of arithmetic coding with that of string matching. The size of S, $|S|$, is defined as the number of G variables in S.

The rest of this work mainly describes the implementation of the grammar-based algorithm from different aspects.

4.1 *Implementation Issues of Encoding*

By studying the implementation of the grammar-based coding, we discovered two main implementation issues: the large processing time and limited number of G variables. Although the compression ratio obtained by the implementation of the code is comparable and even better than the conventional lossless schemes, these two factors limit its use for real-time and large image analysis. Typically, medical images such as mammograms are of large type and hence to effectively apply the grammar-based code, a new method has been derived. This work focuses on these issues.

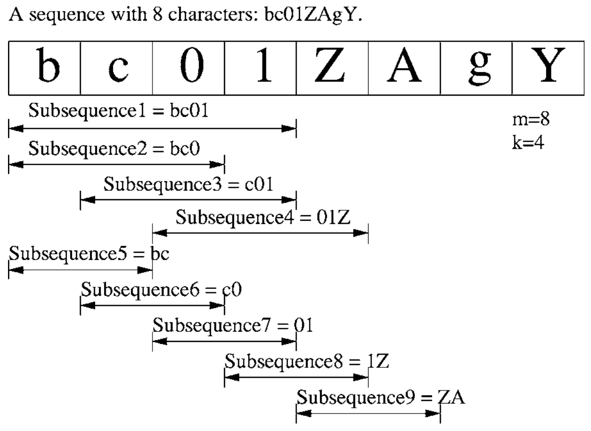


Fig. 5 The longest nonoverlapping common subsequence match search in the worst case in a sequence with even numbered characters.

4.1.1 *Processing-time issue*

From the study, we found that as we used more G variables for compression, the processing time we needed for converting from a source image to an irreducible grammar G increased at about a quadratical rate.

For the among-search, where $i \neq j$, if the size of G variables, $|S|$ equal N, matching subsequences from all the G variables takes maximum $N(N+1)/2$ time. This makes the algorithm time consuming. A larger image usually means a larger $|S|$ and the computational time becomes longer. When $i=j$, the program does a search within one production rule, and if $i=j=1$, such a production rule will be the first production rule. Figure 5 illustrates how the program does the longest nonoverlapping common subsequence match search within one production rule. It divides the sequence on the right-hand side of the production rule into two parts evenly. Assume that there are m characters in the sequence. The program defines that if m is an even number, then half of it will be $m/2$, otherwise $(m-1)/2$. Let k represent $m/2$ or $(m-1)/2$. Since we want to search the longest nonoverlapping matching subsequence, the program starts by setting a subsequence that contains the first k characters of the sequence, and looks for a match to the subsequence in the rest of the sequence.

1. If a match is found, a new G variable is assigned to represent the subsequence, and go to step 3. If a match is not found, go to step 2.
2. Reduce the subsequence by one symbol, and repeat the search for a match. If a match is found, a new G variable is assigned to represent the subsequence, and go to step 3. If a match is not found, repeat step 2.
3. Replace the subsequence in the original sequence by the new G variable and stop the search.

Figure 5 displays that subsequence 9 has to be searched, because no match could be found for subsequences longer than two. The total search time depends on m. If m is an even number, for example m equals 8 in Fig. 5, in the worst case, the total number of searches will be $(k-1)^2=(4-1)^2=9$, while if m is an odd number, the total time of search will be $k(k-1)$.

It was observed that the size of the first production rule increases much faster than others as the G transform continues. The reason is that the G transform appends the next character read from the source image file to the end of the first production rule, but not to the other production rules. If the image is large, the size of production rule 1 will be very large, and consequently the search times will be extremely high.

4.1.2 *Extended grammar codes*

Extended grammar codes are an approach to reduce the processing time. The approach, based on an important feature of the G transform, was discovered through our extensive studies. The following paragraphs describe the discovery and its benefit for the implementation of the grammar-based code.

As mentioned before, although the sequence on the right hand of the first production rule keeps expanding its length during the G-transform process, as shown in Fig. 4, each time at the end of the main loop that starts at the beginning of phase B and ends at the end of phase C, the grammar G is irreducible. Each G variable other than the first G variable is created based on its unique sequence without overlapping matching subsequences. In general, the sequence will not be changed unless a single variable locates in its sequence and is replaced by the sequence of that single variable; this change does not lead to a new matching subsequence. These existing G variables do not need to search subsequence match within themselves. On the other hand, the main loop is executed whenever a next source character is appended to the first production rule: therefore, from the view of the grammar, the only factor that may result in the matched subsequences emerging is the newly appended source character of the first production rule. Consequently, if a matched subsequence exists, this subsequence must be in the first production rule and composed of the newly appended source character and its immediate neighbor. Examples of these subsequences are highlighted with underscores in Eq. (1). Using this observation, we can simplify the match search by just finding the second one of the subsequence from the rest of the first production rule sequence, for example, the grammars G_6', G_7', G_{12} in Eq. (1), and from other production rule sequences, for example, the grammar G_{15} in Eq. (1).

With the new search method, the G transform has been simplified significantly. When these proposed schemes were put to test, the image quality after the G transform was the same as the ones using match search, as illustrated in Fig. 5, but the processing time was reduced from 10 to 20.

Although the processing time has been dramatically reduced, the processing time of the G transform is still longer than its counterpart, the LZW code. The LZW code takes advantage of the hash algorithm to locate a particular sequence, and with good probability can find the target string in one search. The grammar-based code cannot adopt the hash algorithm because it has to search match subsequences, not matched sequences only. However, the reduction of the processing time allows us to use a larger number of G variables.

4.1.3 *Visible variable issue*

Since string match is often used by the G transform and the digital mammograms are encoded using number characters from the ASCII character set, we should also use visible ASCII characters for the G variables. Furthermore, the ASCII character code was chosen because it is very easy to check for valid data ranges, which is a very important feature for programming the grammar-based code. Unfortunately, the number of visible ASCII characters is limited. The maximum number of the characters that can be used in this study as G variables is 75, and these characters are:

@ ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkl
mnopqrstuvwxyz~}|{'_^][? = ;:/ . − , + * $ # !

For the rest of the character set, some are control characters rather than symbols, so that they cannot be utilized in the study. Some are source alphabets that encode the digital mammograms, and the others are used as identification codes for the output of the irreducible grammar, such as " %&' " and "()⟨⟩." Figure 6 shows an example of an original mammographic image segment transformed to a grammar-based encoded file **.G.arith.txt* using visible ASCII characters as the G variables.

As noted in Sec. 3, the G variables that represent the distinct production rules are distinct. $|S|$ is dependent on the image size as the G transform is applied. The larger the image size is, the bigger $|S|$ will be.

To enlarge $|S|$, we propose to use double characters to represent G variables. For example, if there are 64 single characters, we can obtain in total $64^2=4096$ G variables. To ensure every double-character G variable is equally used, the total $|S|$ value has to be the square of an integer, otherwise some characters will be used less and some redundant double-characters will never be used. The use of double-character G variables gets smaller entropy value than the use of single-character G variables. The final codeword lengths of the cases are equal under the condition that all the G variables are equally probable in the transformed grammar.

For the grammar G using single-character as the G-variables, l is defined as 1. If the grammar has N (where $N=|S|$) unique G variables, the number of source symbols is N, the equal probability of the G variables is presented as $1/N$. Define y as the total number of occurrences of these N G variables in a grammar. The equations of entropy H and codeword length is defined in Eq. (2)

$$H(\text{single}-\text{character}) = -\sum_{i=1}^{N} \frac{1}{N} \log N = -\log N$$

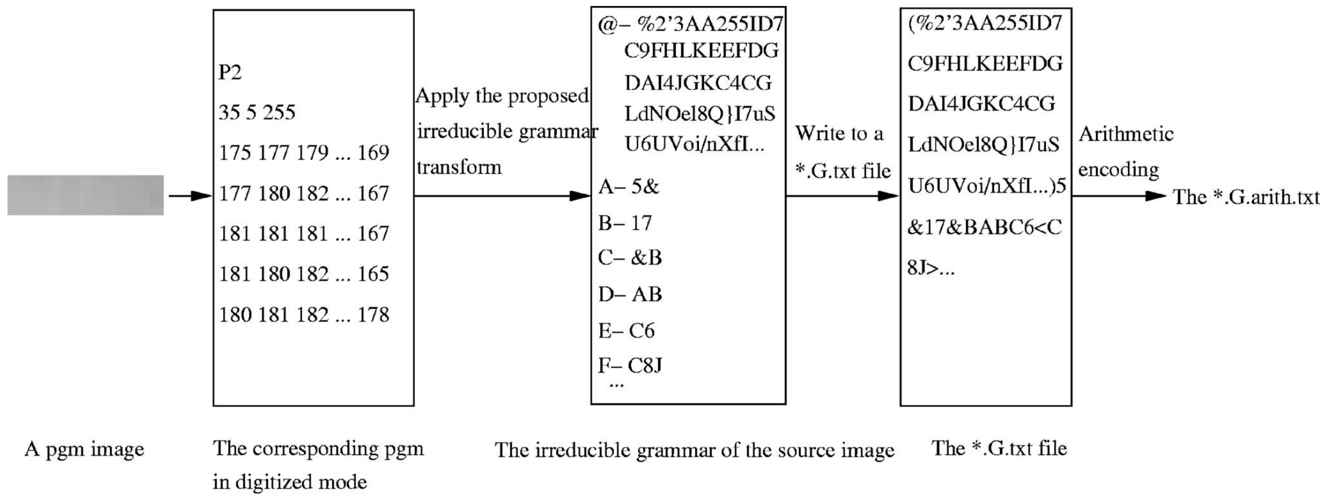

P2
35 5 255
175 177 179 ... 169
177 180 182 ... 167
181 181 181 ... 167
181 180 182 ... 165
180 181 182 ... 178
Apply the proposed irreducible grammar transform
@– %2'3AA255ID7 C9FHLKEEFDG DAI4JGKC4CG LdNOel8Q}I7uS U6UVoi/nXfI...
A– 5&
B– 17
C– &B
D– AB
E– C6
F– C8J
Write to a *.G.txt file
(%2'3AA255ID7 C9FHLKEEFDG DAI4JGKC4CG LdNOel8Q}I7uS U6UVoi/nXfI...)5 &17&BABC6<C 8J>...
Arithmetic encoding
The *.G.arith.txt
A pgm image
The corresponding pgm in digitized mode
The irreducible grammar of the source image
The *.G.txt file

Fig. 6 An example of an original mammographic image segment transformed to a grammar-based encoded file *. *G.arith.txt.*

$$\text{codewordlength(single} - \text{character)}$$
$$= y \times l \times H = y \times 1 \times (-\log N). \quad (2)$$

For the grammar G using double characters as the G-variables, l is defined as 2. If the grammar has N (where $N=|S|$) unique G variables, the number of source symbols is the square of N, the equal probability of the G variables is presented as $1/\sqrt{N}$. Define y as the total number of occurrences of these N G variables in a grammar. The equations of entropy H and codeword length is defined in Eq. (3):

$$H(\text{double} - \text{character}) = -\sum_{i=1}^{\sqrt{N}} \frac{1}{\sqrt{N}} \log \sqrt{N}$$
$$= -\log \sqrt{N} = -\frac{1}{2} \log N$$

$$\text{codewordlength(single} - \text{character)}$$
$$= y \times l \times H = y \times 2 \times \left(-\frac{1}{2}\log N\right) = y \times 1 \times (-\log N). \quad (3)$$

In our study, we found that G variables have near-equal probabilities in most cases. Hence, using a double-character set to increase the G-variable symbols is feasible.

4.2 *Decoding*

The decoding process of the grammar-based code involves two steps: first, arithmetic decoding decodes the **.G.arith.txt* file to the grammar file named **.G.txt*; second, the grammar file is then decoded to the original image file, which involves the removal of the wrappers, such as the special symbols ()⟨⟩, the reconstruction of G-variables, and the parallel replacement of G variables to recover the original image.

The decoding procedure is much faster than the encoding procedure for the grammar-based code. More specifically, it takes at most N times of the G-variable parallel replacement to recover the source image. Therefore, the decoding time is in proportion to the size of G variables N.

4.3 *Results*

In the research, we are using double-character G variables. To compress 30 mini-MIAS images (1024 × 1024 pixels), we utilized two sets of G variables, and these two sets of G variables have sizes of 81 and 324, respectively. We also applied three sets of G variables to compress 30 mini-MIAS images (512 × 512 pixels) and they have sizes of 81, 324, and 1024. The comparisons of compression ratios between the extended grammar-based code and others, such as the Huffman, arithmetic, and LZW codes, and the combination of LZW and arithmetic code, are shown in Tables 1 and 2.

Both tables show that arithmetic coding outperforms Huffman coding in most cases, but both are worse than the grammar-based coding and LZW, if LZW uses a larger dictionary. The reason is that the probability-based codes, such as arithmetic and Huffman codes, cannot capture the higher-order correlation in the source file.

Tables 1 and 2 also show that as the number of G variables increases, the compression performance of the grammar-based code becomes better. For 1024 × 1024 images, the average compression ratios are from 3.5 to 3.6, and for 512 × 512 images, the average compression ratios are from 3.13 to 3.48 to 4.04. It proves that more G variables are used, and more higher-order correlations of images can be separated, so that a higher compression ratio can be achieved. This theory is also applicable to LZW, as seen in Table 1.

Compared with the three conventional lossless compression techniques and the combination of LZW and arithmetic coding, the grammar-based code outperforms others under the same conditions. For example, in Tables 1 and 2, the compression rate of the grammar-based code using 324 G variables (its dictionary size is between 8 and 9 bits) is higher than LZW coding and LZW coding with arithmetic coding using a 9-bit dictionary. Similarly, Table 1 indicates

Table 1 STAGE 2: The comparison of compression ratio between the extended grammar-based code and others on 30 mini-MIAS images (512×512) with an average size of 718,699 bytes. Legends: † MIAS is the Mammographic Image Analysis Society, † G is the grammar-based code, † A is arithmetic, † H is Huffman, and † LZW is Lempel-Ziv-Welch.

Techniques		Compression ratio	
G	81 *G variables*	Size	233,671 bytes
		Ratio	3.13
	8 *bits*<324 *G variables*	Size	211,574 bytes
	<9 *bits*	Ratio	3.48
	1024 *G variables*(10 bits)	Size	183,478 bytes
		Ratio	4.04
LZW	9 *bits*	Size	259,811 bytes
		Ratio	2.84
LZW+A	9 *bits*	Size	247,052 bytes
		Ratio	2.98
LZW	10 *bits*	Size	183,018 bytes
		Ratio	4
LZW+A	10 *bits*	Size	182,378 bytes
		Ratio	4.02
LZW	12 *bits*	Size	182,708 bytes
		Ratio	4.03
LZW+A	12 *bits*	Size	176,725 bytes
		Ratio	4.17
A	Independent of dictionary size	Size	251,778 bytes
		Ratio	2.87
H	Independent of dictionary size	Size	258,797
		Ratio	2.79

Table 2 STAGE 2: The comparison of compression ratio between the extended grammar-based code and others on 30 mini-MIAS images (1024×1024) with an average size of 2,877,523 bytes. Legends: † MIAS is the Mammographic Image Analysis Society, † *G* is the grammar-based code, † A is arithmetic, † H is Huffman, and † LZW is Lempel-Ziv-Welch.

Techniques		Compression ratio	
G	81 *G variables*	Size	837,285 bytes
		Ratio	3.5
	8 *bits*<324 *G variables*	Size	818,961 bytes
	<9 *bits*	Ratio	3.6
LZW	9 *bits*	Size	1,180.474 bytes
		Ratio	2.49
LZW+A	9 *bits*	Size	1,111,890 bytes
		Ratio	2.64
LZW	12 *bits*	Size	742,692 bytes
		Ratio	3.97
LZW+A	12 *bits*	Size	711,838 bytes
		Ratio	3.97
A	Independent of dictionary size	Size	1,013,087 bytes
		Ratio	2.85
H	Independent of dictionary size	Size	1,041,679 bytes
		Ratio	2.78

that the compression rate of the grammar-based code using 1024 *G* variables (its dictionary size equals 10 bits) is also higher to LZW coding and its combination with arithmetic coding when they use a 10-bit dictionary as well.

In the study, we also tested the compression performance of the proposed technique combined with Huffman coding. The result shows that this combination performs worse than its combination with arithmetic coding by 1% on average. The reason is that arithmetic coding does not need to use an integer number of bits to each source symbol, so that the average bit rate achieved is closer to the entropy of the source.

5 Error Control Coding

In this section, we study how the performance of the grammar-based code is affected by transmission errors and how the performance could be improved in that case.

In our study, we tested the case where Gaussian noise is added to the output file compressed by grammar-based codes. After arithmetic decoding and recovering grammar *G* via parallel replacement, the restored images had significant distortions. We group the results into two groups, and Figs. 7 and 8 show the examples of these results. In the restored images, some had substantial distortions, as displayed in Figs. 7 and 8, and some even could not be restored by the grammar decoder at all. The reason is due to the nature of the arithmetic coding. The codeword of arithmetic coding is a floating point number, and each digit should be precise. If any one of them is changed, the decoder will fail and this error will propagate. Since the grammar-based code transforms the source image into a grammar and then encodes it using an arithmetic code, once the codeword of arithmetic coding is infected by errors, the image restored by arithmetic decoding will have errors in the corresponding position. In other words, the initial part of the image can be recovered correctly until an error ap-

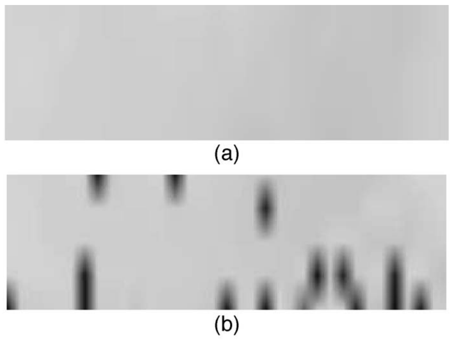


Fig. 7 The comparison of images before and after noise added on grammar-based coded files.

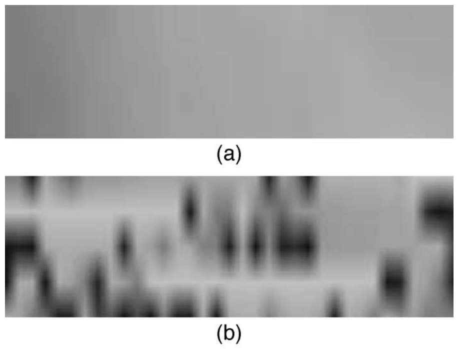


Fig. 8 The comparison of images before and after noise added on grammar-based coded files.

pears. To protect this error from happening, channel coding, which adds redundant bits for error detection and correction, has to be used.

In the study, channel encoding and decoding were applied, and they are the convolutional encoder and the Viterbi decoder with additive white Gaussian noise (AWGN). Even though there are other simpler channel coding schemes such as linear block codes (LBC) and their branch cyclic codes, their error correction capabilities are limited, since they independently detect and correct the data sequence block by block, which means they do not consider the correlation of data blocks. If a higher correction rate is required, more redundant bits are needed, and correspondingly the encoding and decoding efficiencies of the schemes decrease. A convolutional encoder is a code that considers the correlation between data blocks, so that its error correction efficiency is relatively higher than LBC and cyclic codes. Also, it can be easily implemented, and is suitable for a continuous data stream and to a channel in which the transmitted signal is corrupted mainly by AWGN.

In the study, we defined five noise power levels for the AWGN channel by letting a Gaussian random number be divided by noise factors from 5 to 1, respectively. Therefore, the corresponding values of the five noise power levels increase according to the second-order polynomial. We

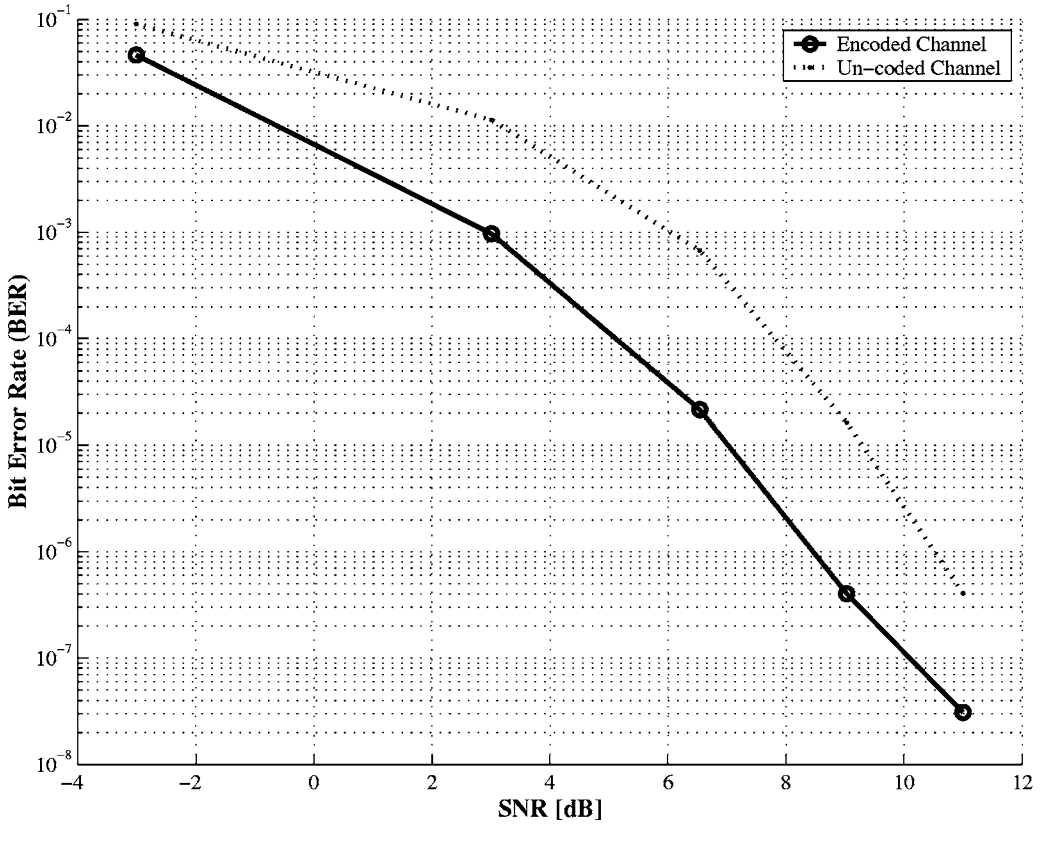


Fig. 9 SNR versus BER average results on 15 mini-MIAS images for rate 1/2 convolutional encoder and the decoding length N=128 bits.

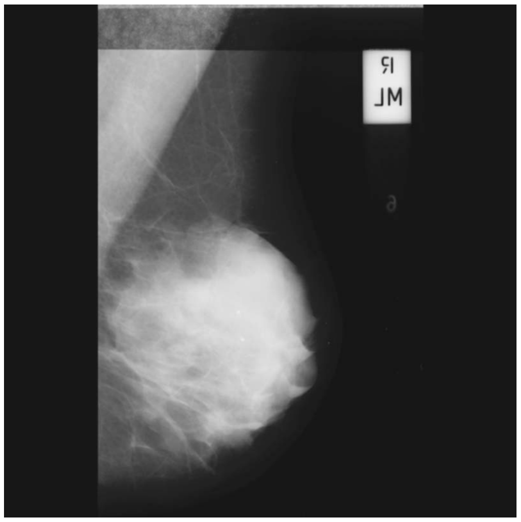


Fig. 10 The completely recovered mini-MIAS image (mdb002.pgm) in a channel with SNR=11 dB (noise factor=5), rate 1/2 convolutional channel encoder, and Viterbi channel decoder.

transmitted 15 mini-MIAS images encoded by the extended grammar-based code through the defined AWGN channel in two cases, channel encoded and channel uncoded. Figure 9 displays the probability of bit error rate (BER) versus received (SNR) signal-to-noise rate of the tests. It indicates the fact that the BER of the channel without using convolutional coding is higher than the channel using convolutional coding under the same SNR. Consequently, the test result shows that an image file in the encoded channel can tolerate noise at least one noise power level higher than the one in the uncoded channel when the rate of 1/2 convolutional encoder is applied. It means that channel encoding can protect data from noise to some extent. For example, when the noise factor is set as 5, the image file in the uncoded channel cannot be restored at all, while the one in the channel with convolutional encoding (rate 1/2) can be restored completely, as displayed in Fig. 10. Furthermore, as the noise factor is set as 4, the noise power becomes higher, the image file in the uncoded channel still cannot be recovered, and in the encoded channel the recovered part of the image file is smaller and will always be the beginning of the image, as mentioned in the second paragraph in this section. The recovered image can be found in Fig. 11. In theory, as the encoding rate of the convolutional encoder is increased, more noise in the file transmitted through the channel can be detected and corrected.

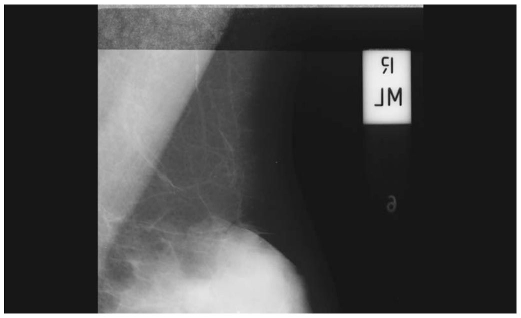


Fig. 11 The first 616 pixel lines recovered mini-MIAS image (mdb002.pgm) in a channel with SNR=9.03 dB (noise factor=4), rate 1/2 convolutional channel encoder, and Viterbi channel decoder.

6 Conclusion and Contributions

For decades, researchers have kept looking for more effective lossless compression techniques for critical and large images such as mammographic images. The lossless compression grammar-based coding algorithm proposed by Yang and Kieffer attracts our attention and prompts us to investigate if it is a promising code, and what are its advantages and disadvantages compared with traditional lossless algorithms. To our knowledge, this work is the first attempt to investigate the applicability of grammar-based codes for compression of a medical image.

The two weaknesses of the original grammar-based codes are the limited G-variable set and the computational time. These two weaknesses are overcome to some extent by applying the extended grammar code technique and double-character G variables proposed in this work. Significantly less computational time is achieved and more G variables could be utilized through the use of double-character representation, and consequently the average bit rate could reduce if all the encoding characters are equally probable. These results are also proved analytically.

In our study, the arithmetic coding outperforms Huffman coding in most cases. LZW coding and grammar-based coding both are dictionary-based compression methods. They are more comparable and perform better than arithmetic and Huffman coding when they have larger dictionary size. Although LZW and grammar-based coding are dictionary-based lossless compression techniques, both do not require sending the dictionary to the decoder. The study also shows that, under the same dictionary size, the grammar-based code has better performance than LZW coding. The LZW process is faster than the grammar-based code because it can utilize the Hash algorithm, while the nature of grammar-based coding does not allow it to apply the Hash algorithm, as discussed in Sec. 4.1.

Also, such a technique is very sensitive to errors because of its use of the irreducible G transform and arithmetic algorithm. Section 5 shows that arithmetic code can be easily infected by errors, but cannot do self-synchronization to avoid error propagation as the Huffman code does. Therefore, data correction coding has to be adopted for the transmission.

Acknowledgment

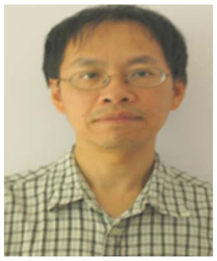
I wish to acknowledge those who I feel have greatly aided me in completing this research. I would like to thank Professor S. Krishnan for providing an open study and discussion environment, constant encouragement and willingness to provide advice, and very clear guidance toward the success of this project. I would also like to thank Professor N. W. Ma for his enlightening guidance, constructive suggestions, high-standard requirement, and unconditional support. Many thanks to those who provide me technical support and help. Finally, special thanks go to my family.

References

1. N. R. Wanigasekara, S. Ding, Z. Yan, and Y. Zeng, "Quality evaluation for JPEG 2000 based medical image compression," *EMBS/BMES Conf. Proc.* **2**, 1019–1020 (2002).
2. M. Y. Al-Saiegh and S. Krishnan, "Fixed block-based lossless compression of digital mammograms," *Electric. Computer Eng. Conf.* **2**, 937–942 (2001).
3. H. S. Wong, L. Guan, and H. Hong, "Compression of digital mammogram databases using a near-lossless scheme," *Image Process. Proc.* **2**, 21–24 (1995).
4. H. D. Lin, K. P. Lin, and S. L. Lou, "Digital mammographic image analysis and data compression," *IEEE Eng. Med. Biol. Mag.* **2**, 1025–1027 (1998).
5. Z. Yang, M. Kallergi, R. A. DeVore, B. J. Lucier, Q. Wei, R. A. Clark, and L. P. Clarke, "Effect of wavelet bases on compressing digital mammograms," *IEEE Eng. Med. Biol. Mag.* **14**(5), 570–577 (1995).
6. D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE* **40**, 1098–1101 (1952).
7. T. A. Welch, "A technique for high performance data compression," *IEEE Trans. Comput.* **17**(6), 8–19 (1984).
8. R. Pasco, "Source coding algorithms for fast data compression," PhD Thesis, Stanford University, CA (1976).
9. E. H. Yang and J. C. Kieffer, "Universal source coding theory based on grammar transform," *Proc. 1999 IEEE Info. Theory Commun. Workshop*, pp. 75–77 (1999).
10. E. H. Yang and J. C. Kieffer, "Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform Part one: Without context models," *IEEE Trans. Inf. Theory* **46**, 755–788 (May 2000).
11. SlipStream Company, see http://www.slipstream.com/tech.html.
12. Mammographic Image Analysis Society (MIAS) images data webpage, see http://peipa.essex.ac.uk/ipa/pix/mias/.

Xiaoli Li received her MASc degree in computer networks from Ryerson University, Toronto, Ontario, in 2004. She is currently pursuing her PhD in electrical and computer engineering at Ryerson University. Her research interests include image analysis and multimedia networking.

Sridhar (**Sri**) **Krishnan** received his BE degree in electronics and communication engineering from Anna University, Madras, India, in 1993, and his MSc and PhD degrees in electrical and computer engineering from the University of Calgary, Calgary, Alberta, Canada, in 1996 and 1999 respectively. He joined the Department of Electrical and Computer Engineering, Ryerson University, Toronto, Ontario, Canada in July 1999, and currently he is an associate professor and chairman of the department. His research interests include adaptive signal processing, biomedical signal/image analysis, and multimedia processing and communications.



Ngok-Wah Ma received his BASc, MASc, and PhD degrees in electrical engineering from the University of Waterloo, Ontario, Canada in 1982, 1984, and 1988, respectively. In September 1988, he joined the Department of Electrical and Computer Engineering, Ryerson University, Toronto, Ontario, Canada, where he is currently a professor and the director of the computer Networks Master of Engineering program. His current research interests are in quality of service of internet, key distribution for secure multicast, and digital signal processing.