COMPRESSION OF SPEECH AND IMAGE SIGNALS

K.K. Paliwal
Signal Processing Laboratory
School of Microelectronic Engineering
Griffith University
Brisbane, QLD 4111, Australia

1. Introduction

Signal compression is the process of finding a compact digital representation of a signal. Its aim is to reduce the bit-rate of a digital signal with or without loss of information about the signal. When compression is achieved with a loss of signal information, it is called lossy signal compression; otherwise it is called lossless signal compression. In the literature, the process of signal compression is also described by other names (such as signal coding, bandwidth compression, data compression, source coding, digital coding, etc.). In this paper, we will use the terms signal compression and coding alternately, both describing the same process.

Currently, signal compression (or, coding) is very much a part of our everyday life. Its applications are primarily in transmission and storage. When you talk to your friend on a cellular phone, your speech is first converted into digital form and then compressed so that it can be transmitted over a limited-bandwidth radio channel. Image and video clips are compressed prior to their storage on network servers. When you download an image through the Internet, you most likely receive a compressed image and your web browser decompresses it prior to its display on your computer. In this case, the compression process is reducing both the storage requirement on the net server and transmission time on the Internet.

In the Signal Processing Laboratory at Griffith University, we are interested in the compression of telephone speech, wideband speech, audio, image and video signals. In this paper, we confine our scope to speech and image coding, and describe the research work done at our laboratory in the following three areas: 1) lossy speech coding, 2) lossy image coding, and 3) lossless image coding.

2. Lossy speech coding (after [1])

In lossy speech coding, compression is associated with a loss of information; i.e., it is not possible to reconstruct the original speech signal from the compressed signal. Basic aim of lossy speech coding research is to minimize either the bit rate of the speech signal at a given audible distortion, or the audible distortion in the reconstructed speech signal at a given bit rate. In this paper, we deal only with the low bit rate coding of telephone speech which has a bandwidth of 3.2 kHz. We describe here some of the popular speech coders and discuss the current trends in this area of research which are aimed at reducing the bit rate below 4 kb/s. For a detailed treatment of speech coding, see [2, 3].

Though the speech coding research was originally initiated with the need of bandwidth compression for long-range transmission of telephone signals in mind, this need has now practically disappeared due to the availability of microwave and optical transmissions having almost infinite bandwidth. However, with the introduction of network services (such as mobile (or cellular) telephony, personal communication network (PCN), integrated services digital network (ISDN), digital packetized transmission, etc.), the need for speech coding is now again there. Speech coding is also required for secure voice transmission. In addition to these digital transmission applications, speech coding is useful for digital storage and is used in a number of applications, such as in storing messages in voice mail and voice messaging services. Because of these applications, speech coding is an important area of research.

There are four important attributes that describe a given speech coder. These are: quality, bit rate, delay and complexity. An ideal speech coder should have high quality, low bit rate, low delay and low complexity. Other parameters that have to be taken into account during the design of a speech coder
are the effect of channel errors and tandeming on its performance and its ability to handle nonvoice signals such as voiceband modem waveforms.

The attribute that describes the performance of a speech coder best is its quality at a given bit rate. Quality of the coded speech can be objectively measured generally in terms of signal-to-noise ratio and segmental signal-to-noise ratio. These objective measures are used extensively in the development phase of a coder for its evaluation. But, these measures do not always do a good job of predicting the subjective quality of coded speech. Therefore, subjective measures are required to get a good assessment of its quality which is needed, at least, during the final phase of its development. There are at present three major procedures for assessing the subjective quality of speech coders: the diagnostic rhyme test (DRT), the diagnostic acceptability measure (DAM), and the mean opinion score (MOS). The DRT measures intelligibility, whereas the DAM provides a characterization of coded speech in terms of a broad range of distortions [4]. The MOS procedure combines all aspects of performance in one single number, and is perhaps the most commonly used procedure for measuring the subjective quality of the coded speech. The MOS score is obtained by averaging the scores given by a set of 30-60 untrained listeners. Each listener characterizes the coded speech signal by a score on a scale from 1 (for unacceptable quality) to 5 (for excellent quality). MOS scores of 4.0 or higher define good quality or toll quality (near transparent) coding. An MOS score 3.5 to 4.0 defines the communication quality coding which is good enough to support natural telephone communication, but the coded speech has detectable distortion. MOS scores lower than 3.0 signify synthetic quality coding, where speech is intelligible, but lacks in naturalness and speaker recognizability.

As mentioned earlier, speech coding aims at reducing the bit rate of the speech signal without distorting voice quality. In order to do this, it is necessary to exploit properties of the speech production mechanism as well as the human auditory system. Speech production system introduces certain redundancies in the speech signal which, when removed from it, allow its coding at lower bit rates. A proper understanding of speech production mechanism makes it possible to construct models which capture these redundancies efficiently. These models describe the speech signal in terms of a few parameters which, due to limitations on the shape and rate of movement of the vocal tract, vary slowly over time and have a limited dynamic range. Slow variation of a parameter over time allows less frequent updates and interpolation of its value, while a small dynamic range allows usage of less number of quantization levels. The all-pole model is a typical example used in speech coding for capturing the redundancy in the speech signal. This model represents short-term spectral envelope of the speech signal and is characterized in terms of a few linear prediction (LP) coefficients.

Understanding of human auditory mechanism is important for quantifying the perceived distortion. In speech coders, it is the perceived distortion that is minimized, not the root mean square (RMS) difference between the original and the coded speech waveforms. Generally, a speech coder can be designed better by taking advantage of the properties of the human auditory system. For example, it is known that the human auditory system is more tolerant to quantization errors in the formant regions of the speech spectrum than in the nonformant regions [5]. Incorporation of this property into the distortion criterion has led to a number of speech coding techniques [6, 7].

2.1. LP-based speech coders

Speech coding is essentially a process of redundancy removal. In speech signal, the main types of redundancy arise from (1) correlation between successive samples of speech signal (or, lack of flatness in power spectrum), and (2) periodicity during voiced sounds. LP analysis captures the first type of redundancy in the form of LP parameters. Due to limitations on the shape and rate of movement of the vocal tract, the LP parameters are computed from the speech signal at a slow rate, typically at 50 frames/s. These parameters are used for removing the redundancy from the speech signal and the resulting residual signal is quantized and transmitted. Quantized values of the LP parameters are transmitted as side information. At the receiving end, the quantized residual signal and LP parameters are used to reconstruct the speech signal. In this section, different aspects of LP-based speech coding are briefly described. This includes estimation of LP parameters, their quantization and some of the popular LP-based speech coders. For more details, see [8, 9].
2.1.1. Speech analysis

Consider a frame of speech signal having $N$ samples, $\{s_1, s_2, \ldots, s_N\}$. In LP analysis, it is assumed that the current sample is approximately predicted by a linear combination of $p$ past samples; i.e.,

$$\hat{s}_n = -\sum_{k=1}^{p} a_k s_{n-k},$$

where $p$ is the order of LP analysis and $\{a_1, \ldots, a_p\}$ are the LP coefficients. Let $e_n$ denote the error between the actual value and the predicted value; i.e.,

$$e_n = s_n - \hat{s}_n = s_n + \sum_{k=1}^{p} a_k s_{n-k}.$$  \hspace{1cm} (2)

Since $\{e_n\}$ is obtained by subtracting $\{\hat{s}_n\}$ from $\{s_n\}$, it is called as the "residual" signal, and if $p$ is large enough, its samples are uncorrelated; i.e., it has flat spectrum. By taking $z$ transform of Eq. (2), it follows

$$E(z) = A(z)S(z),$$

where $S(z)$ and $E(z)$ are the $z$ transforms of the speech signal and the residual signal, respectively, and

$$A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k}.$$ \hspace{1cm} (4)

The filter $A(z)$ is known as the "whitening" filter as it removes the correlation present in the speech signal. This property makes it useful for redundancy removal in speech coding. We see from Eq. (3) that since $E(z)$ has flat spectrum, the signal spectrum is modeled in LP analysis by an all-pole (or, autoregressive) model

$$H(z) = 1/A(z).$$ \hspace{1cm} (5)

The filter $A(z)$ is also known as the "inverse" filter as it is the inverse of the all-pole model $H(z)$ of the speech signal.

In LP analysis, the short-time power-spectral envelope of speech is obtained by evaluating $H(z)$ on unit circle. However, for this, the LP coefficients have to be computed first from the speech signal. These are determined by minimizing the total-squared LP error,

$$E = \sum_{n=n_1}^{n_2} e_n^2,$$ \hspace{1cm} (6)

where the summation range $[n_1, n_2]$ depends on which of the two methods (the autocorrelation method and the covariance method) is used for LP analysis. In the autocorrelation method of LP analysis, the summation range is $[-\infty, \infty]$ which means that the speech signal has to be available for all time. For short-time LP analysis, this can be achieved by windowing the speech signal and assuming the samples outside this window to be zero. For windowing, the tapered cosine window functions (such as the Hamming and Hanning window functions) are preferred over the rectangular window function and the speech signal is multiplied by one of these window functions prior to its LP analysis. In the covariance method of LP analysis, the summation range is $[p+1, N]$. Therefore, there is no windowing required here. Minimization of the total-squared-error, $E$, results in a set of $p$ normal equations which can be solved for the LP coefficients by the computationally efficient algorithms [8].

2.1.2. Quantization of LP parameters

Considerable work has been done in the past to develop quantization procedures, both scalar and vector, to represent the spectral envelope information with smallest numbers of bits. The scalar quantizers
quantize each LP parameter independently. The vector quantizers consider the entire set of LP parameters as an entity and allow for direct minimization of quantization distortion. Because of this, the vector quantizers result in smaller quantization distortion than the scalar quantizers at any given bit rate. Here, we provide a brief glimpse of LP quantization results using the scalar as well as the vector quantization techniques. For more details, see [10].

Direct scalar quantization of the LP coefficients is usually not done as small quantization errors in the individual LP coefficients not only produce relatively large spectral errors, but can also affect the stability of the all-pole filter. Therefore, it is necessary to transform the LP coefficient representation to new representations which ensure stability of the all-pole filter after LP quantization. In addition, these representations should have one-to-one mapping; i.e., it should be possible to transform from one representation to another without loosing any information about the all-pole filter. In the literature, a few representations which have these properties have been proposed. These are: the arcsine reflection coefficient (ASRC) representation [12], the log-area ratio (LAR) representation [11], and the line spectral frequency (LSF) representation [13]. We provide in Table 1 results for scalar quantization of LP information using these representations. Table 1 also has results where the LSF differences (LSFDs) are used for the scalar quantization of LP information [14]. We measure LP quantization performance in terms of spectral distortion which is defined as the root mean square difference between the original LP log-power spectrum and the quantized LP log-power spectrum. We can get transparent quantization of LP information if we maintain the following conditions: 1) The average distortion is about 1 dB, 2) There is no outlier frame having spectral distortion larger than 4 dB, and 3) The number of outlier frames having spectral distortion in the range 2-4 dB is less than 2%. From Table 1, we can see that transparent quantization of LP information is possible using about 32-34 bits/frame.

Juang et al. [15] have studied vector quantization of LP parameters using the likelihood distortion measure and shown that the resulting vector quantizer at 10 bits/frame is comparable in performance to a 24 bits/frame scalar quantizer. This vector quantizer at 10 bits/frame has an average spectral distortion of 3.35 dB, and is not acceptable for high-quality speech coders. For transparent quantization of LP information, the vector quantizer needs more bits to quantize one frame of speech. This means that the vector quantizer will have a large number of codevectors in its codebook. Such a vector quantizer has the following two problems. Firstly, a large codebook requires prohibitively large amount of training data and the training process can take too much of computation time. Secondly, the storage and computational requirements for vector quantization encoding will be prohibitively high. Because of these problems, a sub-optimal vector quantizer has to be used for getting transparent quantization

<table>
<thead>
<tr>
<th>Bits used</th>
<th>Parameter</th>
<th>Av. SD (in dB)</th>
<th>Outliers (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2-4 dB</td>
<td>&gt;4 dB</td>
</tr>
<tr>
<td>36</td>
<td>LSF</td>
<td>0.79</td>
<td>0.46</td>
</tr>
<tr>
<td>36</td>
<td>LSFD</td>
<td>0.75</td>
<td>0.60</td>
</tr>
<tr>
<td>36</td>
<td>ASRC</td>
<td>0.81</td>
<td>0.90</td>
</tr>
<tr>
<td>36</td>
<td>LAR</td>
<td>0.80</td>
<td>1.09</td>
</tr>
<tr>
<td>34</td>
<td>LSF</td>
<td>0.92</td>
<td>1.00</td>
</tr>
<tr>
<td>34</td>
<td>LSFD</td>
<td>0.86</td>
<td>1.10</td>
</tr>
<tr>
<td>34</td>
<td>ASRC</td>
<td>0.92</td>
<td>2.05</td>
</tr>
<tr>
<td>34</td>
<td>LAR</td>
<td>0.92</td>
<td>1.65</td>
</tr>
<tr>
<td>32</td>
<td>LSF</td>
<td>1.10</td>
<td>2.21</td>
</tr>
<tr>
<td>32</td>
<td>LSFD</td>
<td>1.05</td>
<td>3.13</td>
</tr>
<tr>
<td>32</td>
<td>ASRC</td>
<td>1.04</td>
<td>3.30</td>
</tr>
<tr>
<td>32</td>
<td>LAR</td>
<td>1.04</td>
<td>3.20</td>
</tr>
<tr>
<td>28</td>
<td>LSF</td>
<td>1.40</td>
<td>9.21</td>
</tr>
<tr>
<td>28</td>
<td>LSFD</td>
<td>1.25</td>
<td>7.36</td>
</tr>
<tr>
<td>28</td>
<td>ASRC</td>
<td>1.32</td>
<td>9.29</td>
</tr>
<tr>
<td>28</td>
<td>LAR</td>
<td>1.34</td>
<td>9.51</td>
</tr>
</tbody>
</table>
of LP information. Various forms of sub-optimal vector quantizers have been suggested in the past which reduce the computational complexity and/or memory requirement, but at the cost of reduced performance [21]. Known most among these are the tree-search, multi-stage and product-code vector quantizers. Here, we provide results for the 2-part split vector quantizer using the weighted LSF distance measure [10]. The split vector quantizer is a type of product-code vector quantizer where the LSF vector is divided into two parts, and each part is vector-quantized independently. We can see from Table 2 that this vector quantizer can perform transparent quantization of LP information with 24 bits/frame.

### Table 2: LP quantization results using the split vector quantizer.

<table>
<thead>
<tr>
<th>Bits used (in dB)</th>
<th>Av. SD</th>
<th>Outliers (in %)</th>
<th>2-4 dB</th>
<th>&gt;4 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>0.90</td>
<td>0.44</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>0.96</td>
<td>0.61</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>24</td>
<td>1.03</td>
<td>1.03</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>23</td>
<td>1.10</td>
<td>1.60</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>22</td>
<td>1.17</td>
<td>2.73</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>21</td>
<td>1.27</td>
<td>4.70</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>1.34</td>
<td>6.35</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### 2.1.3. Interpolation of LP parameters

In speech coding, LP parameters are quantized frame-wise and transmitted. Frames are typically updated every 20 ms. This slow update of frames can lead to large changes in LP parameter values in adjacent frames which may introduce undesired transients (or, clicks) in the reconstructed (or, synthesized) speech signal. To overcome this problem, interpolation of LP parameters is used at the receiving end to get smooth variations in their values. Usually, interpolation is done linearly at a few equally-spaced time instants (called subframes) within each frame. The LP parameters can, in principle, be interpolated on a sample-by-sample basis. However, it is not necessary to perform such a fine interpolation. In addition, it is computationally expensive. Linear interpolation is generally done at a subframe interval of about 5 ms.

Any representation of the LP parameters which has a one-to-one correspondence to the LP coefficients can be used for interpolation, including the LP coefficient, the reflection coefficient, the log-area-ratio, arc-sine reflection coefficient, the cepstral coefficient, the line spectral frequency (LSF), the autocorrelation coefficient, and the impulse response representations. Though each of these representations provides equivalent information about the LP spectral envelope, their interpolation performance is different. A few studies have been reported in the literature [16, 17, 18, 19] where some of these representations are investigated for interpolation. For example, Itakura et al. [16, 17] and Atal et al. [18] have studied log-area-ratio (LAR), arc-sine reflection coefficient and line spectral frequency (LSF) representations for interpolation and found the LSF representation to be the best. We have investigated the interpolation performance of all of these LP parametric representations and report results in terms of spectral distortion measure which is defined as the root-mean-square difference between the original LP log-power spectrum and the interpolated LP log-power spectrum. For more details about these results, see [20].

It may be noted that some of these LP parametric representations (LP coefficient, cepstral coefficient and impulse response representations) may result in an unstable LP synthesis filter after interpolation. If these representations are used for interpolation, the LP parameters after interpolation must be processed to make the resulting LP synthesis filter stable. This processing is computationally expensive and, hence, these unstable representations should not be used for interpolation, if possible. However, some of the popular speech coding systems reported in the literature [31] have used the unstable LP coefficient representation for interpolation. Therefore, we use the number of unstable subframes resulting from the interpolation process as another measure of interpolation performance.

Interpolation is done linearly at a subframe interval of 5 ms. Results for the case when frame interval is 20 ms (i.e., frame rate is 50 frames/s) are listed in table 15. It can be seen from this table that the line spectral frequency representation provides the best interpolation performance in terms of spectral
Table 3: Interpolation performance of different LP parametric representations. Interpolation is done from LP parameters computed from speech at a frame interval of 20 ms.

<table>
<thead>
<tr>
<th>Representation</th>
<th>Av. SD (in dB)</th>
<th>Outliers (in %)</th>
<th>Unstable subframes (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP coefficient</td>
<td>1.35</td>
<td>15.7</td>
<td>2.1</td>
</tr>
<tr>
<td>Reflection coefficient</td>
<td>1.56</td>
<td>18.6</td>
<td>4.6</td>
</tr>
<tr>
<td>Log area ratio</td>
<td>1.50</td>
<td>17.7</td>
<td>4.1</td>
</tr>
<tr>
<td>Arc-sine reflection</td>
<td>1.53</td>
<td>18.2</td>
<td>4.4</td>
</tr>
<tr>
<td>Cepstral coefficient</td>
<td>1.40</td>
<td>18.6</td>
<td>1.3</td>
</tr>
<tr>
<td>Line spectral frequency</td>
<td>1.31</td>
<td>15.3</td>
<td>1.3</td>
</tr>
<tr>
<td>Autocorrelation coefficient</td>
<td>1.47</td>
<td>18.8</td>
<td>3.1</td>
</tr>
<tr>
<td>Impulse response</td>
<td>1.50</td>
<td>22.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

distortion. In addition, it always results in stable LP synthesis filters after interpolation. The LP coefficient representation also provides good interpolation performance in terms of spectral distortion measure. But, since it causes some unstable subframes, it is not a good choice for interpolation.

2.1.4. Popular LP-based speech coders

2.1.4.1. LP vocoder: In the LP vocoder [8], the speech-generating linear system is modeled by an all-pole filter and it is assumed that the speech signal is produced by exciting this filter either by a periodic pulse train (for voiced speech) or by a white random number sequence (for unvoiced speech). The speech signal is typically analyzed at the rate of 50 frames/s. For each frame, the following information (in quantized form) is transmitted: 1) the voiced/unvoiced decision, and if the frame is voiced, its pitch period, 2) the LP coefficients, and 3) the gain which is the square-root of the energy of the residual signal. Using these parameters, speech is reconstructed at the receiving end of the vocoder (shown in Fig. 1).

![Figure 1: LP vocoder.](image)

The LP vocoder can code speech signal at 2.4 kbits/s [22]. This bit-rate can be reduced to 800
bits/s by vector-quantizing the LP information [23]. Though the speech coded by the LP vocoder is intelligible, it has a synthetic quality because of the following two reasons. First, the LP vocoder model assumes a dichotomization of the speech signal in the voiced and the unvoiced classes which is not true. There are portions of speech where it is not clear whether the signal is voiced or unvoiced. Even if this dichotomization were possible, pitch extraction (i.e., voiced-unvoiced detection and pitch estimation) is a difficult problem and is usually prone to errors. Second, the model provides for excitation only one pulse per pitch period, which is inaccurate even for the clearly voiced speech sounds. Because of these limitations in the model, the LP vocoder can not produce high-quality speech, even at high bit-rates. These limitations can be overcome by the multipulse model (described in the next subsection) which does not require the voiced-unvoiced dichotomization and provides more than one pulse per pitch period for excitation.

2.1.4.2. Multipulse excited LP coder: The multipulse-excited LP coder (shown in Fig. 2) uses the multipulse model for synthesizing speech [6]. In this model, a sequence of multiple pulses excites the cascade of the pitch synthesis filter, $1/P(z)$, and the LP synthesis filter, $1/A(z)$. The pitch synthesis filter, $1/P(z)$, models the long-term correlation (2.5 ms to 15 ms) and introduces voice periodicity. This filter usually contains only one tap and its parameters (pitch period and amplitude) are determined for a given speech frame by searching for a maximum in the autocorrelation function in the 2.5-15 ms range. It may be noted here that the maximum in the autocorrelation function always defines the pitch, irrespective of whether the speech frame is voiced or unvoiced. In this sense, it is slightly misnomer to call this maximum as the pitch. The LP synthesis filter, $1/A(z)$, models the short-term correlation (less than 2 ms) and introduces the LP spectral envelope to the synthetic speech signal.

![Diagram](image)

**Figure 2:** Multipulse excited LP coder.

For estimating the multipulse sequence, Atal and Remde [6] have proposed an analysis-by-synthesis method. This method is a non-iterative sequential method where one pulse is estimated at a time. In this method, the speech signal is processed in the blocks of 10-20 ms duration. It might be noted here that the block duration is always less than the frame-update interval. For each block, the objective error is computed by subtracting from the original speech signal the synthetic signal generated by the cascade of the pitch and the LP synthesis filters. The objective error is filtered by a frame-specific weighting filter, $A(z)/A(z/r)$, where $r$ is the weighting factor and typically has a value of 0.9. This weighting filter tries to exploit the properties of human auditory masking [5]. It attenuates those frequencies where the error is perceptually less important and amplifies those frequencies where the error is perceptually more important. The location and the amplitude of each new pulse are determined by minimizing the total-squared value of the perceptually-weighted error.
Instead of computing the pitch parameters frame-wise, it is possible to compute them block-wise to get better performance from the coder. If this computation is done through the closed-loop analysis, the coder performance becomes even better [24]. In addition to the type of pitch analysis used, there are other parameters which affect the performance of the multipulse coder [25]. These parameters include the frame duration, the frame-update interval, the block duration, the LP analysis order and the number of pulses/block. Among these parameters the number of pulses per block affects the coder performance most. Typical values of these parameters are: frame duration=20 ms, frame-update interval=20 ms, block duration=5 ms, LP analysis order=10, and number of pulses/block = 5.

The multipulse coder can give good-quality speech at bit-rates in the range of 12-16 kbits/s. However, this coder cannot be used for bit-rates lower than 12 kbits/s because it shows drastic degradation in speech quality at these lower bit-rates. This happens because of the following two reasons. First, the analysis-by-synthesis procedure estimates the pulses sequentially with one pulse at a time. Thus, this procedure is sub-optimum [6]. Second, the pulse positions and amplitudes are scalar-quantized, thus requiring too many bits for their quantization. These limitations are overcome in the stochastic-excited LP coder by using the concepts of codebook coding (or vector quantization) for quantizing the excitation information.

2.1.4.3. Stochastic-excited LP coder: The stochastic-excited LP coder (also known as the code excited linear prediction (CELP) coder) [7] uses a stochastic model (shown in Fig. 3a) for synthesizing speech. Here the excitation is represented in terms of a sequence of random numbers. The procedure for estimating the optimum excitation sequence is shown in Fig. 3b. Here, the speech signal is processed block-wise where each block has M speech samples. For each block, the pitch parameters (pitch period and amplitude) are computed in a closed-loop analysis [24] and the optimum excitation sequence is selected from a codebook of random number sequences to optimize the given fidelity criterion. The codebook contains N M-dimensional codevectors whose components are white Gaussian random numbers. Each codevector is scaled by gain factor, G, that remains constant for the given speech block. The scaled codevector is filtered by a cascade of the pitch synthesis filter, \(1/P(z)\), and the LP synthesis filter, \(1/A(z)\). The objective error is computed as the difference between the original and the synthetic speech signals. This error is further filtered by a frame-specific perceptual weighting filter, \(A(z)/A(z/\tau)\), which tries to exploit the human auditory masking properties [5]. In order to find the optimum excitation for the given speech block, all the N codevectors in the codebook are exhaustively searched and the optimum excitation sequence is selected as the codevector (along with its gain factor) which minimizes the total-squared value of the perceptually-weighted error.

(a)

| Stochastic excitation | \(1/P(z)\) | \(1/A(z)\) |

(b)

Speech

Codebook \(\rightarrow\) \(G\) \(\rightarrow\) \(1/P(z)\) \(\rightarrow\) \(1/A(z)\)

\[\text{Minimization} \rightarrow \frac{A(z)}{A(z/\tau)}\]

Figure 3: Stochastic excited LP coder.
Like the multipulse coder, performance of the stochastic coder depends on a number of parameters [26], such as frame duration, frame-update interval, block duration, order of the LP analysis and codebook size, N. But, the codebook size, N, affects it most. Typical values of these parameters are: frame duration=20 ms, frame-update interval=20 ms, LP analysis order=10, block duration=5 ms (i.e., M=40 at 8 kHz sampling rate), and N=256.

2.2. Speech coding standards

As mentioned earlier, speech coding has applications within telephone network, and is used for mobile (or cellular) telephony and secure voice transmission. A number of speech coding standards have been created in the last few years by the standardization agencies around these applications. These standards are listed in Table 3.

Table 4: Speech coding standards (after [27]).

<table>
<thead>
<tr>
<th>Standard</th>
<th>Bit-rate</th>
<th>Year</th>
<th>MOS</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711 PCM</td>
<td>64 kb/s</td>
<td>1972</td>
<td>4.3</td>
<td>network</td>
</tr>
<tr>
<td>G.722 ADPCM</td>
<td>32 kb/s</td>
<td>1988</td>
<td>4.1</td>
<td>network</td>
</tr>
<tr>
<td>G.728 LD-CELP</td>
<td>16 kb/s</td>
<td>1992</td>
<td>4.1</td>
<td>network</td>
</tr>
<tr>
<td>G.729 CS-ACELP</td>
<td>8 kb/s</td>
<td>1995</td>
<td>4.1</td>
<td>multimedia</td>
</tr>
<tr>
<td>G.723.1 MFC-MLQ</td>
<td>5.3, 6.4 kb/s</td>
<td>1996</td>
<td>4.0</td>
<td>multimedia</td>
</tr>
<tr>
<td>RPE-LTP (GSM)</td>
<td>13 kb/s</td>
<td>1987</td>
<td>3.7</td>
<td>cellular</td>
</tr>
<tr>
<td>IS-54 VSELP (TIA)</td>
<td>8 kb/s</td>
<td>1990</td>
<td>3.7</td>
<td>cellular</td>
</tr>
<tr>
<td>FS-1015 LPC-1E</td>
<td>2.4 kb/s</td>
<td>1984</td>
<td>2.5</td>
<td>secure voice</td>
</tr>
<tr>
<td>FS-1016 CELP</td>
<td>4.5 kb/s</td>
<td>1991</td>
<td>3.0</td>
<td>secure voice</td>
</tr>
<tr>
<td>MELP</td>
<td>2.4 kb/s</td>
<td>1997</td>
<td>2.5</td>
<td>secure voice</td>
</tr>
</tbody>
</table>

The International Telecommunications Union (ITU) has created for the telephone network application the following three coders: the 64 kb/s pulse code modulation (PCM) coder, the 32 kb/s adaptive differential pulse code modulation (ADPCM) coder, and the 16 kb/s low delay code excited linear prediction (LD-CELP) coder [28]. For cellular application, the 13 kb/s regular pulse excited (RPE) LP coder [29, 30] has been standardized by the Group Speciale Mobile (GSM), Europe, and the 8 kb/s CELP coder [31] by the Telecommunications Industry Association (TIA), USA. The national security agency (NSA), U.S. Government, has standardized for secure voice application the following three coders: the 2.4 kb/s linear predictive coding (LPC) vocoder [33], the 4.8 kb/s CELP coder [32] and the 2.4 kb/s MELP coder [34]. Note that all these coders, except for the PCM coder, belong to the class of LP-based coders, described in the preceding section.

2.3. Current trends in low bit rate speech coding

Low bit rate speech coding research has recently gained a new momentum due to the increased interest in digital voice transmission for mobile and personal communications. This interest is reflected by the exponential growth in the number of customers over the last five years using these services. A number of new speech coding standards have been developed to satisfy customer demands. Most of the standard coders are based on either the CELP or its closely related algorithms. The CELP coders can result in high quality speech at bit rates higher than 6 kb/s. However, their performance deteriorates rapidly for bit rates lower than 6 kb/s. At 4 kb/s, the quality of coded speech from a CELP coder is not acceptable for the mobile radio application.

In CELP coding, pitch filter is always used whether the speech frame is voiced or unvoiced. During a voiced frame, it does not make much sense to transmit the information about all the pitch cycles present in the frame. Instead, it will be advantageous to transmit information only about the first pitch cycle and generate the rest of the pitch cycles in the voiced frame by periodic extension (or interpolation). Also, it is obviously not necessary to use the pitch filter during the unvoiced frames. These properties have been used recently to reduce the bit rate the CELP coder to 4 kb/s [35, 36, 37, 38]. In the new coder, a voiced/unvoiced decision is made first about the given frame. If the frame is voiced, only the
first pitch cycle of the LP residual waveform is transmitted. The other cycles are generated at the receiving end by periodic interpolation. For the quantization of the first pitch cycle, the analysis-by-synthesis procedure of the CELP coder is used. Note that the codebook excitation is not used during the voiced frames. During the unvoiced frames, no information is sent about the pitch filter. Only the codebook excitation (computed through the analysis-by-synthesis procedure of the CELP coder) is transmitted during the unvoiced frames. Quantization of pitch cycle waveform can be done either in the time domain [35] or in the frequency domain [37, 38]. It has been shown that the resulting 4 kb/s coder delivers the coded speech with quality comparable to that of the TIA’s 8 kb/s CELP standard.

Recently, Kleijn et al. [39] have proposed a generalized analysis-by-synthesis procedure for estimating the parameters of the codebook excitation and the pitch filter in a CELP coder. In this procedure, a multitude of modified speech signals is generated, with the constraint that each of these signals is perceptually close or identical to the original speech signal. The speech coder performance is evaluated for each of these modified signals, and the modified signal that results in the best coding performance is selected. Parameters of codebook excitation and pitch filter that correspond to this modified signal are transmitted to the receiver. This procedure has been applied to transmit pitch filter parameters at a much lower rate than the conventional approaches, without compromising the speech quality [39].

Most of the currently available speech coders perform well for clean speech spoken by a single speaker. When the speech signal is corrupted by another person’s speech (cocktail party effect) or by ambient noise (e.g., noise picked by cellular phone in a moving car), these coders do not perform well. Current challenge is to make these coders robust to such type of distortions. Another challenge comes from the use of packet-switched networks which are becoming increasingly popular for speech transmission. Speech coders have to be tuned to these network environments. This means that they have to operate in a variable bit-rate mode and must be adaptive to changing network conditions. In addition, they should be able to handle problems due to loss of packets (which happens quite often during high traffic on the network).

2. Lossy image coding (after [40])

Lossy compression techniques involve some loss of information. An image that has been compressed by lossy coding techniques cannot be reconstructed exactly from its compressed image. As mentioned in the preceding section, the LP-based coders are very successful for lossy speech compression. These coders have been tried for lossy image coding, especially in earlier days. Differential pulse code modulation (DPCM) coder, developed by Cutler in in early 50’s, is one example of these types of coders. Since the image signal does not follow the all-pole (or, LP) model very well, these coders do not perform well for lossy images and are not very fashionable these days.

The transform coding techniques, introduced in early 70’s, have displaced the LP-based techniques for lossy image coding and are still quite popular. In these techniques, the image signal is decomposed into a number of components using a linear transform. The linear transform is designed in such a way that it satisfies the following two criteria: 1) It should produce decorrelated components, and 2) It should pack the energy of the signal into as few components as possible. Energy compaction criterion helps in improving coding efficiency by allowing for adaptive bit allocation. The decorrelation criterion ensures that quantization distortion does not leak from one component to another. Though the Karhunen-Loeve transform (KLT) is the optimum energy-packing orthonormal transform, it is not very convenient as it changes from image to image, and it is computationally very expensive. The discrete cosine transform (DCT) is an orthonormal transform having energy-packing efficiency close to the KLT for the image signals. A number of fast algorithms exist for computing the DCT efficiently. In addition, the DCT converts the image signal into a form where perceptual properties of the human visual system can be easily incorporated. Because of these reasons, the DCT is the most popular transform used in transform coding techniques. The current standard for lossy image coding, JPEG [62], is based on transform coding technique. It utilizes the DCT for the decomposition of image signal.

Currently, the subband coding techniques have taken over the transform coding techniques for image coding in terms of popularity as they provide more general framework. Transform coding can be considered as a special case of subband coding. In subband coding, a bank of quadrature mirror filters is used to decompose the image signal into a number of frequency bands. These filters need not have uniform bandwidths. Nonuniform bandsplitting provides a multiresolution representation of an
image. When octave band decomposition is used, the resulting subband coder is referred to as a wavelet coder [41]. Beside excellent compression, these subband coders provide the successive approximation capability; as higher frequency components are added at the reconstruction stage, better-quality images are obtained. Coding efficiency of wavelet coders can be improved by using the embedded zero-tree (EZW) algorithm [42]. This algorithm improves the compression performance by assuming that if a coefficient is zero at a low frequency, it is highly likely that all the coefficients at the same spatial location at all higher frequencies will also be zero. Thus, this algorithm allows the coder to discard the whole tree of coefficients in higher frequency bands if the root node at lower frequency band is zero.

Subband coding [43] has proven to be an efficient method of coding images at low bit-rates. In subband coding, the image is first decomposed into a number of critically sampled subbands and then quantized and transmitted to the decoder. In a subband decomposed image, the different subbands usually contain vastly different amounts of energy. This property of subbands is taken advantage of in coding. The bands which contain more energy are quantized using a finer quantizer and those bands which contain less energy are quantized more coarsely.

The choice of fine and coarse quantizers corresponds with the number of bits used by each quantizer. Usually an optimization algorithm [45] is used to allocate bits to each subband according to the energy in that subband and the rate-distortion characteristics of the quantizers being used. In effect, the optimization algorithm minimizes the mean-squared-error (MSE) of the reconstructed image for a given overall bit-rate. In this fashion the non-uniform distribution of energy across the subbands is used to achieve compression. However, a close examination of a typical subband decomposed image reveals that the spatial distribution of energy within the subbands is also far from uniform.

Most of the energy within subbands is confined to areas corresponding to edges and strong textures in the original image. This non-uniformity within the subbands can be exploited to make coding more efficient. Chen and Smith [63] proposed such a scheme for discrete cosine transform (DCT) based coding of images. In their scheme, the image is divided into a number of equal-sized square blocks which are classified according to their energy. Each DCT coefficient within each class is then assigned a number of bits according to the average energy of the particular transform coefficient in that class and the overall bit budget. The Chen-Smith type of classification can be easily adapted for use in a subband coder. Each subband is divided into a number of equal-sized blocks which are classified according to their energies. An optimization algorithm is then used to select an appropriate quantizer for each class of each of the subbands. In the Chen-Smith type classification, the classes are chosen such that they are equally populated. However, this is non-optimal and recently better classification schemes have been devised [48], [50]. In [46], Joshi et al. provide a thorough study of a number of classification schemes for subband coding.

Although classification provides considerable coding gain, this gain comes at a cost. The decoder needs to be made aware of the classification information. This is normally done by transmitting a classification table which indicates the classes to which the blocks of subband samples belong. Using smaller sized blocks results in a higher coding gain, but it also increases the amount of classification information which needs to be sent along. The choice of an appropriate block size is a trade-off between the coding gain resulting from classification and the amount of classification information. Several methods for the reduction of classification information have been proposed in the literature [46]. However, at low bit-rates, the classification information can still amount up to 20% of the total bit budget.

In our Signal Processing Laboratory, we have developed a lossy image coder which provides more efficient classification by using smaller blocks where required (in areas of high activity) and larger blocks in other areas. We describe below this coder in more detail.

As mentioned previously, the energy in the subbands is not distributed uniformly. In a typical subband decomposed image, there are small areas of high activity which correspond to edges and strong textures, and large areas with little activity corresponding to the smoother areas in the original image. In our coder, we attempt to exploit this property by allocating smaller block sizes over the non-uniform (high activity) areas of the subbands and larger block sizes in the areas of uniformity (low activity). This added degree of adaptivity allows for more efficient classification of the subband samples for a given classification bit budget.

In the case of non-uniform block sizes, the decoder also needs to be made aware of the sizes and the locations of the blocks used. The quadtree structure [49] was selected as an efficient method of encoding the blocking scheme. In the following sections, we will describe the algorithms for generating
the quadtrees and the methods of encoding the quadtrees.

2.1. Generating the quadtrees

Similar to the binary-tree, the quadtree is a tree structure. However, instead of nodes branching off to two children as is the case for a binary tree, the quadtree’s nodes branch off to four children. In our application, the root of the quadtree corresponds to the entire image (or a particular subband) and each node which descends from the root corresponds to a square block within that image. The quadtrees used in this paper are not balanced and hence, to encode them one bit must be sent along for each node in the tree to indicate whether or not that node is to be split.

Now we will examine how the quadtrees are generated. The aim of our quadtree generation process is to split an image (or subband) into small subblocks, each of which have roughly uniform properties. The algorithm used in this case is based on growing the quadtree one step at a time, while splitting a block in each step of the growth. The choice of which block is to be split is made on the basis of an objective criterion, which depends on the degree of uniformity within the block as well as the size of the block. We will refer to this objective criterion as Splitting Gain (SG).

In this paper, we define the splitting gain in two different ways. In the first definition, we use the notion of the classification gain for a non-stationary source [46] and define the splitting gain (SG) as follows:

$$SG = \frac{N_p \sigma_p^2}{\prod_{i=1}^{4} (\sigma_i^2)^{1/4}}.$$  (7)

where $N_p$ is the number of samples in the parent block and $\sigma_p^2$ the sample variance of the parent block. The $\sigma_i^2$'s in this equation represent the variances of the four blocks which would be formed as a result of splitting the parent block. In the subsequent sections, we will refer to this definition of the splitting gain as Definition A.

Looking at the problem from a slightly different perspective, we may also define the splitting gain based on how well the energy of the sub-blocks is represented by their parent. That is, if the energy (standard deviation) of all subblocks is similar to that of the parent, we may wish to leave that particular block unsplit. On the other hand if some subblocks have energies which differ greatly from that of the parent block, we would wish to split the block. This definition is closer in line with that of conventional quadtree based image coding. In this case, the splitting gain is defined as follows:

$$SG = \frac{N_p}{4} \sum_{i=1}^{4} (\sigma_p - \sigma_i)^2.$$  (8)

We will refer to this definition as Definition B.

The algorithm for growing the quadtree is as follows:

1. Initialize the quadtree root to the entire image (or subband).
2. Split the root into 4 equal sized blocks.
3. Calculate the splitting gain (SG) for each block.
4. Split the block with the largest splitting gain into 4 blocks.
5. Calculate splitting gain for the new blocks.
6. Repeat from Step 4 until a maximum number of blocks is reached.

We should note that when quadtrees are utilized in a coder, the structure of the quadtree needs to be made known to the decoder. Hence, the number of the bits used to encode the quadtree is of some concern to us.

Once the quadtree has been generated using Definition A or Definition B, it is simply encoded using one bit for each node in the quadtree to indicate whether or not it has been split.
Since our algorithm for generating the quadtree is based on successive splitting of the blocks, it is appropriate to evaluate the length of the quadtree in terms of the number of block splits performed:

\[ \text{QuadtreeLength(\text{bits})} = 4N_s + 1. \]  \hspace{1cm} (9)

where \( N_s \) is the number of splits performed on the quadtree root.

In cases where a minimum block size has been set, the leaf nodes of that size will not require an additional bit since they are always left unsplit. Therefore (3) only sets an upper limit on the number of bits required to encode the quadtree.

### 2.2. Using the quadtrees in a subband coder

So far we have explained how quadtrees can be generated in order to split an image into blocks with uniform activity levels (as measured by variance or standard deviation). Now we will take a look at how this concept can be utilized in a classification based subband coder. The subband coder used in this paper relies on a 22-band decomposition as used in [46] which is shown in Fig. 4.

In [46], Joshi et al. have experimented with a number of classification schemes. The most successful of these schemes is based on classifying equal-sized blocks (2x2 for bands 0-6 and 4x4 for bands 7-21) into one of four classes within each subband. Two algorithms named Maximum Classification Gain and Equal Mean-Normalized Standard Deviation (EMNSD) are proposed for classification. They have shown that these algorithms perform almost equally well and they outperform the Chen-Smith [63] type of classification. Unlike Chen-Smith classification, both of these algorithms result in classes with unequal populations.

Joshi et al. ([46]) have also devised methods of reducing the classification information which needs to be sent along by exploiting various dependencies both between and within the classification maps of the subbands. However, despite the reductions, the classification information still comprises a large portion of the total bit-rate. With the use of the quadtree structures described in the previous section, we aim to reduce this overhead through a better (adaptive) choice of block sizes.

The simplest method of utilizing the quadtrees in this scheme would be to generate a quadtree for each subband and then perform the classification accordingly. However, in that case, the cost of encoding the quadtrees themselves can become prohibitive. A typical quadtree (with around 400-500 blocks) used for a subband can contribute around 0.002 bpp to the overall bit-rate (for a 512x512 pixel image). Thus encoding 22 such quadtrees would take up around 0.04 bpp which is quite expensive when the total bit budget is around 0.5 bpp or less.
The smallest four subbands (subbands 0-3) require at most 512 bits to classify them into 4 classes of 2x2 blocks. This is equivalent to a contribution of approximately 0.002 bpp to the total bit-rate which is hardly worthwhile attempting to reduce. On the other hand, the higher frequency subbands 10-21 usually contain very little energy and in our range of target bit-rates are mostly quantized to zero. The subbands which interest us the most are subbands 4-9 where the majority of the classification information is required.

We will examine 3 different methods for incorporating the quadtrees into the subbands classification:

- Method 1: Subbands 0-3 are divided into 2x2 sample blocks and classified. A single quadtree is generated on the original (512x512) image, and scaled down appropriately for use in subbands 4-21. The minimum block size in the quadtree is limited to 16x16 pixels which corresponds to 2x2 samples in subbands 4-6 and 4x4 samples in subbands 7-21 after appropriate down-scaling.

- Method 2: Design a quadtree for each of the subbands 4-9. Uniform sized blocks (2x2 samples) used for subbands 0-3. Subbands 10-21 will use the same quadtree as subbands 7, 8 or 9 depending on their orientation. That is, the diagonal bands (subbands 18 and 21) use the quadtree generated for subband 9. The vertical bands (14, 15, 16, 17 and 20) use the same classification map as subband 8 and so on.

- Method 3: Design a quadtree for each of the subbands 4, 5 and 6. Subbands 0-3 will be divided into 2x2 blocks as before. Subbands 7-21 will use scaled up versions of the quadtrees for subbands 4, 5 or 6 depending on their orientation (determined as in Method 2).

Figure 5 is an example of a quadtree (of size 421 blocks) generated from the original (512x512) Lena image by Method 1 using Definition A of splitting gain. The quadtree has been superimposed onto the original image to show where the splits have been made.

![Quadtree generated using Method 1A and superimposed onto the Lena image](image)

The subband coder used to demonstrate the quadtree based classification is very similar to the coders used in [46] and [47]. The quantizer used is the Arithmetic and Trellis Coded Quantizer (ACTCQ) described in [47]. At the heart of the ACTCQ system, lies a scalar quantizer with uniform thresholds. The codewords of the scalar quantizer are divided into a number of subsets corresponding to different states of the trellis. The Viterbi algorithm [54] is then used to choose the trellis path which minimizes the distance between the quantizer’s inputs and its outputs. An arithmetic coder is used to encode the trellis codewords. For a more detailed description of the ACTCQ system, refer to [47].

ACTCQ and other similar quantizers such as ECTCQ [51], have demonstrated excellent rate-distortion performance for the quantization of Generalized Gaussian (GG) sources. The performance of
these quantizers makes them ideal candidates for use in a subband coder. Operational rate-distortion curves for Generalized Gaussian sources with different shape parameters (in this case 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0) are generated and stored for subsequent use by the bit allocation algorithm.

Once the subbands are divided into blocks (using quadtrees or equal-sized blocks), the blocks within each subband are classified into four classes. The classification algorithm used is the Equal Mean-Normalized Standard Deviation (EMNSD) classification as described in [46]. Each of the classes in each subband is modeled as a realization of a Generalized Gaussian source whose variance and shape parameter are estimated.

An optimal bit-allocation algorithm is used to allocate the bit-budget among the classes in the different subbands. The bit allocation algorithm used in this paper is that of Westerink et al. [45]. This algorithm is a greedy (gradient based) algorithm which starts by allocating zero bits to all sources and then increases the bit-rate of the sources one at a time until the bit budget has been exhausted.

Once the bit-allocation is completed the classification maps are encoded and transmitted to the decoder. As described in [46], a number of methods are used to reduce the classification information, these methods can be summarized into the following 3 points:

1. The classification maps of subbands where all classes are allocated zero bits, need not be transmitted.
2. If more than one class in a particular subband has been allocated zero bits then these classes can be combined together into one class.
3. The classification tables are entropy coded using conditional probabilities. The symbol probabilities are conditioned on the classification maps of other subbands as well as the class index of adjacent blocks. In this fashion, interband and intraband dependencies are exploited.

The final step in coding is the quantization of the subband samples. After the ACTCQ has completed the quantization of all subband samples, the encoded file sizes are measured and used to determine the bit rate of the coder.

2.3. Results
The subband coder described in the previous section is used to compare the performance of the various quadtree schemes described in section 2 with the performance of the system using equal-sized blocks as in [46]. The filters used in the subband coder are Antonini et al.'s 7-9 tap perfect reconstruction filter pair [52] and Johnston's 32D (32-tap) filter [53]. For target bit-rates below 0.3 bpp, the 7-9 tap filter pair provides better performance both in perceptual and PSNR terms. The short filter lengths result in less noticeable ringing around the edges at low bit-rates. However, the 32-tap filter gives slightly better results at around 0.3 bpp and higher. Thus, the subband filters are selected in each case depending on the target bit-rate.

Coding results at a bit-rate of 0.25 bpp for the Lena image (512x512 pixels, 256 grey levels) are listed in Table 5. We have observed that quadtrees of around 400-500 blocks lead to the best results in all methods and, hence, have used these sizes in this experiment. The allocation of the bit-rate between the classification and quantization has been "tweaked" so that the overall bit rate is as close as possible to the target bit-rate. Normally, this is not necessary; however, in this case, it is needed to enable meaningful comparisons to be made among the various methods.

The columns in Table 5 correspond to the different methods of using the quadtree (see section 2.2), while the rows correspond to the definition of splitting gain (SG) used in the generation of the quadtree (see section 2.1). It is clear that Definition A of the splitting gain produces better results regardless of how the quadtree being utilized. This is not surprising since this definition closely follows the concept of classification gain. It can also be observed that Method 3 of using the quadtree produces the best results. Method 3 is a good compromise between Method 1 (where one quadtree is generated for all subbands) and Method 2 (where a quadtree is generated for each of subbands 4,5,6,7,8 and 9).

Method 3A (quadrees generated according to Method 3 and Definition A of the splitting gain) results in a PSNR improvement of almost 0.3 dB over the uniform block-size scheme of [46], which is a small but significant improvement. We should also note that in Method 3A the classification overhead (including the quadtrees) amounted to slightly over 0.02 bpp which is about half of that reported in
<table>
<thead>
<tr>
<th></th>
<th>Method 1</th>
<th>Method 2</th>
<th>Method 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SG Definition A</strong></td>
<td>34.46</td>
<td>34.53</td>
<td>34.61</td>
</tr>
<tr>
<td><strong>SG Definition B</strong></td>
<td>34.31</td>
<td>34.38</td>
<td>34.44</td>
</tr>
<tr>
<td><strong>Uniform Sizes</strong></td>
<td></td>
<td></td>
<td>34.32</td>
</tr>
</tbody>
</table>

Table 5: PSNR (dB) for various methods of encoding Lena at 0.25 bpp

[46]. This reduction in the classification information effectively leaves the quantizers with more available bits.

As the bit-rate increases the improvement due to the use of quadtree becomes less significant. At around bit-rate of 0.50 bpp the advantage of Method 3A over the uniform block sized scheme is around 0.08 dB which is almost insignificant (37.96 dB PSNR for Method 3A compared to 37.88 dB [46] for uniform block sizes). At this bit-rate other quadtree based methods demonstrate almost no advantage over the equal block-sized scheme. The classification overhead for Method 3A at 0.5 bpp is 0.042 bpp which is 0.01 bpp less than the overheads for equal sized blocks. At bit-rates above 0.5 bpp the gain due to quadtrees continues to diminish as the classification information comprises an increasingly smaller portion of the total bit-rate.

3. Lossless image coding (after [55])

In lossless image compression, the original image can be exactly reconstructed from the coded image. Lossy compression techniques (described in the preceding section) are more popular as they can provide better compression efficiency. But they have the disadvantage that they lead to coding distortions (or artifacts) which are not tolerable in certain applications. In areas such as medical image coding and some satellite imaging for instance, coding artifacts can have potentially adverse consequences or can corrupt data which has been obtained at a great cost. It is in these areas that lossless image coding methods are utilized. In recent years, a great deal of research in image coding has concentrated on lossy methods. The development of better transforms, quantizers and particularly adaptivity in coders has resulted in significant advances in this area. An adaptive coder is able to adapt its workings to the local characteristics within the different regions of an image.

Use of "context" [56] information to provide adaptivity has demonstrated significant improvements in lossless compression results. However, many of these techniques are based on heuristics and often do not fully exploit the gains offered by adaptive coding.

Figure 6 depicts a simple lossless image coder. In its simplest form, this coder is implemented using a fixed predictor and a fixed entropy coder. The predictor makes a prediction for each pixel based on previously transmitted values. The predicted values are then subtracted from the original values, thus leaving the prediction error (or residual). An entropy coder is then used to encode the residual signal.

![A DPCM-based lossless coder](image)

Figure 6: A DPCM-based lossless coder

Adaptivity can be provided by making the predictor and/or the entropy coder adaptive. An adaptive predictor is able to adapt to the characteristics of various textures or the direction of edges and hence produce more accurate predictions. An adaptive entropy coder, on the other hand, can adjust its workings to better match the local statistics of an image and produce shorter-length codes.
3.1. Context classification and adaptive prediction

In this paper, we concentrate on one particular type of adaptive prediction which is performed by switching among a finite number of predictors. The decision of which predictor to use is made for each pixel being encoded and is solely based on previously transmitted pixel values.

The difficulty with this type of adaptive prediction schemes is in the trade-off that exists between the number of predictors and computational complexity. Since the appropriate predictor is often selected using an exhaustive search [58] the computational complexity is high and can only be reduced by limiting the number of predictors. We have previously presented an optimal scheme for the design of a small number of predictors [58]; however, this scheme requires a computationally intensive design process.

The main performance obstacle in the path of the adaptive prediction schemes mentioned above is the exhaustive search required to map the local context from pixel domain to an appropriate predictor. The performance can be significantly improved if a faster mapping is used. Such a mapping has been recently proposed by Wu and Memmon [59] [64]. The coder named CALIC [64], has many distinct features which contribute to its state-of-the-art performance; however its most significant (and possibly understated) feature is its Context Classification (or Quantization).

In CALIC, a fast method is used to classify the causal neighborhood (context) of the pixel to be encoded into a finite number of contexts. For each of the possible contexts, a bias value (a scalar) is maintained which is added to the prediction made by a less sophisticated linear predictor in order to improve its performance. CALIC assumes that the predictor is consistently repeating a similar prediction error over the same context which can be compensated for by adding the bias value.

In this paper, we take a slightly different approach and set out to exploit this fast mapping to the fullest. We do so by using context classification to select a linear predictor from a set of 1024 linear predictors. In this fashion, a pixel-by-pixel decision is made as to which linear predictor should be used for predicting that particular pixel.

As done in [59], we define the prediction context of a pixel $X$ (see Fig. 7) as:

$$C(X) = (w, ww, nw, n, nn, ne, 2n - nn, 2w - ww).$$  \hfill (10)

where each of the elements is the pixel in that direction (north, south, ...) with respect to the pixel being encoded. The two values $2n - nn$ and $2w - ww$ do not correspond to actual pixel values. These values are calculated from the values of pixels $n, nn, w$ and $ww$ and are used to provide some measure of the gradient of intensity within the context. For the purpose of notation, we also index the values in $C(X)$ such that $C_1(X) = w$, $C_2(X) = ww$, ..., $C_8(X) = 2w - ww$.

Next, we calculate the value $\mu$ which is the mean of all pixels in the context of $X$:

$$\mu = \frac{1}{8} \sum_{i=1}^{8} C_i(X).$$  \hfill (11)

Using the value $\mu$ as a reference point, we are now in a position to generate the 8-bit, binary string $S(X)$ which defines the shape or the texture of the context:

$$S(X) = (S_1(X), S_2(X), ..., S_8(X)).$$  \hfill (12)

where,

$$S_i(X) = \begin{cases} 0, & \text{if } C_i(X) \geq \mu, \\ 1, & \text{otherwise}. \end{cases} \quad \text{for } i = 1, 2, ..., 8.$$  \hfill (13)

The 8-bit binary string $S(X)$, can also be interpreted as a binary number ranging between 0 and 255. We use this number as an index for the classification of the context $C(X)$. This index classifies each context according to its texture, gradient, edges and so on. This type of classification is arguably the key factor in the success of CALIC. However, rather than using the Gradient Adjusted Predictor (GAP) used in CALIC to produce a reference value, we have used the mean value $\mu$ which is easier to calculate.

So far, the shape of the context has been classified into one of 256 different shapes. However, since in calculating $S(X)$, all magnitude information has been discarded and only sign information has been retained, $S(X)$ does not contain any information about the strength of textures, steepness of gradients
or the sharpness of edges. In order to incorporate some of this information into the context classification, we also calculate a separate index corresponding to the level of activity (or energy) within the context.

To calculate the energy index, we first need to define some measure of the energy within the context. We do so by calculating the standard deviation of the values in each context:

\[ \sigma = \sqrt{\frac{1}{8} \sum_{i=1}^{8} [C_i(X) - \mu]^2} \]  

(14)

The standard deviation value \( \sigma \) is then quantized into one of 4 levels using Lloyd-Max [65] scalar quantizer. This quantizer is designed once using data from a set of training images and is kept for all future usage. In this fashion, the quantization index \( q(X) \in \{1, 2, 3, 4\} \) is found for the context of each pixel to be predicted. This index provides the activity information which is not contained in the index \( S(X) \) calculated previously. The two indices complement each other in terms of information and their combination is used to produce an effective classification for each context:

\[ I(X) = q(X) \cdot S(X). \]  

(15)

In this way, \( I(X) \) becomes an integer between 0 and 1023 which is used as the final classification index for each context. We design an optimum 4-th order linear predictor for each of the 1024 contexts using either training data or the image to be encoded. In coding, the context of each pixel is classified and then the appropriate predictor is used for the prediction of that particular pixel. It should be noted that the predictors for each context are designed using a Mean-Squared-Error (MSE) criterion.

3.2. The minimum-entropy clustering algorithm

The adaptive prediction scheme described in the previous section, decorrelates neighboring pixels in the image. However, upon inspecting the residual image, it becomes clear that large values of residuals (prediction error) tend to be confined to certain areas while small residual values are also grouped together.

This non-uniformity in the local statistics of the areas within the residual image can be exploited in entropy coding. This is done through using multiple entropy coders, each of which is matched to a particular type of region in the residual image. The question remains as to how the different areas within the image can be classified into different types and how the entropy coders can be designed. To this end, we utilize the Minimum-Entropy Clustering (MEC) [60] algorithm.

Our aim in the use of the Minimum-Entropy Clustering algorithm is to classify blocks of samples and then encode the samples in each block using a suitable entropy coder. The classification and design of the entropy coders should be performed in a way such that the overall entropy is minimized.

In order to design a coding system with \( N \) entropy coders, the MEC algorithm operates as follows:

1. **Initialization**: \( N \) probability distribution functions (PDF's) are defined. These PDF's will define the initial classes.

2. **Minimum-Code-Length classification**: Each block of samples \( B = (b_0, b_1, ..., b_{m-1}) \) is classified as belonging to class \( C \) such that the code length (after entropy coding) \( L = -\sum_{i=0}^{m-1} \log_2 p(b_i|C) \) is minimized. This is similar to a nearest neighbor selection in VQ design.
3. **Re-Estimate class statistics**: Estimate the new class PDF's. This will ensure that the class statistics are matched to those of the samples in that class and hence the entropy is further reduced. This step is similar to a centroid calculation in VQ design.

4. **Iteration**: Stop if a maximum number of iterations is reached or the classes have converged. Otherwise, go to step 2.

Steps 2 and 3 form the core of the MEC algorithm. In each of these two steps, the overall entropy is reduced. There are a number of choices available for the initial classification. However, a better definition of the initial classes can reduce the number of iterations required. Since we wish to group together blocks with similar statistics, a reasonable choice for initial classification would be classification based on the variance of the blocks. To do so, we choose a classification similar to that used by Chen and Smith [63]. The variance of each block is estimated and the blocks are sorted in the order of increasing (or decreasing) variance.

Using the sorted list, \(m-1\) threshold values (of variance) are selected and used to classify the blocks into \(m\) classes. After this classification, the class PDF's are estimated and used to define the initial classes.

This algorithm may be used in either a parametric or a non-parametric form. In its parametric form, all distributions are modeled as Generalized Gaussian (GG) distributions [57] whose shape parameter and variance are estimated. In this case, step 3 of the algorithm becomes a maximum-likelihood estimation of the shape parameter and variance. The main disadvantage of using the parametric form of the MEC algorithm is the additional processing required in the calculation of probabilities (Step 2) and the parameter estimation (Step 3).

In the non-parametric version of the MEC algorithm, frequency tables are maintained for each of the classes. These frequency tables are updated in step 3 of the algorithm. After the MEC algorithm has converged, these frequency tables are used to design the entropy coders. It is for this reason that the frequency tables must also be known at the decoder. In the parametric case, the probabilities can be efficiently described to the decoder by transmitting two parameters (shape and variance) per class.

In the non-parametric version, the frequency tables must be explicitly transmitted. To reduce the amount of information, we may take advantage of the symmetry in the frequency tables. One half of each frequency table is quantized using 6-8 bits and then transmitted to the decoder. If adaptive entropy coders are used, the probability tables are quickly adapted to match the source statistics and any mismatches caused by quantization rapidly disappear.

There is a trade-off between two versions of the MEC algorithm mentioned above. The non-parametric version offers faster execution times at the expense of added side-information; although the parametric version is slower to run it is much more concise to transmit its parameters. However, in terms of compression the two versions of the MEC algorithm produce quite similar results. The results in this paper have been obtained using the non-parametric version of the MEC algorithm.

### 3.3. Lossless image coder with adaptive prediction and entropy coding

In this section, we examine the design of the adaptive predictor and the entropy coders used to encode the prediction residuals. As mentioned in the previous section, the predictors can be designed either for a training set of images or specifically for the image to be encoded. There is a trade-off between the two methods which parallels the trade-offs experienced in quantizer design.

If a training set of images are used, the training is performed off-line, however the predictors are not designed for the particular image and the predictors are slightly inferior in performance. Alternatively, if the predictors are designed for the image to be encoded, then some online training is required and the predictor coefficients must be transmitted to the receiver.

In this paper, we examine both of the above-mentioned scenarios and compare their performance. The training algorithm for the predictors is as follows:

1. For each pixel in the image (Training image, or image to be encoded) the context \(C(X)\) is found.

2. We calculate and remove average value \(\mu\) from the pixels in \(C(X)\).
3. Take the sign of each of the mean removed values as $+1$ and $-1$ to make an 8-bit binary string $S(X)$.

4. Calculate the standard deviation of the values in $C(X)$.

5. Quantize the standard deviation into one of 4 levels to find energy index $q(X)$.

6. The pixel is classified as belonging to 1024 contexts by combining $S(X)$ and $q(X)$.

7. Update autocorrelation values for that context.

8. Return to Step 1 (until the entire image has been processed)

The standard deviation values are quantized using a Lloyd-Max quantizer [65]. Once the classification has been completed and the autocorrelation values are finalized, the autocorrelation values are used to design a 4th-order optimum linear predictor for each context. If a particular context appears rarely or not at all (in the training set), then it is allocated a trivial second order predictor with two coefficients of 0.5.

It should also be noted that if the predictors are specifically designed for the image to be encoded, their coefficients must be transmitted as side information and contribute around 0.01 to 0.02 bpp to the overall bit-rate. All predictor coefficients are quantized using 10 bits per predictor coefficient.

It is interesting to note that within a typical image, only 100-300 of the possible 1024 contexts appear. This is a significant advantage with respect to the transmission of the predictor coefficients, since only a small portion of the predictor coefficients need to be transmitted.

The MEC algorithm was used to define a classification scheme and design the entropy coders. Classification was made on blocks of 8x8 pixels since it was experimentally found to produce the best results. The 8x8 blocks were classified into 16 classes and hence encoded using 16 different entropy coders. As mentioned previously, along with the classification scheme, the frequency tables for the entropy coders must also be transmitted to the decoder. For a 512x512 pixel image this results in an added bit-rate of 0.01 bpp. For a 256x256 pixel image this overhead increases to approximately 0.04 bpp.

<table>
<thead>
<tr>
<th>Coded Image</th>
<th>CCBAC-TS</th>
<th>CCBAC-SI</th>
<th>CALIC [59]</th>
<th>JPEG [62]</th>
</tr>
</thead>
<tbody>
<tr>
<td>barb</td>
<td>4.34</td>
<td>4.29</td>
<td>4.43</td>
<td>5.53</td>
</tr>
<tr>
<td>fruit</td>
<td>4.42</td>
<td>4.36</td>
<td>4.55</td>
<td>4.91</td>
</tr>
<tr>
<td>lena</td>
<td>3.94</td>
<td>3.90</td>
<td>4.05</td>
<td>4.64</td>
</tr>
<tr>
<td>man</td>
<td>4.38</td>
<td>4.35</td>
<td>4.43</td>
<td>4.93</td>
</tr>
<tr>
<td>goldhill</td>
<td>4.70</td>
<td>4.64</td>
<td>4.72</td>
<td>5.05</td>
</tr>
</tbody>
</table>

Table 6: A comparison of lossless compression results (quoted in bpp)

3.4. Results

The coding results are listed in Table 6. The Context Classification Based Adaptive Coder trained on a Training Set is referred to as CCBAC-TS and the coder trained on the Specific Image is referred to as CCBAC-SI. A set of ten images, excluding the training set was used for training the CCBAC-TS. The values quoted in the table are based on actual file sizes and include all overheads.

The results obtained from CALIC [59], were also provided for comparison. From Table 6, it is clear that both proposed methods significantly outperform CALIC which is considered to be the state-of-the-art in lossless image coding.

As expected, the image specific training algorithm CCBAC-SI outperforms the pre-trained predictors used in CCBAC-TS. What is surprising however, is the small difference between the two sets of results. This suggests that in many cases, the pre-trained predictors may be advantageous, since they require no “online” training.
The use of "off-line" training (using a training set of images), is even more appropriate when the coder is being used on satellite or medical images. For instance, if the coder is going to be used in encoding X-ray images, then the predictors can be trained using a suite of X-ray images.

Both of the tested coders outperform CALIC. We must, however, acknowledge the increased computational complexity of both coders compared to CALIC. Work is currently in progress toward further optimizing the speed of the presented algorithms.

The prediction scheme presented in this paper, demonstrates the advantages of using multiple predictors and the fast context classification scheme presented. The fact that the context classification scheme does not use an exhaustive search allows a large number of contexts to be utilized. The combination of this type of prediction with an adaptive entropy coding scheme such as the MEC algorithm was shown to produce some of the best results obtained in image coding to date.

Acknowledgements

We are thankful to Dr. B.S. Atal (AT&T Bell Labs., Murray Hill, NJ, USA) and Mr. F. Golchin for their collaboration in the research work reported in this paper. Mr. F. Golchin is also thanked for his help in preparing this paper.

References


